aws

Benutzer-Leitfaden

Amazon Neptune



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

Amazon Neptune: Benutzer-Leitfaden

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

Die Marken und Handelsmarken von Amazon dürfen nicht in einer Weise in Verbindung mit nicht von Amazon stammenden Produkten oder Services verwendet werden, die geeignet ist, Kunden irrezuführen oder Amazon in irgendeiner Weise herabzusetzen oder zu diskreditieren. Alle anderen Handelsmarken, die nicht Eigentum von Amazon sind, gehören den jeweiligen Besitzern, die möglicherweise zu Amazon gehören oder nicht, mit Amazon verbunden sind oder von Amazon gesponsert werden.

Table of Contents

Was ist Neptune? 1
Neueste Updates 4
Erste Schritte mit Neptune 76
Was ist eine Graphdatenbank? 76
Wozu dienen Graphen? 77
Graphdatenbank-Anwendungen 78
Graph-Abfragesprachen 82
Abfragebeispiele 82
Neptune-Online-Kurs 84
Eingehendere Informationen 84
Schneller Start mit CloudShell 85
Voraussetzungen 85
Schritt 1: Connect zu Ihrem Neptun-Cluster her 86
Schritt 2: Daten einfügen und abfragen 86
Nächste Schritte 87
Neptune mit Grafiknotizbüchern verwenden 88
Verwenden der Neptune-Workbench 89
Bereitstellen mit CloudFormation 94
Protokolle aktivieren CloudWatch 94
Lokales Hosting 96
Migration JupyterLab zu 4 97
Voraussetzungen 101
Lebenszyklus-Konfiguration 102
Synchronisieren aus einem Snapshot 102
Nach der Erstellung des Notebooks 103
Erstellen Sie das Notizbuch 103
Manuelles Einrichten Neptune Neptune-Notebooks 105
Notizbücher mit Neptun-Grafik und Magie 106
Variable Injektion 107
Allgemeine Abfrageargumente 108
%seed 110
%load 110
%load_ids 110
%load_status 110

%cancel_load ... 110
%status ... 111
%gremlin_status ... 111
%opencypher_status oder %oc_status ... 111
%stream_viewer ... 112
%sparql_status ... 112
%graph_notebook_config ... 112
%graph_notebook_host ... 113
%graph_notebook_version ... 113
%graph_notebook_service ... 113
%graph_notebook_vis_options ... 113
%statistics ... 114
%summary ... 114
%reset_graph ... 115
%get_graph ... 115
Verteilung in% ... 115
%%sparql ... 116
%%gremlin ... 117
%%opencypher oder %%oc ... 118
%%graph_notebook_config ... 119
%%graph_notebook_vis_options ... 120
%neptune_ml ... 120
%%neptune_ml ... 125
Visualisierung von Grafikdaten ... 127
Oberfläche der Registerkarte für Diagramme ... 127
Gremlin-Visualisierung ... 129
SPARQL-Visualisierung ... 130
Tutorials für die Visualisierung ... 131
Fehlerbehebung bei Notebooks ... 132
Desynchronisierung des Widget-Status ... 132
JupyterLab 4.x Verzögerung beim ersten Laden des Bildschirms ... 132
Probleme beim Zugriff auf Graph Explorer ... 133
Allgemeine Tipps ... 133
Neptune-Einrichtung ... 134
DB-Instance-Typen ... 135
Zuteilung von Instance-Ressourcen ... 135

t3 und t4g ... 136
r4-Instances ... 137
r5-Instances ... 137
r5d-Instances ... 138
r6g-Instances ... 138
r6i-Instances ... 138
x2g-Instances ... 139
x2iezn-Instances ... 139
x2iedn-Instances ... 139
r8g-Instances ... 139
r7g-Instances ... 140
r7i-Instances ... 141
serverless-Instances ... 142
Speichertypen ... 142
I/O—Optimierter Speicher ... 142
Neptun-Cluster erstellen ... 144
Voraussetzungen ... 145
Erstellen eines Clusters ... 150
Die VPC konfigurieren ... 153
Subnetze hinzufügen ... 154
VPC konfigurieren ... 155
Eine Sicherheitsgruppe erstellen ... 155
DNS in Ihrer VPC ... 157
Verbinden mit einem Cluster ... 157
Netzwerkkonnektivität ... 157
Überprüfen Sie die Verbindung ... 158
In der VPC ... 160
Außerhalb der VPC ... 162
Über ein privates Netzwerk ... 163
Öffentliche Endpunkte ... 163
Sicherung des Zugangs zu Neptune ... 174
IAM-Richtlinien ... 174
VPC-Sicherheitsgruppen ... 174
IAM-Authentifizierung ... 175
Zugreifen auf Grafikdaten ... 176
Tools ... 176

Abfragesprachen ... 177
Verwenden von Befehlszeilentools ... 178
Verwenden von Gremlin ... 182
Verwenden von openCypher ... 187
Verwenden von SPARQL ... 187
Laden von Daten ... 189
Überwachung des Neptun-Clusters ... 189
Fehlerbehebung für Neptun-Cluster ... 189
Neptune Serverlos ... 191
Anwendungsfälle für Serverless ... 191
Beschränkungen ... 192
Kapazitätsskalierung ... 193
Einstellung des Minimums ... 195
Festlegen des Maximums ... 195
Konfigurieren von Kapazitätseinstellungen ... 196
Zusätzliche Konfiguration ... 198
Gespeicherte Konfiguration ... 198
Festlegung von Aktionsstufen ... 198
Reader-writer Ausrichtung ... 199
Vermeiden Sie sehr große Timeout-Werte ... 200
Ihre Konfiguration optimieren ... 200
Serverless verwenden ... 201
Erstellung eines Serverless-Clusters ... 201
Konvertieren auf Serverless ... 202
Ändern des Kapazitätsbereichs ... 203
Eine Instance auf „Bereitgestellt“ ändern ... 204
Client-Konfiguration ... 204
Überwachen ... 204
Globale Neptun-Datenbanken ... 206
-Übersicht ... 206
Vorteile ... 207
Einschränkungen ... 208
Einrichtung einer globalen Datenbank ... 209
Konfigurationsanforderungen ... 209
Erstellen einer globalen Datenbank ... 210
Verwenden eines vorhandenen DB-Clusters als primärer Cluster ... 212

Hinzufügen einer sekundären Region ... 213
Herstellen von Verbindungen ... 214
Verwaltung einer globalen Datenbank ... 215
Entfernen eines Clusters ... 215
Löschen einer globalen Datenbank ... 216
Ändern einer globalen Datenbank ... 216
Notfallwiederherstellung in der Neptune-Datenbank ... 217
Trennen und Hochstufen ... 218
Verwaltete geplante Failover ... 220
Überwachung Neptune Neptun-Einsätzen ... 222
Übersicht über Neptune ... 224
Einhaltung von Standards ... 225
Einhaltung von Gremlin-Standards ... 225
Einhaltung von SPARQL-Standards ... 242
Einhaltung der OpenCypher-Spezifikationen ... 248
Diagrammdatenmodell ... 263
Verzeichnis ... 263
Indizierungsstrategie ... 264
Nachschlage-Cache ... 268
Anwendungsfälle für den Nachschlage-Cache ... 268
Verwenden des Caches ... 269
Transaktionssemantik ... 271
Isolationsstufen ... 271
Neptune-Isolationsstufen ... 272
Transaktionsbeispiele ... 280
Cluster und Instances ... 285
Primäre DB-Instance ... 285
Read-replica Instanzen ... 285
Dimensionieren von Instances ... 286
Überwachen von Instances ... 287
Speicher, Zuverlässigkeit und Verfügbarkeit ... 288
I/O—Optimierter Speicher ... 288
Zuteilung ... 289
Speicherfakturierung ... 289
Bewährte Methoden für Speicher ... 290
Zuverlässigkeit und hohe Verfügbarkeit ... 291

Endpunktverbindungen ... 293
Cluster-Endpunkte ... 293
Reader-Endpunkte ... 294
Instance-Endpunkte ... 295
Benutzerdefinierte Endpunkte ... 295
Überlegungen zu Endpunkten ... 296
Arbeiten mit benutzerdefinierten Endpunkten ... 298
Labor-Modus ... 302
Verwenden des Labor-Modus ... 302
OSGP-Index ... 303
Müllsammlung im Wörterbuch ... 304
Transaktionssemantik ... 304
Erweiterte Datetime-Unterstützung ... 305
StrictTimeoutValidation ... 305
AccurateQRCMemoryEstimation ... 305
Neptune-DFE-Engine ... 307
Steuern der DFE-Verwendung ... 307
Von der DFE-Engine ausgeführte Abfragen ... 308
DFE-Statistiken ... 310
Größenbeschränkungen ... 311
Status der Statistiken ... 311
Automatische Berechnung deaktivieren ... 315
Re-enable automatisch berechnen ... 317
Manuelles Generieren von Statistiken ... 319
Überwachen von Statistiken ... 321
IAM-Authentifizierung ... 323
Löschen von Statistiken ... 326
Häufige Fehler ... 328
Diagrammübersichts-API ... 329
Abrufen einer Diagrammübersicht ... 330
Der Parameter `mode` ... 331
Eigenschaftsdiagrammübersicht ... 331
RDF-Diagrammübersicht ... 333
Beispiel für eine PG-Übersicht ... 334
Beispiel für eine RDF-Übersicht ... 338
IAM und Diagrammübersichten ... 342

Häufige Diagrammübersichtfehler ... 342
JDBC-Konnektivität ... 346
Erste Schritte ... 346
Verwenden von Tableau ... 347
Fehlerbehebung ... 349
Updates der Neptune-Engine ... 351
Ausnahmen und Wiederholungen ... 352
Warum Wiederholungslogik wichtig ist ... 352
Die Risiken wahlloser Wiederholungsversuche ... 352
Entwicklung einer praktischen Wiederholungsstrategie ... 353
Beispielszenarien ... 354
Überwachung und Beobachtbarkeit ... 356
Sicherheit ... 358
Upgrades des Amazon Neptune Neptune-Betriebssystems ... 359
Minimierung der Ausfallzeiten für Betriebssystem-Upgrades, die einen Neustart erfordern ... 360
Betriebssystem-Upgrades auf Ihre Neptune-DB-Instance anwenden ... 360
Schutz von Daten in Neptune-Datenbanken ... 362
Sicherung von Neptune-Datenbanken mit Amazon VPC ... 364
Verbindungen zu Neptune-Datenbanken verschlüsseln ... 364
Verschlüsselung von Daten im Ruhezustand ... 366
Authentifizieren mit IAM ... 370
Verschiedene Rollen ... 371
Verwenden von Identitäten ... 371
Aktivieren von IAM ... 373
Verbindung mit IAM-Authentifizierung herstellen ... 374
Voraussetzungen ... 376
Von der Befehlszeile aus eine Verbindung herstellen ... 377
Von der Gremlin-Konsole aus eine Verbindung herstellen ... 379
Verbindung mit Gremlin Java herstellen ... 382
Verbindung mit SPARQL Java herstellen ... 386
Herstellen einer Verbindung mit SPARQL und Node.js ... 389
Herstellen einer Verbindung mit Python ... 392
Gremlin-Python ... 396
Gremlin Javascript ... 397
Gremlin Go ... 399
Gremlin .NET ... 404

Verwenden von IAM-Richtlinien ... 407
Identity-Based Richtlinien ... 407
Service-Kontrollrichtlinien (SCP) ... 408
Zugriff auf die Neptune-Konsole ... 408
Anfügen einer Richtlinie ... 409
Arten von IAM-Richtlinien ... 409
Verwenden von Bedingungsschlüsseln ... 410
Unterstützung für IAM-Features ... 411
IAM-Richtlinien – Einschränkungen ... 412
Verwendung von verwalteten Richtlinien ... 412
Verwenden von Bedingungsschlüsseln ... 428
Erklärungen zur Verwaltungsrichtlinie für IAM ... 429
Benutzerdefinierte Richtlinienerklärungen zum Datenzugriff ... 455
Service-Linked Rollen ... 475
Rollenberechtigungen ... 476
Eine Rolle erstellen Service-Linked ... 478
Eine Rolle bearbeiten Service-Linked ... 478
Eine Rolle löschen Service-Linked ... 478
Temporäre Anmeldeinformationen ... 481
Holen Sie sich Anmeldeinformationen mit dem AWS CLI ... 482
Einrichten von Lambda ... 486
Einrichten von Amazon EC2 ... 487
Protokollieren und Überwachen ... 489
Schnittstellen-VPC-Endpunkte (AWS PrivateLink) ... 490
Überlegungen zur Einhaltung der Vorschriften ... 490
Aufbau robuster Bereitstellungen ... 491
Migrieren zu Neptune ... 492
Migration von Neo4j ... 493
Allgemeine Informationen ... 493
Vorbereitung der Migration ... 496
Bereitstellen der Infrastruktur ... 503
Datenmigrationen ... 506
Anwendungsmigration ... 513
Neptune-Kompatibilität ... 517
Cypher-Umschreibungen ... 522
Migrationsressourcen ... 530

Migration von TinkerPop ... 531
Migration von RDF ... 532
AWS DMS Wird zur Migration verwendet ... 533
Migration von Blazegraph ... 535
Neptune-Kompatibilität ... 535
Bereitstellen der Infrastruktur ... 536
Exportieren von Daten ... 537
Erstellen eines Amazon-S3-Buckets ... 539
Importieren der Daten ... 540
Laden von Daten ... 542
Neptune-Massen-Loader ... 542
IAM-Rolle und Amazon-S3-Zugriff ... 544
Datenformate ... 555
Beispiel für das Laden ... 570
Optimieren einer Massenladung ... 581
Loader-Referenz ... 583
Laden von Daten mit DMS ... 624
GraphMappingConfig ... 625
Replizieren zu Neptune ... 629
Daten mithilfe von Abfragen laden ... 634
OpenCypher: neptune.read () ... 635
SPARQL: LADEN und ENTLADEN ... 635
Gremlin: io () -Schritt ... 635
Speicher ... 637
Müllabfuhr im Wörterbuch ... 637
Inline-servergenerierte Edge-ID ... 639
Abfrage ... 642
Abfragewarteschlangen ... 643
Ermitteln der Anzahl der Abfragen in einer Warteschlange ... 643
Zeitüberschreitungen bei Abfragen ... 643
Plan-Cache abfragen ... 644
Wie erzwingt man die Aktivierung oder Deaktivierung des Abfrageplan-Caches ... 644
Wie kann man feststellen, ob ein Plan zwischengespeichert ist oder nicht ... 646
Räumung ... 648
Bedingungen, die dazu führen, dass der Plan nicht zwischengespeichert wird ... 648
Benutzerdefinierte queryId ... 649

Verwenden des HTTP-Headers. ... 650
Verwenden eines SPARQL-Abfragehinweises ... 650
Verwenden von queryId zum Prüfen des Status ... 650
Gremlin ... 651
Installieren der Gremlin-Konsole ... 654
HTTPS REST ... 660
Java ... 664
Python ... 677
.NET ... 680
Node.js ... 683
Go ... 685
AWS SDK ... 689
Abfragehinweise ... 693
Abfragestatus ... 704
Abfrageabbruch ... 709
Skriptbasierte Gremlin-Sitzungen ... 711
Gremline-Transaktionen ... 714
Streaming-Ergebnisse ... 718
Verwenden der Gremlin-API ... 723
Zwischenspeichern von Abfrageergebnissen ... 723
Effiziente Upserts ab 3.6.x ... 731
Effiziente Upserts vor 3.6.x ... 740
Gremlin `explain` ... 754
Gremlin und DFE ... 809
openCypher ... 812
Gremlin und openCypher ... 813
Verwenden von openCypher ... 813
Statusendpunkt ... 815
HTTPS-Endpunkt ... 823
AWS SDK ... 831
Verwenden des Bolt-Protokolls ... 835
Parametrisierte Beispiele ... 860
Datenmodell ... 863
openCypher `explain` ... 864
Transaktionen ... 891
Streaming-Ergebnisse ... 899

Abfragehinweise ... 905
Einschränkungen ... 913
Ausnahmen ... 913
Erweiterungen ... 920
Räumliche Daten ... 950
SPARQL ... 964
RDF4J-Konsole ... 965
RDF4J Workbench ... 968
Java ... 970
HTTP-API ... 974
Abfragehinweise ... 988
DESCRIBE und das Standarddiagramm ... 1006
Abfragestatus ... 1009
Abfrageabbruch ... 1011
Graph-Store-Protokoll ... 1013
SPARQL `explain` ... 1015
SPARQL-Erweiterung `SERVICE` ... 1048
Tools für die Visualisierung ... 1051
Graph-explorer ... 1051
Graph-explorer in einem Notizbuch ... 1052
Graph-explorer auf Fargate ... 1052
Demo ... 1055
Tom Sawyer Software ... 1056
Cambridge Intelligence ... 1057
Graphistry ... 1058
metaphacts ... 1059
G.V() Graph-Datenbank-Client ... 1060
Schlüssel-Features ... 1060
Verfügbarkeit und Updates ... 1062
Vorführung des Produkts ... 1062
Weitere Informationen ... 1062
Linkurious ... 1062
Linkurious Enterprise ... 1062
Ogma ... 1063
Graph.Build ... 1064
Graph.Build Studio: ... 1065

Graph.Build Transformatoren: ... 1065
Graph.Build Autoren: ... 1065
Exportieren von Daten ... 1067
Neptun-Export ... 1067
Neptune-Export Dienst ... 1068
Installieren des Services ... 1068
Aktivieren des Zugriffs auf Neptune ... 1072
Aktivieren Sie den Zugriff auf Neptune-Export ... 1073
Ausführen eines Exportauftrags ... 1073
Neptune-Export-Hilfsprogramm ... 1076
Voraussetzungen ... 1076
Ausführen von neptune-export ... 1078
Beispielbefehle ... 1079
Exportierte Dateien ... 1080
Export-Parameter ... 1081
command ... 1083
outputS3Path ... 1083
expectedBucketOwner ... 1084
jobSize ... 1084
params ... 1085
additionalParams ... 1085
params ... 1086
Filterbeispiele ... 1098
Fehlerbehebung ... 1102
Häufige Fehler ... 1104
Exportieren von Gremlin-Abfrageergebnissen nach Amazon S3 ... 1105
Beispiele ... 1106
Voraussetzungen ... 1106
Reaktionsformat ... 1108
Zugriff auf die Amazon S3 S3-Exportfunktion von Gremlin gewähren ... 1111
Verwalten von Neptune ... 1115
Neptun-Lösung Blue/Green ... 1117
Voraussetzungen für Neptune Blue/Green ... 1118
Wird verwendet CloudFormation , um die Lösung auszuführen ... 1119
Überwachen des Fortschritts ... 1120
Wechsel zum aktualisierten Cluster ... 1123

Bereinigen ... 1124
Best Practices ... 1124
Fehlerbehebung ... 1125
IAM-Benutzerberechtigungen ... 1126
Serviceverknüpfte Rollenrichtlinie ... 1126
Erstellen eines neuen IAM-Benutzers ... 1127
Parametergruppen ... 1129
Bearbeiten einer Parametergruppe ... 1131
Erstellen einer Parametergruppe ... 1132
Parameters ... 1134
neptune_enable_audit_log ... 1135
neptune_enable_slow_query_log ... 1135
neptune_slow_query_log_threshold ... 1135
neptune_lab_mode ... 1136
neptune_query_timeout ... 1136
neptune_streams ... 1137
neptune_streams_expiry_days ... 1137
neptune_lookup_cache ... 1138
neptune_autoscaling_config ... 1138
neptune_ml_iam_role ... 1139
neptune_ml_endpoint ... 1139
neptune_enable_inline_server_generated_edge_id ... 1139
neptune_dfe_query_engine ... 1140
neptune_query_timeout ... 1140
neptune_result_cache ... 1141
UndoLogPurgeConfig ... 1141
neptune_enforce_ssl ... 1141
Starten über die Konsole ... 1142
Stoppen und Starten eines Clusters ... 1150
Übersicht über Stoppen und Starten ... 1150
Stoppen eines Clusters ... 1151
Starten eines DB-Clusters ... 1152
Fast-Reset-API ... 1154
Verwenden von IAM-Auth ... 1157
Die Magics %db_reset ... 1158
Häufige Fehler ... 1159

Hinzufügen von Reader-Instances ... 1161
Erstellen einer Reader-Instance ... 1162
Ändern eines DB-Clusters ... 1164
Ändern einer Instance ... 1165
Leistung und Skalierung ... 1167
Speicherskalierung ... 1167
Skalieren von Instances; ... 1167
Skalieren von Lesevorgängen ... 1167
Auto Scaling ... 1169
Auto Scaling und Serverless ... 1171
Aktivieren von Auto Scaling ... 1171
Entfernen von Auto Scaling ... 1175
Clusterwartung ... 1176
Versionsnummern ... 1176
Arten der Veröffentlichung ... 1177
Lebensdauern der Engine-Version ... 1179
Verwalten von Engine-Updates ... 1181
Upgrade-Prozess ... 1186
Upgrade zu 1.2.0.0 oder höher ... 1188
Aktualisierung über CloudFormation ... 1190
1.2.0.1 auf 1.2.0.2 ... 1191
1.1.1.0 auf 1.2.0.2, Standard ... 1194
1.1.1.0 auf 1.2.0.2, benutzerdefiniert ... 1196
1.1.1.0 auf 1.2.0.2, gemischt ... 1198
Klonen eines DB-Clusters ... 1203
Einschränkungen ... 1205
Copy-on-Write Protocol (Protokoll) ... 1205
Löschen einer Quelldatenbank ... 1207
Verwalten von Instances ... 1208
Burstable-T3-Instances ... 1209
Ändern einer Instance ... 1212
Umbenennen einer Neptune-DB-Instance ... 1217
Neustarten einer DB-Instance ... 1219
Löschen einer DB-Instance ... 1221
Neptune-Streams ... 1224
Verwenden von Streams ... 1227

Aktivieren von Streams ... 1227
Deaktivieren von Streams ... 1228
Aufrufen der Streams-API ... 1228
Streams-Antwort ... 1230
Stream-Daten lesen ... 1233
Streams-Ausnahmen ... 1235
Streams-Datensatzformate ... 1237
PG_JSON ... 1237
RDF-NQUADS ... 1240
Beispiele für Streams ... 1241
Beispiele für AT_SEQUENCE_NUMBER ... 1241
Beispiel für AFTER_SEQUENCE_NUMBER ... 1242
Beispiel für TRIM_HORIZON ... 1243
Beispiel für LATEST ... 1243
Beispiel für Komprimierung ... 1245
Neptune-to-Neptune Einrichtung der Replikation ... 1246
Wählen Sie eine CloudFormation Vorlage ... 1247
Hinzufügen von Stack-Details ... 1249
Ausführen der Vorlage ... 1253
Aktualisieren des Stream-Pollers ... 1253
Streams zur Notfallwiederherstellung ... 1254
Einrichten der Replikation ... 1255
Weitere Überlegungen ... 1259
Neptun-Volltextsuche ... 1260
Full-text Einrichtung der Suche ... 1262
CloudFormation Vorlage ... 1263
Bestehende Datenbanken ... 1270
Aktualisieren des Pollers ... 1272
Anhalten und Starten des Pollers. ... 1273
OpenSearch Serverlos ... 1274
Für die AWS Lambda-Funktion des Pollers sind Änderungen erforderlich ... 1274
Für das Massenimport-Hilfsprogramm sind Änderungen erforderlich ... 1274
Erforderliche Änderungen an der IAM-Rolle, die für die Abfrage von Neptune verwendet wird ... 1275
Abfragen mit differenzierter Zugriffskontrolle ... 1275
Verwendung der Lucene-Syntax ... 1276

Neptune-Datenmodell für die Volltextsuche ... 1277
SPARQL-Beispieldokument ... 1278
Gremlin-Beispieldokument ... 1280
Parameter für die Volltextsuche ... 1281
Indizierung ohne Zeichenfolgen ... 1286
Aktualisieren eines vorhandenen Stacks ... 1287
Ausschließen von Feldern ... 1288
Datentypzuordnungen ... 1291
Validierung des Datentyps ... 1293
Beispielabfragen ... 1299
Full-text-search Ausführung von Abfragen ... 1301
Beispiele für SPARQL-Volltextsuchabfragen ... 1303
Übereinstimmungsabfrage ... 1303
prefix ... 1303
fuzzy ... 1304
term ... 1304
query_string ... 1304
simple_query_string ... 1305
nach Zeichenfolgenfeld sortieren ... 1305
nach Nicht-Zeichenfolgenfeld sortieren ... 1305
nach ID sortieren ... 1306
nach Etikett sortieren ... 1306
nach doc_type sortieren ... 1307
Lucene-Syntax ... 1307
Beispiele für Gremlin-Volltextsuchabfragen ... 1308
Grundlegende `match` ... 1308
`match` ... 1309
`fuzzy` ... 1309
`query_string` fuzzy ... 1309
`query_string` regex ... 1310
Hybrid-Abfrage ... 1310
Beispiel für eine Volltextsuche ... 1310
`query_string`, „+“ und „-“ ... 1311
`query_string`, `AND` und `OR` ... 1312
`term` ... 1313
`prefix` ... 1313

Lucene-Syntax ... 1313
Modernes Diagramm TinkerPop ... 1315
Sortieren nach Zeichenfolgenfeld ... 1315
Sortierung nach Nicht-Zeichenfolgenfeld ... 1316
Sortieren nach ID-Feld ... 1316
Sortieren nach Etikettfeld ... 1316
Sortieren nach `document_type`-Feld ... 1316
Fehlerbehebung und Metriken ... 1317
Fehlerbehebung bei Lesevorgängen ... 1318
Fehlerbehebung bei Schreibvorgängen ... 1318
Out-of-sync Probleme ... 1319
AWS Lambda Funktionen ... 1321
Gremlin-Verbindungen WebSocket ... 1321
Empfehlungen für Gremlin Lambda ... 1322
Empfehlungen für Schreibanforderungen ... 1323
Empfehlungen für Leseanforderungen ... 1324
Latenz bei Kaltstart ... 1325
Erstellen einer Lambda-Funktion ... 1325
Beispiele für Lambda-Funktionen ... 1328
Java-Beispiel ... 1329
JavaScript Beispiel ... 1334
Python-Beispiel ... 1339
Neptune Machine Learning ... 1344
Neptune-ML-Fähigkeiten ... 1344
Neptune-ML-Einrichtung ... 1346
Einrichtung mit CloudFormation ... 1348
Manuelle Einrichtung ... 1352
Mit dem AWS CLI ... 1361
Verwenden von Neptune ML ... 1365
Starten des Workflows ... 1365
Umgang mit sich entwickelnden Daten ... 1368
Aktualisieren der Modellartefakte ... 1368
Benutzerdefinierter Modell-Workflow ... 1370
Auswahl der Instance ... 1371
Für die Datenverarbeitung ... 1371
Für Modelltraining und Modelltransformation ... 1371

Für einen Inferenzendpunkt ... 1372
Datenexport ... 1373
Beispiele für Neptune-Export ... 1373
Einstellungen für Parameter ... 1376
AdditionalParams ... 1377
targets ... 1380
features ... 1387
Beispiele ... 1397
Datenverarbeitung ... 1413
Verwalten der Datenverarbeitung ... 1413
Verarbeiten von Aktualisierungen ... 1415
Feature-Kodierung ... 1418
Bearbeiten einer Trainingsdatendatei ... 1427
Modelltrainings ... 1439
Modelle und Training ... 1446
Anpassen von Hyperparametern ... 1450
Bewährte Trainingsmethoden ... 1464
Modelltransformation ... 1468
Inkrementelle Inferenzen ... 1468
Modelltransformation für Aufträge ... 1470
Modellartefakte ... 1473
Artefakte für verschiedene Aufgaben ... 1473
Generieren neuer Artefakte ... 1473
Maßgeschneiderte Modelle ... 1476
Übersicht über benutzerdefinierte Modelle ... 1477
Entwicklung kundenspezifischer Modelle ... 1484
Inferenzendpunkt ... 1489
Verwalten von Inferenzendpunkten ... 1489
Inferenzabfragen ... 1493
Gremlin-Inferenzabfragen ... 1494
SPARQL-Inferenzabfragen ... 1520
Neptune-ML-API ... 1527
Der Datenverarbeitungsbefehl ... 1528
Der Befehl modeltraining ... 1540
Der Befehl modeltransform ... 1556
Der Befehl endpoints ... 1570

Ausnahmen 1584
Einschränkungen 1585
SageMaker KI-Grenzen 1586
Überwachen von Neptune 1587
Instance-Status 1588
Beispielausgabe für 1591
Benutzen CloudWatch 1592
Verwenden der Konsole 1593
Mit dem AWS CLI 1593
Verwenden der CloudWatch API 1594
Überwachen der Instance-Leistung 1595
Neptune-Metriken 1596
Neptune-Dimensionen 1629
Prüfprotokolle mit Neptune 1630
Aktivieren von Prüfprotokollen 1630
Anzeigen von Prüfprotokollen 1630
Details zu Prüfprotokollen 1631
Neptun-Logs CloudWatch 1632
Logs in Logs veröffentlichen (CloudWatch Konsole) 1633
Auditprotokolle in Logs (CLI) veröffentlichen CloudWatch 1633
Logs mit langsamen Abfragen in Logs (CloudWatch CLI) veröffentlichen 1634
Überwachen von Protokollereignissen 1634
Notizbuch-Protokolle CloudWatch 1635
Slow-query Logs 1637
Anzeigen von Protokollen in der Konsole 1638
Slow-query Protokolldateien 1638
`info`-Modusattribute 1638
`debug`-Modusattribute 1642
Ausgabebeispiel 1644
Neptune-API-Aufrufe protokollieren mit AWS CloudTrail 1646
Neptun-Informationen in CloudTrail 1646
Grundlagen zu -Protokolldateieinträgen 1648
Ereignisse 1650
Kategorien und Nachrichten 1651
Anzeigen von Ereignissen 1665
Abonnieren von Ereignissen 1667

Verwalten von Abonnements ... 1668
Neptun-Ressourcen taggen ... 1669
Markieren – Übersicht ... 1669
Taggen in der Konsole ... 1672
Taggen mit der CLI ... 1673
Markierung über die -API ... 1673
Arbeiten mit ARNs ... 1675
Sichern und Wiederherstellen ... 1680
Sicherungen ... 1680
Fehlertoleranz ... 1681
Backup-Metriken ... 1683
Wiederherstellen von Daten ... 1684
Backup window ... 1685
Erstellen eines Snapshots ... 1686
Verwendung der Konsole ... 1686
Wiederherstellung aus einem Snapshot ... 1687
Wichtige Überlegungen zur Wiederherstellung ... 1687
Wiederherstellung ... 1689
Kopieren eines Snapshots ... 1691
Einschränkungen ... 1691
Aufbewahrung der Snapshot-Kopie ... 1692
Verschlüsselung ... 1692
Cross-Region Kopieren von Snapshots ... 1693
Kopieren eines Snapshots in der Konsole ... 1693
Kopieren eines Snapshots mit dem AWS CLI ... 1694
Freigeben eines Snapshots ... 1698
Verschlüsselte Snapshots ... 1699
Teilen ... 1702
Löschen eines Snapshots ... 1705
Verwenden der Konsole ... 1705
Mit dem AWS CLI ... 1705
Verwendung der Neptune-API ... 1705
Best Practices ... 1706
Grundlegende Anleitungen für den Betrieb ... 1708
Sicherheit ... 1710
Vermeiden Sie unterschiedliche Instance-Größen ... 1711

Vermeiden Sie Massenladeneustarts ... 1711
Wenn Sie viele Prädikate haben ... 1711
Vermeiden Sie lang laufende Transaktionen ... 1712
Verwenden von Metriken ... 1712
Optimieren von Abfragen ... 1713
Lastausgleich ... 1714
Verwenden einer temporären Instance ... 1714
Größenanpassung einer Instance ... 1715
„Task Interrupted“-Fehler ... 1715
Gremlin (allgemein) ... 1716
Heartbeat-Konfiguration ... 1717
Optimieren von Upsert-Abfragen ... 1719
Unterschiede bei der Ausführung ... 1720
Multi-Thread-Schreibvorgänge ... 1722
Bereinigen von Datensätzen ... 1724
`datetime( )` ... 1724
Systemeigenes Datum und systemeigene Uhrzeit ... 1725
Gremlin (Java-Client) ... 1727
Wiederverwenden des Client-Objekts ... 1728
Trennen von Clients für Lese- und Schreibvorgänge ... 1728
Mehrere Replikat-Endpunkte ... 1728
Schließen des Clients nach Abschluss des Vorgangs ... 1729
Neue Verbindung nach Failover ... 1729
Setzen Sie = maxInProcess PerConnection maxSimultaneousUsage PerConnection ... 1729
Senden von Abfragen als Bytecode ... 1730
Vollständige Nutzung der Abfrageergebnisse ... 1731
Massenweises Hinzufügen von Scheitelpunkten und Edges ... 1732
Deaktivieren von JVM-DNS-Caching ... 1733
Zeitüberschreitungen pro Abfrage ... 1733
Umgang mit einem `TimeoutException` ... 1735
openCypher und Bolt ... 1736
Erneute Verbindung nach Failover ... 1737
Erneutes Verwenden des Treiber-Objekts ... 1737
Umgang mit Lambda-Verbindungen ... 1738
Bevorzugung direktionaler Edges ... 1738
Keine gleichzeitigen Transaktionsabfragen ... 1739

Schließen von Treiberobjekten ... 1739
Verwenden expliziter Transaktionsmodi ... 1740
Logik für Wiederholversuche ... 1743
Verwenden der SET-Klausel ... 1746
Parametrisierte Abfragen verwenden ... 1747
UNWIND-Klasse ... 1748
Ausdrücke mit variabler Länge (VLP-Ausdrücke) ... 1749
Verwenden Sie detaillierte Beziehungsnamen ... 1750
Geben Sie Kantenbezeichnungen an ... 1751
Vermeiden Sie die WITH-Klausel ... 1752
Restriktive Filter ... 1753
Explizit Eigenschaften prüfen ... 1753
Vermeiden Sie den benannten Pfad ... 1754
Vermeiden Sie COLLECT (DISTINCT ()) ... 1755
Bevorzugen Sie die Eigenschaftenfunktion ... 1755
Statische Berechnungen ... 1756
Eingaben stapeln ... 1757
Bevorzugen Sie benutzerdefiniert IDs ... 1757
Vermeiden Sie ~id-Berechnungen ... 1759
Aktualisieren/Zusammenführen mehrerer Knoten ... 1759
SPARQL ... 1760
Abfrage aller benannten Graphen ... 1761
Angeben eines benannten Graphen für Load ... 1761
FILTER vs. WERTE ... 1761
Amazon ECS und Amazon EKS ... 1763
Lebenszyklus der Verbindung ... 1764
Austausch des Hosts ... 1765
Netzwerke und IAM ... 1767
AWS CLI und SDKs ... 1769
Neptune-Einschränkungen ... 1772
Regionen ... 1772
China-Regionen ... 1773
Asien-Pazifik (Hyderabad) ... 1773
Cluster-Volume-Größe ... 1773
Instance-Größen ... 1774
Pro Konto ... 1774

VPC erforderlich ... 1774
SSL erforderlich ... 1774
Availability Zones und Subnetzgruppen ... 1775
HTTP-Anforderungsnutzlast ... 1775
Gremlin ... 1775
Keine Nullzeichen ... 1775
SPARQL UPDATE LOAD ... 1775
Authentifizierung und Zugriff ... 1776
WebSockets Grenzwerte ... 1776
Eigenschaften und Bezeichnungen ... 1780
Bulk-loading ... 1780
Neptune-Integrationen ... 1781
Tools und Hilfsprogramme ... 1783
GraphQL-Hilfsprogramm ... 1783
Installation und Einrichtung ... 1785
Verwendung vorhandener Daten ... 1785
Verwenden eines Schemas ohne Direktiven ... 1786
Arbeiten mit Direktiven ... 1792
Command-line Argumente ... 1797
Nodestream ... 1801
Neptune-Fehler ... 1803
Es sind nicht genügend DB-Instances verfügbar ... 1803
Engine-Fehlercodes ... 1803
Fehlerformat ... 1804
Abfragefehler ... 1804
IAM-Fehler ... 1810
API-Fehler ... 1813
Loader-Fehler ... 1815
Engine-Versionen ... 1819
Planung der Lebensdauer von Engine-Versionen ... 1822
Veröffentlichung: 1.4.7.0 (03.03.2020) ... 1823
Features ... 1823
Verbesserungen ... 1824
Behobene Fehler ... 1824
Unterstützte Query-Language Versionen ... 1825
Upgrade-Pfade ... 1825

Wird geupgradet 1825
Veröffentlichung: 1.4.6.3 (18.12.2025) 1827
Die Fehler wurden behoben 1827
Unterstützte Versionen in Abfragesprache 1828
Upgrade-Pfade 1828
Wird geupgradet 1828
Veröffentlichung: 1.4.6.2 (18.11.2025) 1831
Fehler behoben 1831
Unterstützte Versionen in Abfragesprache 1831
Upgrade-Pfade 1832
Wird geupgradet 1832
Veröffentlichung: 1.4.6.1 (18.09.2025) 1834
Fehler behoben 1834
Unterstützte Versionen in Abfragesprache 1835
Upgrade-Pfade 1835
Wird geupgradet 1835
Veröffentlichung: 1.4.6.0 (02.09.2025) 1837
Neue Features 1838
Verbesserungen 1838
Fehler behoben 1838
Unterstützte Versionen in Abfragesprache 1839
Upgrade-Pfade 1839
Wird geupgradet 1839
Veröffentlichung: 1.4.5.1 (30.06.2025) 1842
Fehler behoben 1842
Unterstützte Versionen in Abfragesprache 1842
Upgrade-Pfade 1843
Wird geupgradet 1843
Veröffentlichung: 1.4.5.0 (09.04.2025) 1845
Neue Features 1845
Verbesserungen 1846
Fehler behoben 1847
Unterstützte Versionen in Abfragesprache 1848
Upgrade-Pfade 1849
Wird geupgradet 1849
Veröffentlichung: 1.4.4.0 (24.02.2025) 1851

Fehler behoben 1852
Unterstützte Versionen in Abfragesprache 1852
Upgrade-Pfade 1852
Wird geupgradet 1852
Veröffentlichung: 1.4.3.0 (21.01.2025) 1855
Neue Features 1855
Verbesserungen 1856
Fehler behoben 1857
Unterstützte Versionen in Abfragesprache 1857
Upgrade-Pfade 1857
Wird geupgradet 1857
Veröffentlichung: 1.4.2.0 (19.12.2024) 1860
Fehler behoben 1861
Unterstützte Versionen in Abfragesprache 1861
Upgrade-Pfade 1861
Wird geupgradet 1861
Veröffentlichung: 1.4.1.0 (21.11.2024) 1864
Neue Features 1864
Verbesserungen 1865
Fehler behoben 1866
Unterstützte Versionen in Abfragesprache 1867
Upgrade-Pfade 1867
Wird geupgradet 1867
Veröffentlichung: 1.4.0.0 (06.11.2024) 1869
Neue Features 1870
Verbesserungen 1871
Fehler behoben 1872
Unterstützte Versionen in Abfragesprache 1873
Upgrade-Pfade 1873
Wird geupgradet 1873
Veröffentlichung: 1.3.4.0 (2024-10-01) 1876
Verbesserungen 1877
Fehler behoben 1877
Unterstützte Versionen in Abfragesprache 1878
Upgrade-Pfade 1879
Wird geupgradet 1879

Veröffentlichung: 1.3.3.0 (05.08.2020) 1881
Fehler behoben 1882
Behebung von Problemen in Version 1.3.3.0 1882
Unterstützte Versionen in Abfragesprache 1883
Upgrade-Pfade 1884
Wird geupgradet 1884
Veröffentlichung: 1.3.2.1 (20.06.2020) 1886
Fehler behoben 1887
Die Änderungen in 1.3.2.1 wurden von 1.3.2.0 übernommen 1888
Upgrade-Pfade 1893
Wird geupgradet 1894
Veröffentlichung: 1.3.2.0 (10.06.2020) 1896
Verbesserungen 1897
Fehler behoben 1898
Behebung von Problemen in Version 1.3.2.0 1900
Unterstützte Versionen in Abfragesprache 1902
Upgrade-Pfade 1903
Wird geupgradet 1903
Veröffentlichung: 1.3.1.0 (06.03.2024) 1905
Verbesserungen 1906
Fehler behoben 1906
Unterstützte Versionen in Abfragesprache 1907
Upgrade-Pfade 1907
Wird geupgradet 1907
Release: 1.1.0.0 (15.11.2023) 1910
Neue Features 1910
Verbesserungen 1910
Behobene Fehler 1913
Unterstützte Versionen in Abfragesprache 1914
Upgrade-Pfade 1914
Wird geupgradet 1914
Veröffentlichung: 1.2.1.2 (05.08.2020) 1917
Behobene Fehler 1918
Unterstützte Versionen in Abfragesprache 1918
Upgrade-Pfade 1918
Wird geupgradet 1919

Veröffentlichung: 1.2.1.1 (11.03.2024) 1921
Verbesserungen 1922
Behobene Fehler 1922
Unterstützte Versionen in Abfragesprache 1923
Upgrade-Pfade 1923
Wird geupgradet 1924
Release: 1.2.1.0 (08.03.2023) 1926
Patch-Veröffentlichungen 1927
Neue Features 1927
Verbesserungen 1929
Behobene Fehler 1930
Unterstützte Versionen in Abfragesprache 1931
Upgrade-Pfade 1931
Wird geupgradet 1932
Release: 1.2.1.0.R7 (06.10.2023) 1934
Release: 1.2.1.0.R6 (12.09.2023) 1938
Release: 1.2.1.0.R5 (02.09.2023) 1942
Release: 1.2.1.0.R4 (10.08.2023) 1946
Release: 1.2.1.0.R3 (13.06.2023) 1950
Release: 1.2.1.0.R2 (02.05.2023) 1956
Release: 1.2.0.2 (20.11.2022) 1960
Patch-Veröffentlichungen 1961
Neue Features 1962
Verbesserungen 1962
Unterstützte Versionen in Abfragesprache 1962
Upgrade-Pfade 1963
Wird geupgradet 1963
Release: 1.2.0.2.R6 (12.09.2023) 1965
Release: 1.2.0.2.R5 (16.08.2023) 1969
Release: 1.2.0.2.R4 (08.05.2023) 1973
Release: 1.2.0.2.R3 (27.03.2023) 1976
Release: 1.2.0.2.R2 (15.12.2022) 1981
Release: 1.2.0.1 (26.10.2022) 1986
Patch-Veröffentlichungen 1987
Neue Features 1987
Verbesserungen 1987

Behobene Fehler ... 1988
Unterstützte Versionen in Abfragesprache ... 1988
Upgrade-Pfade ... 1988
Wird geupgradet ... 1988
Wartungs-Release: 1.2.0.1.R3 (27.09.2023) ... 1991
Wartungs-Release: 1.2.0.1.R2 (13.12.2022) ... 1995
Release: 1.2.0.0 (21.07.2022) ... 1999
Patch-Veröffentlichungen ... 2000
Neue Features ... 2001
Verbesserungen ... 2001
Behobene Fehler ... 2002
Unterstützte Versionen in Abfragesprache ... 2004
Upgrade-Pfade ... 2004
Wird geupgradet ... 2005
Release: 1.2.0.0.R4 (29.09.2023) ... 2008
Release: 1.2.0.0.R3 (15.12.2022) ... 2013
Release: 1.2.0.0.R2 (14.10.2022) ... 2018
Release: 1.1.1.0 (19.04.2022) ... 2024
Patch-Veröffentlichungen ... 2025
Neue Features ... 2025
Verbesserungen ... 2026
Behobene Fehler ... 2027
Unterstützte Versionen in Abfragesprache ... 2028
Upgrade-Pfade ... 2029
Wird geupgradet ... 2029
Release: 1.1.1.0.R7 (23.01.2023) ... 2032
Release: 1.1.1.0.R6 (23.09.2022) ... 2037
Release: 1.1.1.0.R5 (21.07.2022) ... 2043
Release: 1.1.1.0.R4 (23.06.2022) ... 2048
Release: 1.1.1.0.R3 (07.06.2022) ... 2053
Wartungs-Release: 1.1.1.0.R2 (16.05.2022) ... 2059
Release: 1.1.0.0 (19.11.2021) ... 2064
Patch-Versionen ... 2065
Neue Features ... 2065
Verbesserungen ... 2066
Behobene Fehler ... 2067

Unterstützte Versionen in Abfragesprache ... 2068
Upgrade-Pfade ... 2068
Wird geupgradet ... 2068
Wartungs-Release: 1.1.0.0.R3 (23.12.2022) ... 2070
Wartungs-Release: 1.1.0.0.R2 (16.05.2022) ... 2075
Release: 1.0.5.1 (01.10.2021) ... 2080
Patch-Veröffentlichungen ... 2080
Neue Features ... 2080
Verbesserungen ... 2081
Behobene Fehler ... 2081
Unterstützte Versionen in Abfragesprache ... 2082
Upgrade-Pfade ... 2082
Wird geupgradet ... 2082
Wartungs-Release: 1.0.5.1.R4 (16.05.2022) ... 2084
Release: 1.0.5.1.R3 (13.01.2022) ... 2087
Release: 1.0.5.1.R2 (26.10.2021) ... 2089
Release: 1.0.5.0 (27.07.2021) ... 2092
Patch-Veröffentlichungen ... 2092
Neue Features ... 2092
Verbesserungen ... 2093
Behobene Fehler ... 2095
Unterstützte Versionen in Abfragesprache ... 2095
Upgrade-Pfade ... 2095
Wird geupgradet ... 2095
Wartungs-Release: 1.0.5.0.R5 (16.05.2022) ... 2097
Release: 1.0.5.0.R3 (15.09.2021) ... 2100
Release: 1.0.5.0.R2 (16.08.2021) ... 2103
Release: 1.0.4.2 (01.06.2021) ... 2105
Release: 1.0.4.2.R5 (16.08.2021) ... 2106
Release: 1.0.4.2.R4 (23.07.2021) ... 2106
Release: 1.0.4.2.R3 (28.06.2021) ... 2107
Release: 1.0.4.2.R2 (01.06.2021) ... 2109
Release: 1.0.4.2.1 (27.05.2021) ... 2113
Release: 1.0.4.1 (08.12.2020) ... 2113
Patch-Veröffentlichungen ... 2113
Neue Features ... 2114

Verbesserungen 2114
Behobene Fehler 2114
Unterstützte Versionen in Abfragesprache 2115
Upgrade-Pfade 2115
Wird geupgradet 2115
Release: 1.0.4.1.R1.1 (22.03.2021) 2117
Release: 1.0.4.1.R2 (24.02.2021) 2120
Release: 1.0.4.0 (12.10.2020) 2126
Patch-Veröffentlichungen 2126
Neue Features 2126
Verbesserungen 2126
Behobene Fehler 2128
Unterstützte Versionen in Abfragesprache 2128
Upgrade-Pfade 2128
Wird geupgradet 2129
Release: 1.0.4.0.R2 (24.02.2021) 2131
Release: 1.0.3.0 (08.03.2020) 2134
Patch-Veröffentlichungen 2134
Neue Features 2134
Verbesserungen 2135
Behobene Fehler 2135
Unterstützte Versionen in Abfragesprache 2136
Upgrade-Pfade 2136
Wird geupgradet 2136
Release: 1.0.3.0.R3 (19.02.2021) 2138
Release: 1.0.3.0.R2 (12.10.2020) 2141
Release: 1.0.2.1.R6 (22.04.2020) 2145
Patch-Veröffentlichungen 2145
Verbesserungen 2145
Behobene Fehler 2146
Unterstützte Versionen in Abfragesprache 2146
Upgrade-Pfade 2147
Wird geupgradet 2147
Release: 1.0.2.2.R6 (19.02.2021) 2149
Release: 1.0.2.2.R5 (12.10.2020) 2152
Release: 1.0.2.2.R4 (23.07.2020) 2155

Release: 1.0.2.2.R3 (22.07.2020) 2158
Release: 1.0.2.2.R2 (02.04.2020) 2158
Release: 1.0.2.1 (22.11.2019) 2161
Patch-Veröffentlichungen 2161
Neue Features 2162
Verbesserungen 2162
Behobene Fehler 2163
Unterstützte Versionen in Abfragesprache 2163
Upgrade-Pfade 2163
Wird geupgradet 2164
Release: 1.0.2.1.R6 (22.04.2020) 2166
Release: 1.0.2.1.R5 (22.04.2020) 2169
Release: 1.0.2.1.R4 (20.12.2019) 2169
Release: 1.0.2.1.R3 (12.12.2019) 2172
Release: 1.0.2.1.R2 (25.11.2019) 2175
Release: 1.0.2.0 (08.11.2019) 2177
WICHTIG: DIESE ENGINE-VERSION IST INZWISCHEN VERALTET 2177
Patch-Veröffentlichungen 2178
Neue Features 2178
Unterstützte Versionen in Abfragesprache 2178
Upgrade-Pfade 2178
Wird geupgradet 2178
Release: 1.0.2.0.R3 (05.05.2020) 2180
Release: 1.0.2.0.R2 (21.11.2019) 2184
Release: 1.0.1.2 (10.06.2020) 2187
WICHTIG: DIESE ENGINE-VERSION IST INZWISCHEN VERALTET 2187
Verbesserungen 2187
Behobene Fehler 2187
Unterstützte Versionen in Abfragesprache 2187
Release: 1.0.1.1 (26.06.2020) 2188
WICHTIG: DIESE ENGINE-VERSION IST INZWISCHEN VERALTET 2188
Behobene Fehler 2188
Unterstützte Versionen in Abfragesprache 2188
Release: 1.0.1.0 (02.07.2019) 2188
WICHTIG: DIESE ENGINE-VERSION IST INZWISCHEN VERALTET 2188
Version 1.0.1.0.200502.0 (31.10.2019) 2188

Release 1.0.1.0.200463.0 (15.10.2019) 2189
Release 1.0.1.0.200457.0 (19.06.2019) 2190
Release 1.0.1.0.200369.0 (13.08.2019) 2191
Release 1.0.1.0.200366.0 (26.07.2019) 2192
Release 1.0.1.0.200348.0 (02.07.2019) 2195
Frühere Versionen 2195
Neptune verwenden APIs 2208
Geteilte IAM-Aktionen 2208
Management-API-Referenz 2215
Cluster 2222
Erstellen DBCluster 2222
Löschen DBCluster 2234
Modifizieren DBCluster 2242
Starten DBCluster 2253
Stopp DBCluster 2259
AddRoleToDBCluster 2265
RemoveRoleFromDBCluster 2266
Failover DBCluster 2267
PromoteReadReplicaDBCluster 2273
Beschreiben DBClusters 2279
_____ 2281
DBCluster 2281
DBClusterMitglied 2288
DBClusterRolle 2288
CloudwatchLogsExportConfiguration 2289
PendingCloudwatchLogsExports 2290
ClusterPendingModifiedValues 2290
Globale Datenbanken 2291
CreateGlobalCluster 2292
DeleteGlobalCluster 2295
ModifyGlobalCluster 2296
DescribeGlobalClusters 2299
FailoverGlobalCluster 2301
RemoveFromGlobalCluster 2303
_____ 2305
GlobalCluster 2305

GlobalClusterMember 2307
Instances 2307
Erstellen DBInstance 2308
Löschen DBInstance 2321
Modifizieren DBInstance 2327
Starten Sie neu DBInstance 2341
Beschreiben DBInstances 2346
DescribeOrderableDBInstanceOptionen 2348
DescribeValidDBInstanceÄnderungen 2350
_____ 2351
DBInstance 2351
DBInstanceStatusInfo 2356
Bestellbare Option DBInstance 2357
PendingModifiedValues 2359
ValidStorageOptions 2361
Gültig DBInstance ModificationsMessage 2361
Parameter 2361
DBParameterGruppe kopieren 2362
Kopieren DBCluster ParameterGroup 2364
DBParameterGruppe erstellen 2366
Erstellen DBCluster ParameterGroup 2368
DBParameterGruppe löschen 2371
Löschen DBCluster ParameterGroup 2371
DBParameterGruppe ändern 2372
Modifizieren DBCluster ParameterGroup 2374
DBParameterGruppe zurücksetzen 2376
Zurücksetzen DBCluster ParameterGroup 2377
Beschreiben DBParameters 2378
Beschreiben Sie DBParameter Gruppen 2380
Beschreiben Sie die DBCluster Parameter 2381
Beschreiben DBCluster ParameterGroups 2383
DescribeEngineDefaultParameters 2384
DescribeEngineDefaultClusterParameters 2386
_____ 2387
Parameter 2387
DBParameterGruppe 2388

DBClusterParameterGroup ... 2389
DBParameterGroupStatus ... 2390
Subnetze ... 2391
DBSubnetGruppe erstellen ... 2391
DBSubnetGruppe löschen ... 2393
DBSubnetGruppe ändern ... 2394
Beschreiben Sie DBSubnet Gruppen ... 2395
_____ ... 2397
Subnetz ... 2397
DBSubnetGruppe ... 2397
Snapshots ... 2398
DBClusterSnapshot erstellen ... 2399
DBClusterSchnappschuss löschen ... 2402
DBClusterSnapshot kopieren ... 2405
Modifizieren DBCluster SnapshotAttribute ... 2410
Wiederherstellen DBCluster FromSnapshot ... 2412
Wiederherstellen DBCluster ToPointInTime ... 2422
Beschreiben Sie Schnappschüsse DBCluster ... 2433
Beschreiben DBCluster SnapshotAttributes ... 2436
_____ ... 2437
DBClusterSchnappschuss ... 2437
DBClusterSnapshotAttribute ... 2440
DBClusterSnapshotAttributesResult ... 2440
--Ereignisse ... 2441
CreateEventSubscription ... 2442
DeleteEventSubscription ... 2445
ModifyEventSubscription ... 2447
DescribeEventSubscriptions ... 2450
AddSourceIdentifierToSubscription ... 2451
RemoveSourceIdentifierFromSubscription ... 2453
DescribeEvents ... 2455
DescribeEventCategories ... 2457
_____ ... 2458
Ereignis ... 2458
EventCategoriesMap ... 2459
EventSubscription ... 2459

Sonstige 2461
AddTagsToResource 2461
ListTagsForResource 2462
RemoveTagsFromResource 2463
ApplyPendingMaintenanceAction 2464
DescribePendingMaintenanceActions 2465
DBEngineVersionen beschreiben 2466
_____ 2468
DBEngineVersion 2468
EngineDefaults 2470
PendingMaintenanceAction 2470
ResourcePendingMaintenanceActions 2471
UpgradeTarget 2472
Markierung 2473
Datentypen 2473
AvailabilityZone 2474
DBSecurityGroupMembership 2474
DomainMembership 2474
DoubleRange 2475
Endpunkt 2475
Filter 2476
Bereich 2476
Serverless V2 ScalingConfiguration 2477
Serverless V2 ScalingConfigurationInfo 2477
Zeitzone 2478
VpcSecurityGroupMembership 2478
API-Störungen 2478
AuthorizationAlreadyExistsFault 2481
AuthorizationNotFoundFault 2481
AuthorizationQuotaExceededFault 2482
CertificateNotFoundFault 2482
DBClusterAlreadyExistsFault 2482
DBClusterNotFoundFault 2483
DBClusterParameterGroupNotFoundFault 2483
DBClusterQuotaExceededFault 2483
DBClusterRoleAlreadyExistsFault 2484

DBClusterRoleNotFoundFault 2484
DBClusterRoleQuotaExceededFault 2484
DBClusterSnapshotAlreadyExistsFault 2485
DBClusterSnapshotNotFoundFault 2485
DBInstanceAlreadyExistsFault 2485
DBInstanceNotFoundFault 2486
DBLogFileNotFoundFault 2486
DBParameterGroupAlreadyExistsFault 2486
DBParameterGroupNotFoundFault 2487
DBParameterGroupQuotaExceededFault 2487
DBSecurityGroupAlreadyExistsFault 2487
DBSecurityGroupNotFoundFault 2488
DBSecurityGroupNotSupportedFault 2488
DBSecurityGroupQuotaExceededFault 2488
DBSnapshotAlreadyExistsFault 2489
DBSnapshotNotFoundFault 2489
DBSubnetGroupAlreadyExistsFault 2489
DBSubnetGroupDoesNotCoverEnoughAZs 2490
DBSubnetGroupNotAllowedFault 2490
DBSubnetGroupNotFoundFault 2490
DBSubnetGroupQuotaExceededFault 2491
DBSubnetQuotaExceededFault 2491
DBUpgradeDependencyFailureFault 2491
DomainNotFoundFault 2492
EventSubscriptionQuotaExceededFault 2492
GlobalClusterAlreadyExistsFault 2492
GlobalClusterNotFoundFault 2493
GlobalClusterQuotaExceededFault 2493
InstanceQuotaExceededFault 2493
Unzureichend DBCluster CapacityFault 2494
Unzureichend DBInstance CapacityFault 2494
InsufficientStorageClusterCapacityFault 2494
Ungültig DBCluster EndpointStateFault 2495
Ungültig DBCluster SnapshotStateFault 2495
Ungültig DBCluster StateFault 2495
Ungültig DBInstance StateFault 2496

Ungültig DBParameter GroupStateFault ... 2496
Ungültig DBSecurity GroupStateFault ... 2496
Ungültig DBSnapshot StateFault ... 2497
Ungültig DBSubnet GroupFault ... 2497
Ungültig DBSubnet GroupStateFault ... 2497
Ungültig DBSubnet StateFault ... 2498
InvalidEventSubscriptionStateFault ... 2498
InvalidGlobalClusterStateFault ... 2498
InvalidOptionGroupStateFault ... 2499
InvalidRestoreFault ... 2499
InvalidSubnet ... 2499
Ungültig VPCNetwork StateFault ... 2500
KMSKeyNotAccessibleFault ... 2500
OptionGroupNotFoundFault ... 2500
PointInTimeRestoreNotEnabledFault ... 2501
ProvisionedIopsNotAvailableInAZFault ... 2501
ResourceNotFoundFault ... 2501
SNSInvalidTopicFault ... 2502
SNSNoAuthorizationFault ... 2502
SNSTopicArnNotFoundFault ... 2502
SharedSnapshotQuotaExceededFault ... 2503
SnapshotQuotaExceededFault ... 2503
SourceNotFoundFault ... 2503
StorageQuotaExceededFault ... 2504
StorageTypeNotSupportedFault ... 2504
SubnetAlreadyInUse ... 2504
SubscriptionAlreadyExistFault ... 2505
SubscriptionCategoryNotFoundFault ... 2505
SubscriptionNotFoundFault ... 2505
Referenz für Daten-API ... 2506
Allgemeines ... 2510
GetEngineStatus ... 2510
ExecuteFastReset ... 2513
_____ ... 2514
QueryLanguageVersion ... 2514
FastResetToken ... 2515

Abfragen 2515
ExecuteGremlinQuery 2516
ExecuteGremlinExplainQuery 2518
ExecuteGremlinProfileQuery 2520
ListGremlinQueries 2522
GetGremlinQueryStatus 2523
CancelGremlinQuery 2525
_____ 2526
ExecuteOpenCypherQuery 2526
ExecuteOpenCypherExplainQuery 2528
ListOpenCypherQueries 2530
GetOpenCypherQueryStatus 2531
CancelOpenCypherQuery 2533
_____ 2534
QueryEvalStats 2534
GremlinQueryStatus 2535
GremlinQueryStatusAttributes 2535
Bulk-Loader 2536
StartLoaderJob 2536
GetLoaderJobStatus 2543
ListLoaderJobs 2546
CancelLoaderJob 2548
_____ 2549
LoaderIdResult 2549
Streams 2549
GetPropertygraphStream 2549
_____ 2553
PropertygraphRecord 2553
PropertygraphData 2554
Statistiken 2555
GetPropertygraphStatistics 2555
ManagePropertygraphStatistics 2556
DeletePropertygraphStatistics 2558
GetPropertygraphSummary 2559
_____ 2560
Statistiken 2560

StatisticsSummary 2561
DeleteStatisticsValueMap 2561
RefreshStatisticsIdMap 2562
NodeStructure 2562
EdgeStructure 2562
SubjectStructure 2563
PropertygraphSummaryValueMap 2563
PropertygraphSummary 2564
ML-Datenverarbeitung 2565
Starten MLData ProcessingJob 2566
Liste MLData ProcessingJobs 2569
Holen MLData ProcessingJob 2570
Stornieren MLData ProcessingJob 2572
_____ 2573
MlResourceDefinition 2573
MlConfigDefinition 2574
Training von ML-Modellen 2574
Starten MLModel TrainingJob 2574
Liste MLModel TrainingJobs 2578
Holen MLModel TrainingJob 2579
Stornieren MLModel TrainingJob 2581
_____ 2582
CustomModelTrainingParameters 2582
ML-Modelltransformation 2583
Starten MLModel TransformJob 2583
Liste MLModel TransformJobs 2586
Holen MLModel TransformJob 2588
Stornieren MLModel TransformJob 2589
_____ 2590
CustomModelTransformParameters 2590
ML-Inferenzendpunkt 2591
Erstellen MLEndpoint 2591
Liste MLEndpoints 2594
Holen MLEndpoint 2595
Löschen MLEndpoint 2596
Ausnahmen 2598

AccessDeniedException 2599
BadRequestException 2599
BulkLoadIdNotFoundException 2600
CancelledByUserException 2600
ClientTimeoutException 2601
ConcurrentModificationException 2601
ConstraintViolationException 2602
ExpiredStreamException 2602
FailureByQueryException 2603
IllegalArgumentException 2603
InternalFailureException 2604
InvalidArgumentException 2604
InvalidNumericDataException 2604
InvalidParameterException 2605
LoadUrlAccessDeniedException 2605
MalformedQueryException 2606
MemoryLimitExceededException 2606
MethodNotAllowedException 2607
MissingParameterException 2607
MLResourceNotFoundException 2608
ParsingException 2608
PreconditionsFailedException 2609
QueryLimitExceededException 2609
QueryLimitException 2609
QueryTooLargeException 2610
ReadOnlyViolationException 2610
S3Exception 2611
ServerShutdownException 2611
StatisticsNotAvailableException 2612
StreamRecordsNotFoundException 2612
ThrottlingException 2613
TimeLimitExceededException 2613
TooManyRequestsException 2613
UnsupportedOperationException 2614
UnloadUrlAccessDeniedException 2614
..... mmdcxvi

Was ist Amazon Neptune?

Amazon Neptune ist ein schneller, zuverlässiger, vollständig verwalteter Graph-Datenbankservice, mit dem es ganz einfach ist, Anwendungen zu erstellen und auszuführen, die mit stark verbundenen Datensätzen arbeiten. Den Kern von Neptune bildet eine speziell entwickelte, hochleistungsfähige Graphdatenbank-Engine. Diese Engine ist für die Speicherung von Milliarden von Beziehungen und die Abfrage des Graphen mit einer Latenzzeit von Millisekunden optimiert. Neptune unterstützt die beliebten Property-Graph-Abfragesprachen Apache TinkerPop Gremlin und OpenCypher von Neo4j sowie SPARQL, die RDF-Abfragesprache des W3C. Somit können Sie Abfragen erstellen, die effizient durch stark verbundene Datensätze navigieren. Neptune unterstützt Anwendungsfälle für Graphen wie Empfehlungs-Engines, Betrugserkennung, Wissensgraphen, Wirkstoffforschung und Netzwerksicherheit.

Die Neptune-Datenbank ist hochverfügbar und bietet Read Replicas, point-in-time Wiederherstellung, kontinuierliches Backup auf Amazon S3 und Replikation über Availability Zones hinweg. Neptune stellt Datensicherheitsfunktionen bereit, einschließlich Unterstützung für eine Verschlüsselung im Ruhezustand und während der Übertragung. Neptune ist ein vollständig verwalteter Service, Sie müssen sich also keine Gedanken mehr über Datenbankverwaltungsaufgaben wie die Bereitstellung von Hardware, die Anwendung von Software-Patches, Setup, Konfiguration oder Sicherungen machen.

Neptune Analytics ist eine Analysedatenbank-Engine, die die Neptune-Datenbank ergänzt und schnell große Mengen an Grafikdaten im Speicher analysieren kann, um Erkenntnisse zu gewinnen und Trends zu finden. Neptune Analytics ist eine Lösung für die schnelle Analyse vorhandener Graphdatenbanken oder Graphdatensätze, die in einem Data Lake gespeichert sind. Dabei werden gängige Algorithmen für die Graphanalyse und analytische Abfragen mit geringer Latenz verwendet.

Wenn Sie mehr über Amazon Neptune erfahren möchten, empfehlen wir Ihnen, mit den folgenden Abschnitten zu beginnen:

- Erste Schritte mit Amazon Neptune
- Übersicht über Amazon-Neptune-Features

Wenn Sie noch keine Erfahrung mit Graphen haben oder noch nicht bereit sind, in eine vollständige Neptune-Produktionsumgebung zu investieren, erfahren Sie unter Erste Schritte mit Neptune, wie Sie Neptune-Jupyter-Notebooks zum Lernen und Entwickeln verwenden können, ohne dass Ihnen Kosten entstehen.

Bevor Sie mit dem Entwerfen einer Datenbank beginnen, empfehlen wir Ihnen außerdem, das GitHub Repository AWS Reference Architectures for Using Graph Databases zu konsultieren. Dort können Sie sich über Ihre Auswahlmöglichkeiten zu Graphdatenmodellen und Abfragesprachen informieren und Beispiele für Referenzarchitekturen für Implementierungen durchstöbern.

Wichtige Servicekomponenten

- Primäre DB-Instance – Unterstützt Lese- und Schreiboperationen und führt alle Datenänderungen im Cluster-Volume durch. Jeder Neptune-DB-Cluster verfügt über eine primäre DB-Instance, die für das Schreiben (d. h. Laden oder Ändern) von Graphdatenbankinhalten verantwortlich ist.
- Neptune-Replikat – Stellt eine Verbindung zu demselben Speicher-Volume wie die primäre DB-Instance her und unterstützt nur Lesevorgänge. Jeder Neptune-DB-Cluster kann zusätzlich zur primären DB-Instance bis zu 15 Neptune-Replicas besitzen. Dies bewirkt eine hohe Verfügbarkeit durch das Lokalisieren von Neptune-Replicas in separaten Availability Zones und Verteilung der Last vom Lesen von Clients.
- Cluster-Volume – Neptune-Daten werden im Cluster-Volume gespeichert, das auf Zuverlässigkeit und hohe Verfügbarkeit ausgelegt ist. Ein Cluster-Volume besteht aus Kopien der Daten in mehreren Availability Zones in einer einzigen AWS Region. Da Ihre Daten automatisch zwischen Availability Zones repliziert werden, sind Sie sehr lange beständig und es besteht eine geringe Wahrscheinlichkeit für Datenverlust.

Unterstützt Open Graph APIs

Amazon Neptune unterstützt Open Graph sowohl APIs für Eigenschaftsdiagramme (Gremlin und OpenCypher) als auch für RDF-Graphen (SPARQL). Es bietet eine hohe Leistung für beide diese Graphmodelle und deren Abfragesprachen. Sie können das Property Graph (PG) -Modell wählen und auf dasselbe Diagramm zugreifen, sowohl mit der OpenCypher-Abfragesprache als auch mit der Gremlin-Abfragesprache. and/or Wenn Sie das standardmäßige W3C-RDF-Modell (Resource Description Framework) verwenden, können Sie auf Ihren Graphen über die standardmäßige SPARQL-Abfragesprache zugreifen.

Sehr sicher

Neptune stellt mehrere Sicherheitsebenen für Ihre Datenbank bereit. Sicherheitsfunktionen umfassen die Netzwerkisolierung mit Amazon VPC und die Verschlüsselung im Ruhezustand mithilfe von Schlüsseln, die Sie über AWS Key Management Service (AWS KMS) erstellen und steuern. Auf einer verschlüsselten Neptune-Instance werden Daten im zugrunde liegenden Speicher verschlüsselt. Das gilt auch für automatische Sicherungen, Snapshots und Replikate in demselben Cluster.

Vollständig verwaltet

Mit Amazon Neptune müssen Sie sich keine Gedanken mehr über Datenbankverwaltungsaufgaben wie die Bereitstellung von Hardware, die Anwendung von Software-Patches, Setup, Konfiguration oder Sicherungen machen.

Sie können mit Neptune anspruchsvolle, interaktive Graphanwendungen erstellen, die Milliarden von Beziehungen in Millisekunden abfragen können. SQL-Abfragen für starke verbundene Daten sind komplex und schwer in der Leistung zu verbessern. Mit Neptune können Sie die beliebten Graph-Abfragesprachen Gremlin, openCypher und SPARQL verwenden, um leistungsstarke Abfragen auszuführen, die leicht zu verfassen sind und bei verbundenen Daten eine gute Leistung zeigen. Durch diese Funktion wird die Komplexität der Codes erheblich verringert, sodass Sie schnell Anwendungen erstellen können, die Beziehungen verarbeiten.

Neptune wurde entwickelt, um eine Verfügbarkeit von mehr als 99,99 % zu bieten. Es verbessert die Datenbankleistung und -verfügbarkeit durch die enge Integration der Datenbank-Engine in eine SSD-gestützte virtualisierte Speicherebene, die für Datenbank-Workloads entwickelt wurde. Neptune-Speicher ist fehlertolerant und repariert sich selbst. Festplattenausfälle werden im Hintergrund ohne Verlust der Datenbankverfügbarkeit repariert. Neptune erkennt Datenbankabstürze automatisch und startet neu, ohne dass eine Wiederherstellung nach einem Absturz oder die Neuerstellung des Datenbank-Caches erforderlich ist. Fällt die gesamte Instance aus, führt Neptune automatisch einen Failover zu einem der bis zu 15 Lesereplikate durch.

Änderungen und Updates für Amazon Neptune

In der folgenden Tabelle werden wichtige Änderungen in Amazon Neptune beschrieben.

Änderung	Beschreibung	Datum
Neptune in der Region Asien-Pazifik (Hyderabad)	Amazon Neptune ist jetzt in der Region Asien-Pazifik (Hyderabad) (`ap-south-2`) verfügbar.	10. März 2026
Motorversion 1.4.7.0	Ab 2026-03-03 wird die Engine-Version 1.4.7.0 allgemein eingesetzt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informationen zu dieser Engine-Version finden Sie unter Neptune Engine Release 1.4.7.0.	3. März 2026
Motorversion 1.4.6.3	Ab 2025-12-18 wird die Engine-Version 1.4.6.3 allgemein eingesetzt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informationen zu dieser Engine-Version finden Sie unter Neptune Engine Release 1.4.6.3.	18. Dezember 2025
Motorversion 1.4.6.2	Ab dem 18.11.2025 wird die Engine-Version 1.4.6.2 allgemein eingesetzt. Bitte beachten Sie, dass es	18. November 2025

	mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informationen zu dieser Engine-Version finden Sie unter Neptune Engine Release 1.4.6.2.	
Motorversion 1.4.6.1	Ab dem 18.09.2025 wird die Engine-Version 1.4.6.1 allgemein eingesetzt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informationen zu dieser Engine-Version finden Sie unter Neptune Engine Release 1.4.6.1.	18. September 2025
Motorversion 1.4.6.0	Ab 2025-09-02 wird die Engine-Version 1.4.6.0 allgemein eingesetzt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informationen zu dieser Engine-Version finden Sie unter Neptune Engine Release 1.4.6.0.	2. September 2025
Öffentliche Endpunkte	Amazon Neptune unterstützt jetzt öffentliche Endpunkte. Weitere Informationen finden Sie unter Öffentliche Endgeräte.	27. August 2025

Engine-Version 1.4.5.1	Ab dem 30.06.2025 wird die Engine-Version 1.4.5.1 allgemein eingesetzt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informationen zu dieser Engine-Version finden Sie unter Neptune Engine Release 1.4.5.1.	30. Juni 2025
Neptune wurde in den Regionen Asien-Pazifik (Melbourne) und Kanada West (Calgary) eingeführt	Amazon Neptune ist jetzt in den Regionen Asien-Pazifik (Melbourne) (`ap-southeast-4`) und Kanada West (Calgary) (`ca-west-1`) verfügbar.	28. Mai 2025
Motorversion 1.4.5.0	Ab dem 09.04.2025 wird die Engine-Version 1.4.5.0 allgemein eingesetzt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informationen zu dieser Engine-Version finden Sie unter Neptune Engine Release 1.4.5.0.	9. April 2025

Neptune wurde in der Region Asien-Pazifik (Malaysia) eingeführt	Amazon Neptune ist jetzt in der Region Asien-Pazifik (Malaysia) (`ap-southeast-5`) verfügbar. Verwaltete Notebooks werden derzeit nicht für die Region Asien-Pazifik (Malaysia) unterstützt, können aber manuell eingerichtet werden. Weitere Informationen finden Sie unter Manuelles Einrichten von Neptune-Notizbüchern.	28. Februar 2025
Engine-Version 1.4.4.0	Ab dem 24.02.2025 wird die Engine-Version 1.4.4.0 allgemein eingesetzt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informationen zu dieser Engine-Version finden Sie unter Neptune Engine Release 1.4.4.0.	24. Februar 2025
Engine-Version 1.4.3.0	Ab 2025-01-21 wird die Engine-Version 1.4.3.0 allgemein eingesetzt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informationen zu dieser Engine-Version finden Sie unter Neptune Engine Release 1.4.3.0.	21. Januar 2025

Engine-Version 1.4.2.0	Ab 2024-12-19 wird die Engine-Version 1.4.2.0 allgemein eingesetzt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informationen zu dieser Engine-Version finden Sie unter Neptune Engine Release 1.4.2.0.	19. Dezember 2024
Engine-Version 1.4.1.0	Ab 2024-11-21 wird die Engine-Version 1.4.1.0 allgemein eingesetzt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informationen zu dieser Engine-Version finden Sie unter Neptune Engine Release 1.4.1.0.	21. November 2024
Engine-Version 1.4.0.0	Ab 2024-11-06 wird die Engine-Version 1.4.0.0 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informationen zu dieser Engine-Version finden Sie unter Neptune Engine Release 1.4.0.0.	6. November 2024

Engine-Version 1.3.4.0	Ab 2024-10-01 wird die Engine-Version 1.3.4.0 allgemein eingesetzt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informationen zu dieser Engine-Version finden Sie unter Neptune Engine Release 1.3.4.0.	1. Oktober 2024
Motorversion 1.3.3.0	Ab 2024-08-05 wird die Engine-Version 1.3.3.0 allgemein eingesetzt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informationen zu dieser Engine-Version finden Sie unter Neptune Engine Release 1.3.3.0.	5. August 2024
Engine-Version 1.2.1.2	Ab 2024-08-05 wird die Engine-Version 1.2.1.2 allgemein eingesetzt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informationen zu dieser Engine-Version finden Sie unter Neptune Engine Release 1.2.1.2.	5. August 2024

Engine-Version 1.3.2.1	Ab 2024-06-20 wird die Engine-Version 1.3.2.1 allgemein eingesetzt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informationen zu dieser Engine-Version finden Sie unter Neptune Engine Release 1.3.2.1.	20. Juni 2024
Engine-Version 1.3.2.0	Ab 2024-06-10 wird die Engine-Version 1.3.2.0 allgemein eingesetzt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informationen zu dieser Engine-Version finden Sie unter Neptune Engine Release 1.3.2.0.	10. Juni 2024
Engine-Version 1.2.1.1	Ab 2024-03-11 wird die Engine-Version 1.2.1.1 allgemein eingesetzt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informationen zu dieser Engine-Version finden Sie unter Neptune Engine Release 1.2.1.1.	11. März 2024

Engine-Version 1.3.1.0	Ab 2024-03-06 wird die Engine-Version 1.3.1.0 allgemein eingesetzt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informationen zu dieser Engine-Version finden Sie unter Neptune Engine Release 1.3.1.0.	6. März 2024
Aktualisierung der verwalteten Richtlinienberechtigungen AWS	Die `NeptuneReadOnlyAccess` und die `NeptuneFullAccess` verwalteten Richtlinien enthalten jetzt `Sid` (Statement ID) als ID in der Richtlinienerklärung.	22. Januar 2024
Neptune bietet jetzt E/A-optimierten Speicher	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/ODamit verwenden Sie.	13. Dezember 2023

Änderungen der verwalteten IAM-Richtlinie für Neptune	Die von NeptuneConsoleFullAccessIAM verwaltete Richtlinie wurde aktualisiert, um die für die Interaktion mit Neptune Analytics-Diagrammen erforderlichen Berechtigungen zu gewähren, eine neue NeptuneGraphReadOnlyAccessRichtlinie wurde hinzugefügt, um schreibgeschützten Zugriff auf Neptune Analytics-Graphressourcen zu gewähren, und eine neue `AWSServiceRoleForNeptuneGraphPolicy` Richtlinie wurde hinzugefügt, damit Neptune Analytics-Diagramme Betriebs- und Nutzungsmetriken und -protokolle veröffentlichen können. CloudWatch	29. November 2023
Engine-Version 1.3.0.0	Seit dem 15.11.2023 wird die Engine-Version 1.3.0.0 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informationen zu dieser Engine-Version finden Sie unter Neptune-Engine-Version 1.3.0.0.	15. November 2023
Neptune wurde in der Region Israel (Tel Aviv) gestartet	Amazon Neptune ist jetzt in der Region Israel (Tel Aviv) (`il-central-1`) verfügbar.	13. November 2023

Blogbeitrag über die Implementierung von time-to-live Eigenschaftsdiagrammen in Neptune	Weitere Informationen finden Sie unter Implementieren von Time to Live in Amazon Neptune, Teil 1: Property Graph von Melissa Kwok, Mike Havey und Kevin Phillips.	27. Oktober 2023
Engine-Version 1.2.1.0.R7	Seit dem 06.10.2023 wird die Engine-Version 1.2.1.0.R 7 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informati onen zu dieser Engine-Version finden Sie unter Neptune-E ngine-Version 1.2.1.0.R7.	06. Oktober 2023
Engine-Version 1.2.0.0.R4	Ab dem 29.09.2023 wird die Engine-Version 1.2.0.0.R 4 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informati onen zu dieser Engine-Version finden Sie unter Neptune-E ngine-Version 1.2.0.0.R4.	29. September 2023

Engine-Version 1.2.0.1.R3	Ab dem 27.09.2023 wird die Engine-Version 1.2.0.1.R3 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informationen zu dieser Engine-Version finden Sie unter Neptune-Engine-Version 1.2.0.1.R3.	27. September 2023
Engine-Version 1.2.1.0.R6	Ab dem 12.09.2023 wird die Engine-Version 1.2.1.0.R6 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informationen zu dieser Engine-Version finden Sie unter Neptune-Engine-Version 1.2.1.0.R6.	12. September 2023
Engine-Version 1.2.0.2.R6	Ab dem 12.09.2023 wird die Engine-Version 1.2.0.2.R6 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informationen zu dieser Engine-Version finden Sie unter Neptune-Engine-Version 1.2.0.2.R6.	12. September 2023

Blogbeitrag über die Verwendung einer blue/green Bereitstellungsstrategie für Neptune-Triebwerks-Upgrades	Weitere Informationen finden Sie unter Verbessern der Verfügbarkeit von Amazon Neptune während eines Engine-Upgrades mithilfe der blue/green Bereitstellung durch Ankit Gupta und Abhishek Mishra.	11. September 2023
Engine-Version 1.2.1.0.R5	Ab dem 02.09.2023 wird die Engine-Version 1.2.1.0.R5 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informationen zu dieser Engine-Version finden Sie unter Neptune-Engine-Version 1.2.1.0.R5.	2. September 2023
Engine-Version 1.2.0.2.R5	Ab dem 16.08.2023 wird die Engine-Version 1.2.0.2.R5 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informationen zu dieser Engine-Version finden Sie unter Neptune-Engine-Version 1.2.0.2.R5.	16. August 2023

Engine-Version 1.2.1.0.R4	Ab dem 10.08.2023 wird die Engine-Version 1.2.1.0.R 4 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informati onen zu dieser Engine-Version finden Sie unter Neptune-E ngine-Version 1.2.1.0.R4.	10. August 2023
Blog-Beitrag zur Neptune-E ngine-Version 1.2.1.0	Weitere Informationen finden Sie unter Erkunden der funktionsreichen Version 1.2.1.0 für Amazon Neptune von Joy Wang, Kevin Phillips, Andrea Nassisi und Navtanay Sinha.	4. August 2023
Blog-Beitrag über die Entwicklung einer multimoda len Datenbanklösung mit Neptune	Weitere Informationen finden Sie unter Entwerfen einer anwendungsfallorientierten, hoch skalierbaren Multimodell-Datenbanklösung mithilfe von Amazon Neptune von Mike Havey.	18. Juli 2023
Blog-Beitrag über Anwendung sfälle und bewährte Methoden für Neptune Serverless	Weitere Informationen finden Sie unter Anwendungsfälle und bewährte Methoden zur Optimierung von Kosten und Leistung mit Amazon Neptune Serverless von Kevin Phillips und Ankit Gupta.	28. Juni 2023

Engine-Version 1.2.1.0.R3	Ab dem 13.06.2023 wird die Engine-Version 1.2.1.0.R3 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informationen zu dieser Engine-Version finden Sie unter Neptune-Engine-Version 1.2.1.0.R3.	13. Juni 2023
Blog-Beitrag zur Generierung von Freizeitvorschlägen in Echtzeit	Weitere Informationen finden Sie unter Generieren von Vorschlägen für Freizeitaktivitäten in Echtzeit mit Amazon Neptune von Michael Meidlinger und Nils Müller.	6. Juni 2023
Blogbeitrag über molekulare Modellierung mit Neptune und RDKit	Siehe Modelldaten des molekularen SMILES-Modells mit Amazon Neptune und RDKit von Graham Kutchek.	01. Juni 2023
Blog-Beitrag über die Verwendung von Caching zur Beschleunigung der Neptune-Leistung (Teil 3)	Weitere Informationen finden Sie unter Beschleunigen der Leistung von Graphabfragen mit Caching in Amazon Neptune, Teil 3: Cluster-weite Caching-Architekturen von Neptune mit Amazon ElastiCache von Taylor Riggan, Abhishek Mishra, Melissa Kwok und Kelvin Lawrence.	26. Mai 2023

Blog-Beitrag über die Verwendung von Caching zur Beschleunigung der Neptune-Leistung (Teil 2)	Weitere Informationen finden Sie unter Beschleunigen der Leistung von Graphabfragen mit Caching in Amazon Neptune, Teil 2: Weitere Caching-Features von Neptune von Taylor Riggan, Abhishek Mishra, Melissa Kwok und Kelvin Lawrence.	26. Mai 2023
Blog-Beitrag über die Verwendung von Caching zur Beschleunigung der Neptune-Leistung (Teil 1)	Weitere Informationen finden Sie unter Beschleunigen der Leistung von Graphabfragen mit Caching in Amazon Neptune, Teil 1: Abfragen und Puffer-Pool-Caching von Taylor Riggan, Abhishek Mishra, Melissa Kwok und Kelvin Lawrence.	26. Mai 2023
Blog-Beitrag zur Supply-Chain-Analyse mit Neptune	Weitere Informationen finden Sie unter Analyse und Visualisierung von Lieferkettendaten mit Amazon Neptune und der Neptune Workbench von Dhiraj Thakur und Rajdip Chaudhur.	10. Mai 2023

Engine-Version 1.2.0.2.R4	Ab dem 08.05.2023 wird die Engine-Version 1.2.0.2.R 4 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informati onen zu dieser Engine-Version finden Sie unter Neptune-E ngine-Version 1.2.0.2.R4.	8. Mai 2023
Neptune-Start in der Region Naher Osten (VAE)	Amazon Neptune ist jetzt in der Region Naher Osten (VAE) `me-central-1` verfügbar.	2. Mai 2023
Engine-Version 1.2.1.0.R2	Ab dem 02.05.2023 wird die Engine-Version 1.2.1.0.R 2 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informati onen zu dieser Engine-Version finden Sie unter Neptune-E ngine-Version 1.2.1.0.R2.	2. Mai 2023
Blog-Beitrag zum Erstellen eines Wissensgraphen auf Neptune mit KI-gestützter Videoanalyse mithilfe von Media2Cloud	Weitere Informationen finden Sie unter Erstellen eines Wissensgraphen auf Amazon Neptune mit KI-gestützter Videoanalyse mithilfe von Media2Cloud von Mike Havey.	2. Mai 2023

Blogbeitrag darüber, wie mit DevOcean Neptune eine Plattform zur Behebung von Sicherheitslücken entwickelt wurde	Erfahren Sie von Gil Makmel und Charles Ivie, wie mithilfe von Amazon Neptune eine Managementplattform zur Behebung von Sicherheitslücken für Cloud-native Anwendungen DevOcean entwickelt wurde.	25. April 2023
Blog-Beitrag darüber, wie Getir mit Neptune ein Betrugserkennungssystem entwickelte	Weitere Informationen finden Sie in Wie Getir mithilfe von Amazon Neptune und Amazon DynamoDB ein umfassendes Betrugserkennungssystem entwickelte von Berkay Berkman, Mahmut Turan, Mutlu Polatcan, Umut Cemal Kıraç, Yağız Yanıkoğlu und Esra Kayabali.	06. April 2023
Blog-Beitrag darüber, wie Wiz mit Neptune die Cloud-Sicherheit neu erfindet	Siehe Die Welt ist ein Graph: Wie Wiz mit einem Graphen in Amazon Neptune die Cloud-Sicherheit neu erfindet Ami Luttwak und Brad Bebee.	31. März 2023
Engine-Version 1.2.0.2.R3	Ab dem 27.03.2023 wird die Engine-Version 1.2.0.2.R3 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informationen zu dieser Engine-Version finden Sie unter Neptune-Engine-Version 1.2.0.2.R3.	27. März 2023

Blog-Beitrag darüber, wie CSC Generation die Produkterkennung mit Neptune unterstützt	Weitere Informationen finden Sie unter Wie CSC Generation die Produkterkennung mit Wissensgraphen unter Verwendung von Amazon Neptune unterstützt von Bobber Cheng, Ronit Rudra und Melissa Kwok.	21. März 2023
Engine-Version 1.2.1.0	Ab dem 08.03.2023 wird die Engine-Version 1.2.1.0 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informationen zu dieser Engine-Version finden Sie unter Neptune-Engine-Version 1.2.1.0.	08. März 2023
Blogbeitrag über die Erkundung neuer Funktionen von TinkerPop 3.6.x in Neptune	Weitere Informationen finden Sie unter Erkunden der neuen Funktionen von Apache TinkerPop 3.6.x in Amazon Neptune von Stephen Mallette.	08. März 2023
Blog-Beitrag über die Verwendung semantischen Denkens, um neue Fakten aus Ihrem RDF-Graph abzuleiten	Weitere Informationen finden Sie unter Verwenden Sie semantisches Denken, um mithilfe der Integration RDFox mit Amazon Neptune neue Fakten aus Ihrem RDF-Diagramm abzuleiten von Charles Ivie und Diana Marks.	20. Februar 2023

Blog-Beitrag zur Analyse von FHIR-Daten im Gesundheitswesen mit Neptune	Weitere Informationen finden Sie unter Analyse von FHIR-Daten im Gesundheitswesen mit Amazon Neptune von Alena Schmickl.	13. Februar 2023
Blog-Beitrag über die Entwicklung einer Lösung zur Betrugserkennung in Echtzeit mit Neptune ML	Weitere Informationen finden Sie unter Entwicklung einer Lösung zur Betrugserkennung in Echtzeit mit Amazon Neptune ML von Hua Shu und Soji Adeshina.	8. Februar 2023
Engine-Version 1.1.1.0.R7	Ab dem 23.10.2023 wird die Engine-Version 1.1.1.0.R7 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informationen zu dieser Engine-Version finden Sie unter Neptune-Engine-Version 1.1.1.0.R7.	23. Januar 2023
Graph-Explorer veröffentlicht	Graph-Explorer ist ein Open-Source-Frontend-Webanwendungstool zur Visualisierung von Grafikdaten. https://github.com/aws/Siehe Graph-Explorer.	3. Januar 2023

Wartungsversion 1.1.0.0.R3	Ab dem 23.12.2022 wird die Wartungsversion der Engine-Version 1.1.0.0.R3 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informationen zu dieser Engine-Version finden Sie unter Neptune-Engine-Version 1.1.0.0.R3.	23. Dezember 2022
Neptune Workbench läuft jetzt auf Amazon Linux 2 und 3. JupyterLab	Neptune Graph-Notebooks laufen jetzt in einer Amazon Linux 2-Umgebung mit 3. JupyterLab Informationen zur Umstellung auf diese neue Umgebung finden Sie unter Migration Ihrer Neptune-Notizbücher von Jupyter auf JupyterLab 3.	21. Dezember 2022
Blogbeitrag über Leistungsempfehlungen und die Suche mithilfe eines IMDb Wissensgraphen (Teil 3)	Weitere Informationen finden Sie unter Leistungsempfehlungen und suchen mithilfe eines IMDb Wissensgraphen — Teil 3 von Divya Bhargavi, Soji Adeshina, Gaurav Rele, Karan Sindwani, Vidya Sagar Ravipati und Matthew Rhodes.	20. Dezember 2022

Blogbeitrag über Power-Empfehlungen und die Suche mithilfe eines Wissensgraphen IMDb (Teil 2)	Weitere Informationen finden Sie unter Leistungsempfehlungen und suchen mithilfe eines IMDb Wissensgraphen — Teil 2 von Matthew Rhodes, Soji Adeshina, Divya Bhargavi, Gaurav Rele, Karan Sindwani und Vidya Sagar Ravipati.	20. Dezember 2022
Blogbeitrag zu Power-Empfehlungen und zur Suche mithilfe eines Wissensgraphen IMDb (Teil 1)	Weitere Informationen finden Sie unter Leistungsempfehlungen und suchen mithilfe eines IMDb Wissensgraphen — Teil 1 von Gaurav Rele, Soji Adeshina, Divya Bhargavi, Karan Sindwani, Vidya Sagar Ravipati und Matthew Rhodes.	20. Dezember 2022
Neptune Serverless ist jetzt in neuen Regionen verfügbar AWS	Am 16.12.2022 wurde Neptune Serverless in den folgenden neuen AWS - Regionen eingeführt: Kanada (Zentral), Europa (Stockholm), Europa (Frankfurt), Asien-Pazifik (Singapur) und Asien-Pazifik (Sydney). Alle Regionen, in denen Neptune Serverless verfügbar ist, finden Sie unter Amazon-Neptune-Serverless-Einschränkungen.	16. Dezember 2022

Engine-Version 1.2.0.2.R2	Ab dem 15.12.2022 wird die Engine-Version 1.2.0.2.R 2 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informati onen zu dieser Engine-Version finden Sie unter Neptune-E ngine-Version 1.2.0.2.R2.	15. Dezember 2022
Engine-Version 1.2.0.0.R3	Ab dem 15.12.2022 wird die Engine-Version 1.2.0.0.R 3 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informati onen zu dieser Engine-Version finden Sie unter Neptune-E ngine-Version 1.2.0.0.R3.	15. Dezember 2022
Engine-Version 1.2.0.1.R2	Ab dem 13.12.2022 wird die Engine-Version 1.2.0.1.R 2 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informati onen zu dieser Engine-Version finden Sie unter Neptune-E ngine-Version 1.2.0.1.R2.	13. Dezember 2022

Blogbeitrag über den Entwurf einer Big-Data-Analysearchitektur für Bildungszwecke mit AWS	Weitere Informationen finden Sie unter Entwurf einer Big-Data-Analysearchitektur für das Bildungswesen mit AWS von Lavanya Sood.	13. Dezember 2022
Blog-Beitrag darüber, wie Graphdatenbanken das Lernen verbessern können	Weitere Informationen finden Sie unter Wie Graphdatenbanken das Lernen verbessern können von Lavanya Sood.	8 Dezember 2022
Blogbeitrag über das Laden von RDF-Daten in Neptune mit Glue AWS	Weitere Informationen finden Sie unter Laden von RDF-Daten mit AWS Glue in Amazon Neptune von Mike Havey und Fabrizio Napolitano.	23. November 2022
Engine-Version 1.2.0.2	Ab dem 20.11.2022 wird die Engine-Version 1.2.0.2 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informationen zu dieser Engine-Version finden Sie unter Neptune-Engine-Version 1.2.0.2.	20. November 2022

Engine-Version 1.2.0.1	Ab dem 26.10.2022 wird die Engine-Version 1.2.0.1 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informationen zu dieser Engine-Version finden Sie unter Neptune-Engine-Version 1.2.0.1.	26. Oktober 2022
Blog-Beitrag zur Betrugserkennung mit Neptune	Weitere Informationen finden Sie unter Unterstützung der Betrugserkennung bei Delivery Hero mit Amazon Neptune von Wilson Tang, Amr Elnaggar, Matias Pons, Mohammad Azzam, Saurabh Deshpande und Luis Rodrigues Soares.	26. Oktober 2022
Blog-Beitrag über Neptune Serverless	Weitere Informationen finden Sie unter Einführung in Amazon Neptune Serverless – Eine vollständig verwaltete Graphdatenbank, die die Kapazität an Ihre Workloads anpasst von Danilo Poccia.	26. Oktober 2022
Blog-Beitrag über ereignisgesteuerte RDF-Importe zu Neptune mit Lambda und SPARQL UPDATE LOAD	Erfahren Sie von John Walker, Onno Buijs, Charles Ivie und Javy de Koning, wie NXP ereignisgesteuerte RDF-Importe nach Amazon Neptune mithilfe von AWS Lambda und SPARQL UPDATE LOAD durchführt.	20. Oktober 2022

Engine-Version 1.2.0.0.R2	Ab dem 14.10.2022 wird die Engine-Version 1.2.0.0.R 2 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informati onen zu dieser Engine-Version finden Sie unter Neptune-E ngine-Version 1.2.0.0.R2.	14. Oktober 2022
Blog-Beitrag zur Kodierung mehrsprachiger Texteigen schaften in Neptune	Weitere Informationen finden Sie unter Trainieren von Vorhersagemodellen durch Kodierung mehrsprachiger Texteigenschaften in Amazon Neptune von Jiani Zhang.	14. Oktober 2022
Blog-Beitrag zum automatis ierten Testen des Neptune-D atenzugriffs	Siehe Automatisiertes Testen des Amazon Neptune Neptune-Datenzugriffs mit Apache TinkerPop Gremlin von Greg Biegel.	28. September 2022
Engine-Version 1.1.1.0.R6	Ab dem 23.09.2022 wird die Engine-Version 1.1.1.0.R 6 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informati onen zu dieser Engine-Version finden Sie unter Neptune-E ngine-Version 1.1.1.0.R6.	23. September 2022

Blog-Beitrag darüber, wie Informatica® Neptune verwendet	Weitere Informationen finden Sie unter Wie Informatica® Cloud Data Governance and Catalog Amazon Neptune für Wissensgraphen verwendet von Tiju Titus John, Deepak Ram und Farooq Ashraf.	20. September 2022
Blog-Beitrag über die Verwendung von Neptune und Tom Sawyer Perspectives zur Aufdeckung von Finanzbetrug	Siehe Aufdeckung von Finanzbetrug mit Amazon Neptune und Tom Sawyer Perspectives von Janet M. Six, Senior Product Manager bei Tom Sawyer Software.	30. August 2022
Blogbeitrag über den Aufbau einer GNN-basierten Lösung zur Betrugserkennung in Echtzeit mithilfe von Neptune SageMaker und DGL	Weitere Informationen finden Sie unter Entwicklung einer GNN-basierten Lösung zur Betrugserkennung in Echtzeit mithilfe von Amazon SageMaker, Amazon Neptune und der Deep Graph Library von Jian Zhang, Haozhu Wang und Mengxin Zhu.	11. August 2022
Blog-Beitrag zur Verwendung von Ressourcen-Tags zum Stoppen und Starten von Neptune-Umgebungsressourcen	Weitere Informationen finden Sie unter Automatisieren des Stoppens und Startens von Amazon-Neptune-Umgebungsressourcen mithilfe von Ressourcen-Tags von Kevin Phillips.	1. August 2022

Blogbeitrag über einen Künstler, der zu Apache beiträgt TinkerPop	Siehe Beyond Code: Der Künstler, der zu Apache beiträgt TinkerPop von Stephen Mallette und Ketrina Thompson.	1. August 2022
Blog-Beitrag über die differenzierte Zugriffskontrolle für Aktionen auf der Neptune-Datenebene	Weitere Informationen finden Sie unter Differenzierte Zugriffskontrolle für Aktionen auf der Amazon-Neptune-Datenebene von Abhishek Mishra und Ankit Gupta.	29. Juli 2022
Blog-Beitrag über globale Neptune-Datenbanken	Weitere Informationen finden Sie unter Einführung in die globale Amazon-Neptune-Datenbank von Navtanay Sinha.	27. Juli 2022
Engine-Version 1.2.0.0	Ab dem 21.07.2022 wird die Engine-Version 1.2.0.0 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informationen zu dieser Engine-Version finden Sie unter Neptune-Engine-Version 1.2.0.0.	21. Juli 2022

Engine-Version 1.1.1.0.R5	Ab dem 21.07.2022 wird die Engine-Version 1.1.1.0.R 5 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informati onen zu dieser Engine-Version finden Sie unter Neptune-E ngine-Version 1.1.1.0.R5.	21. Juli 2022
Engine-Version 1.1.1.0.R4	Ab dem 23.06.2022 wird die Engine-Version 1.1.1.0.R 4 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informati onen zu dieser Engine-Version finden Sie unter Neptune-E ngine-Version 1.1.1.0.R4.	23. Juni 2022
Vereinfachte Workflows für Graph-Analysen und Machine Learning mit Python-Integration	Mit einer Open-Source-Python -Integration, die Datenwiss enschafts- und ML-Workfl ows vereinfacht, können Sie jetzt Graph-Analyse- und Machine-Learning-Aufgaben für Graphdaten ausführen, die in Amazon Neptune gespeiche rt sind. Weitere Informationen finden Sie in der AWS -Data-Wrangler-Dokumentation für Neptune.	7. Juni 2022

Engine-Version 1.1.1.0.R3	Ab dem 07.06.2022 wird die Engine-Version 1.1.1.0.R 3 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informati onen zu dieser Engine-Version finden Sie unter Neptune-E ngine-Version 1.1.1.0.R3.	7. Juni 2022
Blog-Beitrag zur Erkennung von Fake News	Weitere Informationen finden Sie unter Erkennen von Fake News in sozialen Medien mithilfe von grafischem Machine Learning mit Amazon Neptune ML von Hasan Shojaei und Sarita Joshi.	19. Mai 2022
Blog-Beitrag zur Verwendun g von SQL Server Integration Services (SSIS) mit Neptune	Weitere Informationen finden Sie unter Nutzen neuer Erkenntnisse aus Ihren Daten mithilfe von SQL Server Integration Services (SSIS) und Amazon Neptune von Mesgana Gormley und Melissa Kwok.	18. Mai 2022

Wartungsversion 1.1.1.0.R2	Ab dem 16.05.2022 wird die Wartungsversion der Engine-Version 1.1.1.0.R2 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informationen zu dieser Engine-Version finden Sie unter Neptune-Engine-Version 1.1.1.0.R2.	16. Mai 2022
Wartungsversion 1.1.0.0.R2	Ab dem 16.05.2022 wird die Wartungsversion der Engine-Version 1.1.0.0.R2 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informationen zu dieser Engine-Version finden Sie unter Neptune-Engine-Version 1.1.0.0.R2.	16. Mai 2022
Wartungsversion 1.0.5.1.R4	Ab dem 16.05.2022 wird die Wartungsversion der Engine-Version 1.0.5.1.R4 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informationen zu dieser Engine-Version finden Sie unter Neptune-Engine-Version 1.0.5.1.R4.	16. Mai 2022

Wartungsversion 1.0.5.0.R5	Ab dem 16.05.2022 wird die Wartungsversion der Engine-Version 1.0.5.0.R5 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informationen zu dieser Engine-Version finden Sie unter Neptune-Engine-Version 1.0.5.0.R5.	16. Mai 2022
Blog-Beitrag über die allgemeine Verfügbarkeit von openCypher in Neptune	Weitere Informationen finden Sie unter Ankündigung der allgemeinen Verfügbarkeit der openCypher-Unterstützung für Amazon Neptune von Navtanay Sinha und Dave Bechberger.	22. April 2022
Engine-Version 1.1.1.0	Ab dem 19.04.2022 wird die Engine-Version 1.1.1.0 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informationen zu dieser Engine-Version finden Sie unter Neptune-Engine-Version 1.1.1.0.	19. April 2022

Blog-Beitrag über Data Lineage	Weitere Informationen finden Sie unter Erstellen von Data Lineage für Data Lakes mit AWS Glue, Amazon Neptune und Spline von Khoa Nguyen, Krithivasan Balasubramaniyan und Rahul Shaurya.	1. April 2022
Upgrades auf Engine-Version 1.1.0.0 sind wieder verfügbar	Ab dem 21.02.2022 waren Upgrades auf Engine-Version 1.1.0.0 vorübergehend deaktiviert. Diese sind jetzt wieder aktiviert.	22. März 2022
Blog-Beitrag zur Zuverlässigkeit in technischen Versorgungsunternehmen	Weitere Informationen finden Sie unter Verwendung von cloudbasierten datengestützten Energiesystemmodellen zur Unterstützung der Zuverlässigkeit in Versorgungsunternehmen von Abhineet Parchure.	22. März 2022
Neptune-Start in Afrika (Kapstadt)	Amazon Neptune ist jetzt in der Region Afrika (Kapstadt) (`af-south-1`) verfügbar. Die Neptune-Workbench-Notebook-Unterstützung ist auf der Neptune-Konsole in dieser Region jedoch vorübergehend deaktiviert.	24. Februar 2022
Blog-Beitrag zu modellgesteuerten Graphen mit OWL	Weitere Informationen finden Sie unter Modellgesteuerte Grafiken mit OWL in Amazon Neptune von Mike Havey.	23. Februar 2022

Blog-Beitrag zur Erforschung semantischer Wissensgraphen mit Rhizomer	Weitere Informationen finden Sie unter Erkunden Sie die semantischen Wissensgraphen ohne SPARQL mithilfe von Amazon Neptune mit Rhizomer von Roberto García.	22. Februar 2022
Blog-Beitrag zur graphischen Darstellung von Versorgungsnetzen	Siehe AWS Graphing the Utility Grid on von Bobby Wilson und Joseph Beer.	18. Februar 2022
Neue Text-Feature-Kodierungsoptionen in Neptune ML	Neptune unterstützt jetzt die FastText BERT-Textkodierung von Sentence für das Training. Sehen Sie sich die FastTextFunktionen in Neptune ML und die Funktionen von Sentence BERT in Neptune ML an.	15. Februar 2022
Blogbeitrag über Geodatenabfragen OpenSearch mit Neptune	Weitere Informationen finden Sie unter Kombinieren von Amazon Neptune und Amazon OpenSearch Service für Geodatenabfragen von Ross Gabay und Abhilash Vinod.	1. Februar 2022
Blog-Beitrag zum Erkennen von Finanzkriminalität mithilfe von Amazon EKS und Neptune	Weitere Informationen finden Sie unter Erkennen von Finanzkriminalität mithilfe von Amazon EKS und Graphdatenbanken von Severin Gassauer-Fleissner und Zahi Ben Shabat.	1. Februar 2022

Ein Neptune-Cluster-Volume kann jetzt auf 128 Tebibyte (TiB) anwachsen	In allen unterstützten Regionen außer China und GovCloud China wurde die Größenbeschränkung für ein Neptun-Cluster-Volumen nun von 64 TiB auf 128 TiB erhöht. Dies gilt für alle Engine-Versionen ab 1.0.2.2 beginnen. Weitere Informationen finden Sie auf der Seite Amazon-Neptune-Speicher.	1. Februar 2022
Die Neptun-Volltextsuche ist jetzt in alle Versionen von integriert. OpenSearch	Siehe Volltextsuche in Amazon Neptune mit Amazon OpenSearch Service.	28. Januar 2022
Blog-Beitrag zur Verwendung von Docker-Containern zur Bereitstellung von Graph-Notebooks	Siehe Verwenden von Docker-Containern zur Bereitstellung von Graph Notebooks auf AWS, von Ganesh Sawhney und Qiang Zhang.	22. Januar 2022
Engine-Version 1.0.5.1.R3	Ab dem 13.01.2022 wird die Engine-Version 1.0.5.1.R3 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informationen zu dieser Engine-Version finden Sie unter Neptune-Engine-Version 1.0.5.1.R3.	13. Januar 2022

Blog-Beitrag über ein graphbasiertes Empfehlungssystem mit Neptune ML	Weitere Informationen finden Sie unter Gaphbasiertes Empfehlungssystem mit Neptune ML: Eine Illustration zu Herausforderungen bei der Vorhersage von Links in sozialen Netzwerken von Yanwei Cui und Will Badr.	12. Januar 2022
Blog-Beitrag zur automatischen Skalierung von Neptune	Weitere Informationen finden Sie unter Automatische Skalierung Ihrer Amazon-Neptune-Datenbank zur Erfüllung von Workload-Anforderungen von Navtanay Sinha und Sudhanshu Gupta.	29. November 2021
Blog-Beitrag zu interaktiven Graph-Datenanalysen und Visualisierungen	Weitere Informationen finden Sie unter Erstellen interaktiver Grafikdatenanalysen und Visualisierungen mit Amazon Neptune, Amazon Athena Federated Query und Amazon QuickSight von Sandeep Veldi und Abhishek Mishra.	24. November 2021
Blog-Beitrag zur Erkennung von Identitätsbetrug mithilfe von graphbasiertem Deep Learning	Weitere Informationen finden Sie unter Wie Careem mithilfe von graphbasiertem Deep Learning und Amazon Neptune Identitätsbetrug aufdeckt von Kevin O'Brien, Kamran Habib und Will Badr.	23. November 2021

Engine-Version 1.1.0.0	Ab dem 19.11.2021 wird die Engine-Version 1.1.0.0 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informationen zu dieser Engine-Version finden Sie unter Neptune-Engine-Version 1.1.0.0.	19. November 2021
Blog-Beitrag zur Zentralisierung von Datenschutz und Compliance	Weitere Informationen finden Sie unter Zentralisierung von Datenschutz und Compliance in Amazon Neptune with AWS Backup von Brian O'Keefe.	8. November 2021
Blog-Beitrag zur Bekämpfung von Betrug und unangemessenen Zahlungen	Weitere Informationen finden Sie unter Bekämpfung von Betrug und unangemessenen Zahlungen in Echtzeit im Umfang von Bundesausgaben von Vladi Royzman und Spencer Smith.	2. November 2021
Blog-Beitrag zur Verhinderung gefälschter Kontoanmeldungen	Weitere Informationen finden Sie unter Verhinderung gefälschter Kontoanmeldungen in Echtzeit mit KI mithilfe von Amazon Fraud Detector von Anjan Biswas.	29. Oktober 2021

Engine-Version 1.0.5.1.R2	Ab dem 26.10.2021 wird die Engine-Version 1.0.5.1.R 2 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informati onen zu dieser Engine-Version finden Sie unter Neptune-E ngine-Version 1.0.5.1.R2.	26. Oktober 2021
Blogbeitrag zur HawkEye 360°-Vorhersage von Schiffsri siken mithilfe der Deep Graph Library	See HawkEye 360 prognosti ziert das Schiffsrisiko mithilfe der Deep Graph Library und Amazon Neptune von Tim Pavlick, Ian Avilez, Dan Ford und Gaurav Rele.	15. Oktober 2021
Engine-Version 1.0.5.1	Ab dem 01.10.2021 wird die Engine-Version 1.0.5.1 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informati onen zu dieser Engine-Version finden Sie unter Neptune-E ngine-Version 1.0.5.1.	1. Oktober 2021
Blogbeitrag darüber, warum Entwickler TinkerPop	Lesen Sie, warum Entwickler Apache TinkerPop, ein Open-Source-Framework für Graph Computing, mögen, von Brad Bebee, Kelvin Lawrence und Stephen Mallette.	27. September 2021

Engine-Version 1.0.5.0.R3	Ab dem 15.09.2021 wird die Engine-Version 1.0.5.0.R3 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informationen zu dieser Engine-Version finden Sie unter Neptune-Engine-Version 1.0.5.0.R3.	15. September 2021
Blog-Beitrag zur Entwicklung einer Datenerkennungslösung mit Amundsen und Neptune	Weitere Informationen finden Sie unter Entwicklung einer Datenerkennungslösung mit Amundsen und Amazon Neptune von Peter Hanssens und Don Simpson.	8. September 2021
Neptune hat den Stream-Poller aktualisiert, um Volltext-Suchanfragen ohne Zeichenfolgen zu unterstützen	In dieser Version sind viele Verbesserungen der Volltextsuche enthalten, einschließlich der Unterstützung für die Indizierung von Eigenschaftswerten, bei denen es sich nicht um Zeichenfolgen handelt. Weitere Informationen finden Sie unter OpenSearch Indexierung ohne Zeichenketten in Amazon Neptune.	23. August 2021

Engine-Version 1.0.5.0.R2	Ab dem 16.08.2021 wird die Engine-Version 1.0.5.0.R 2 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informati onen zu dieser Engine-Version finden Sie unter Neptune-E ngine-Version 1.0.5.0.R2.	16. August 2021
Engine-Version 1.0.4.2.R5	Ab dem 16.08.2021 wird die Engine-Version 1.0.4.2.R 5 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informati onen zu dieser Engine-Version finden Sie unter Neptune-E ngine-Version 1.0.4.2.R5.	16. August 2021
Blog-Beitrag zur Unterstützung von Graph Store Protocol in Neptune	Weitere Informationen finden Sie unter Einführung in die Unterstützung von Graph Store Protocol für Amazon Neptune von Chris Smith.	2. August 2021
Blog-Beitrag zu neuen Features in Neptune ML	Weitere Informationen finden Sie unter Gewinnen von mehr Einblicken aus Graphen mit neuen Features in Amazon Neptune ML von Soji Adeshina.	30. Juli 2021

Blog-Beitrag zu schnelleren Prognosen mit Neptune ML	Weitere Informationen finden Sie unter Schnelleres Abrufen von Prognosen für sich entwickelnde Graph-Daten mit Amazon Neptune ML von Soji Adeshina.	30. Juli 2021
Blog-Beitrag zum einfacheren und schnelleren Machine Learning mit Graphen mit Neptune ML	Weitere Informationen finden Sie unter Einfacheres und schnelleres Machine Learning mit Graphen mit Amazon Neptune ML von Soji Adeshina.	30. Juli 2021
Blog-Beitrag zur openCypher-Unterstützung in Neptune	Weitere Informationen finden Sie unter Ankündigung von openCypher für Amazon Neptune: Entwicklung besserer Graph-Anwendungen durch die gemeinsame Nutzung von openCypher und Gremlin von Brad Bebee.	29. Juli 2021
Engine-Version 1.0.5.0	Ab dem 27.07.2021 wird die Engine-Version 1.0.5.0 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informationen zu dieser Engine-Version finden Sie unter Neptune-Engine-Version 1.0.5.0.	27. Juli 2021

Engine-Version 1.0.4.2.R4	Ab dem 23.07.2021 wird die Engine-Version 1.0.4.2.R 4 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informati onen zu dieser Engine-Version finden Sie unter Neptune-E ngine-Version 1.0.4.2.R4.	23. Juli 2021
Neptune-Start in China (Peking)	Amazon Neptune ist jetzt in China (Peking) (`cn-north-1`) verfügbar.	21. Juli 2021
Engine-Version 1.0.4.2.R3	Ab dem 28.06.2021 wird die Engine-Version 1.0.4.2.R 3 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informati onen zu dieser Engine-Version finden Sie unter Neptune-E ngine-Version 1.0.4.2.R3.	28. Juni 2021
Blog-Beitrag darüber, wie Dream11 sein soziales Netzwerk mithilfe von Neptune skaliert hat	Erfahren Sie, wie Dream11, die weltweit größte Fantasy-Sportplattform, ihr soziales Netzwerk mit Amazon Neptune und Amazon skaliert. ElastiCache	25. Juni 2021

Blogbeitrag über die Umwandlung von Daten in Wissen mit Neptune und PoolParty Semantic Suite	Weitere Informationen finden Sie unter Verwandeln Sie Daten mit PoolParty Semantic Suite und Amazon Neptune in Wissen von Ioanna Lytra und Albin Ahmeti.	16. Juni 2021
Blogbeitrag über die Nutzung von Neptune zur Erkundung der Wissensdatenbank UniProt	Siehe Erkundung der UniProt Protein-Wissensdatenbank mit AWS Open Data und Amazon Neptune von Eric Greene, Rafa Xu und Yuan Shi.	10. Juni 2021
Blog-Beitrag über die Verwendung von Neptune für datengesteuerte Risikoanalysen	Siehe Feldnotizen: Datengestützte Risikoanalyse mit Amazon Neptune und Amazon OpenSearch Service von Adriaan de Jonge und Rohit Satyanarayana.	10. Juni 2021
Engine-Version 1.0.4.2.R2	Ab dem 01.06.2021 wird die Engine-Version 1.0.4.2.R2 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informationen zu dieser Engine-Version finden Sie unter Neptune-Engine-Version 1.0.4.2.R2.	1. Juni 2021
Blogbeitrag über die Verwendung von Neptune zur Visualisierung Ihrer Infrastruktur AWS	Siehe Visualisieren Sie Ihre AWS Infrastruktur mit Amazon Neptune und AWS Config von Rohan Raizada und Amey Dhavle.	25. Mai 2021

Blog-Beitrag zur Konfiguration für die Verwendung von Data Lens mit Neptune	Weitere Informationen finden Sie unter Konfigurieren von AWS Diensten zur Erstellung eines Wissensgraphen in Amazon Neptune mithilfe von Data Lens von Russell Waterson.	5. Mai 2021
Blog-Beitrag zur Erstellung eines Wissensgraphen in Neptune mit Data Lens	Weitere Informationen finden Sie unter Erstellung eines Wissensgraphen in Amazon Neptune mit Data Lens von Russell Waterson.	5. Mai 2021
Die Engine-Versionen 1.0.1.0, 1.0.1.1 und 1.0.1.2 sind jetzt veraltet	Ab sofort wird keine neue DB-Instance mit einer dieser Engine-Versionen oder zugehörigen Patches mehr erstellt.	26. April 2021
Englische Übersetzung der Fallstudie zu Neptune vom japanischen Ministerium für Wirtschaft, Handel und Industrie	Weitere Informationen finden Sie unter Das japanische Ministerium für Wirtschaft, Handel und Industrie unterstützt die gBizINFO-Datenbank zur Suche nach Unternehmensinformationen mit AWS.	31. März 2021
Blog-Beitrag zur Verwendung von Neptune mit Amazon Comprehend und Lex	Weitere Informationen finden Sie unter Optimieren Ihres Wissensgraphen mit Amazon Neptune, Amazon Comprehend und Amazon Lex von Dave Bechberger.	31. März 2021

Blog-Beitrag zur Verwendung von Lambda-Funktionen mit Neptune	Siehe Verwenden von AWS Lambda-Funktionen mit Amazon Neptune von Ian Robinson.	26. März 2021
Engine-Version 1.0.4.1.R1.1	Ab dem 22.03.2021 wird die Engine-Version 1.0.4.1.R 1.1 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informationen zu dieser Engine-Version finden Sie unter Neptune-Engine-Version 1.0.4.1.R1.1.	22. März 2021
Engine-Version 1.0.4.1.R2.1	Ab dem 11.03.2021 wird die Engine-Version 1.0.4.1.R 2.1 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informationen zu dieser Engine-Version finden Sie unter Neptune-Engine-Version 1.0.4.1.R2.1.	11. März 2021
Blog-Beitrag zur Verwendung des Open-Source-Graph-Notebooks von Neptune zur Graph-Visualisierung	Weitere Informationen finden Sie unter Erste Schritte mit dem Open-Source-Graph-Notebook zur Graph-Visualisierung von Joy Wang, Ora Lassila und Stephen Mallette.	10. März 2021

Tutorial zur Integration von Neptune mit der Datenerfassungs- und Metadaten-Engine von Amundsen	Weitere Informationen finden Sie unter Verwendung von Amundsen mit Amazon Neptune von Andrew Ciambrone.	2. März 2021
Engine-Version 1.0.4.1.R2	Ab dem 24.02.2021 wird die Engine-Version 1.0.4.1.R2 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informationen zu dieser Engine-Version finden Sie unter Neptune-Engine-Version 1.0.4.1.R2.	24. Februar 2021
Engine-Version 1.0.4.0.R2	Ab dem 24.02.2021 wird die Engine-Version 1.0.4.0.R2 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informationen zu dieser Engine-Version finden Sie unter Neptune-Engine-Version 1.0.4.0.R2.	24. Februar 2021

Engine-Version 1.0.3.0.R3	Ab dem 19.02.2021 wird die Engine-Version 1.0.3.0.R3 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informationen zu dieser Engine-Version finden Sie unter Neptune-Engine-Version 1.0.3.0.R3.	19. Februar 2021
Engine-Version 1.0.2.2.R6	Ab dem 19.02.2021 wird die Engine-Version 1.0.2.2.R6 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informationen zu dieser Engine-Version finden Sie unter Neptune-Engine-Version 1.0.2.2.R6.	19. Februar 2021
Blog-Beitrag über die Erstellung eines Wissensgraphen mithilfe von Amazon Comprehend Events	Weitere Informationen finden Sie unter Erstellen eines Wissensgraphen in Amazon Neptune mithilfe von Amazon Comprehend Events von Brian O'Keefe, Graham Horwood und Navtanay Sinha.	19. Januar 2021
Blog-Beitrag zur Aktivierung von Low-Code-Graph-Daten-Apps	Weitere Informationen finden Sie unter Aktivieren von Low-Code-Graph-Daten-Apps mit Amazon Neptune und Graphistry von Leo Meyerovich, Dave Bechberger und Taylor Riggan.	18. Januar 2021

Notebook-Dokumentation für den Einstieg mit Graph-Daten hinzugefügt.	Es wurde ein Abschnitt hinzugefügt, der in die Neptune-Workbench integriert ist und Ihnen hilft, mit der Erstellung von Graphdaten und der Entwicklung von Graphanwendungen zu beginnen, ohne einen Neptune-Cluster starten zu müssen, bis Sie bereit sind.	15. Januar 2021
Blog-Beitrag zum Zurücksetzen Ihrer Neptune-Graph-Daten in Sekunden	Weitere Informationen finden Sie unter Zurücksetzen Ihrer Graph-Daten in Amazon Neptune in Sekunden von Niraj Jetly und Navtanay Sinha.	17. Dezember 2020
Blogbeitrag darüber, wie die Novartis AG Neptune mit SageMaker BERT verwendet	Siehe Novartis AG verwendet Amazon SageMaker und Amazon Neptune, um mithilfe von BERT einen Wissensgraphen zu erstellen und zu erweitern, von Othmane Hamzaoui, Fatema Alkhanaizi und Viktor Malesevic.	14. Dezember 2020
Blog-Beitrag zur Erstellung eines Wissensgraphen mit Themennetzwerken	Weitere Informationen finden Sie unter Aufbau eines Wissensgraphen mit Themennetzwerken in Amazon Neptune von Edward Brown, Head of AI Projects, Eduardo Piairo, Architekt, Marcia Oliveira, Lead Data Scientist, und Jack Hampson, CEO bei Deeper Insights.	14. Dezember 2020

Engine-Version 1.0.4.1	Ab dem 08.12.2020 wird die Engine-Version 1.0.4.1 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informationen zu dieser Engine-Version finden Sie unter Neptune-Engine-Version 1.0.4.1.	08. Dezember 2020
Blog-Beitrag zu den ersten Schritten mit Neptune ML	Weitere Informationen finden Sie unter Erste Schritte mit Neptune ML von George Karypis, Dave Bechberger und Karthik Bharathy.	08. Dezember 2020
Neptune hat jetzt eine Fast-Reset-API	Mit der Fast-Reset-API können Sie schnell und einfach alle Daten in einem DB-Cluster löschen. Siehe API für schnelles Zurücksetzen.	4. Dezember 2020
Blog-Beitrag über die Entwicklung eines biologischen Wissensgraphen bei Pendulum	Weitere Informationen finden Sie unter Entwicklung eines biologischen Wissensgraphen bei Pendulum mit Amazon Neptune von Connor Skennerton.	26. November 2020
Blogbeitrag über die neuen Funktionen von TinkerPop 3.4.8 in Neptune	Weitere Informationen finden Sie unter Erkunden der neuen Funktionen von Apache TinkerPop 3.4.8 in Amazon Neptune von Stephen Mallette.	18. November 2020

Blog-Beitrag zur Verwendung des Amazon-Kendra-Suchservices mit Neptune	Weitere Informationen finden Sie unter Integrieren Ihres Unternehmens-Wissensgraphen in Amazon Kendra von Yazdan Shirvany, Mohit Mehta und Dipto Chakravarty.	17. November 2020
Ereignisbenachrichtigungen sind jetzt verfügbar	Neptune unterstützt jetzt Ereignisbenachrichtigungen, mit denen Sie DB-Cluster einfacher überwachen können. Weitere Informationen finden Sie unter Verwenden von Neptune-Ereignisbenachrichtigungen.	29. Oktober 2020
Benutzerdefinierte Endpunkte sind jetzt verfügbar	Neptune unterstützt jetzt benutzerdefinierte Endpunkte für eine bessere Kontrolle der Verbindung zu DB-Instances. Weitere Informationen finden Sie unter Verbinden mit Amazon-Neptune-Endpunkten.	29. Oktober 2020
Blogbeitrag zur Verwendung des AWS Database Migration Service (DMS) zum Auffüllen Ihres Neptune-Diagramms	Weitere Informationen finden Sie unter Auffüllen Ihres Diagramms in Amazon Neptune aus einer relationalen Datenbank mithilfe des AWS Database Migration Service (DMS) — Teil 4: Alles zusammenfügen von Chris Smith.	22. Oktober 2020

Blogbeitrag zur Verwendung des AWS Database Migration Service (DMS) zum Auffüllen Ihres Neptune-Diagramms	Weitere Informationen finden Sie unter Auffüllen Ihres Diagramms in Amazon Neptune aus einer relationalen Datenbank mithilfe des AWS Database Migration Service (DMS) — Teil 3: Entwerfen des RDF-Modells von Chris Smith.	22. Oktober 2020
Blogbeitrag zur Verwendung des AWS Database Migration Service (DMS) zum Auffüllen Ihres Neptune-Diagramms	Weitere Informationen finden Sie unter Auffüllen Ihres Diagramms in Amazon Neptune aus einer relationalen Datenbank mithilfe des AWS Database Migration Service (DMS) — Teil 2: Entwerfen des Eigenschaftsdiagrammmodells von Chris Smith.	22. Oktober 2020
Blogbeitrag zur Verwendung des AWS Database Migration Service (DMS) zum Auffüllen Ihres Neptune-Diagramms	Weitere Informationen finden Sie unter Auffüllen Ihres Diagramms in Amazon Neptune aus einer relationalen Datenbank mithilfe des AWS Database Migration Service (DMS) — Teil 1: Die Voraussetzungen schaffen von Chris Smith.	22. Oktober 2020

Engine-Version 1.0.4.0	Ab dem 12.10.2020 wird die Engine-Version 1.0.4.0 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informationen zu dieser Engine-Version finden Sie unter Neptune-Engine-Version 1.0.4.0.	12. Oktober 2020
Engine-Version 1.0.3.0.R2	Ab dem 12.10.2020 wird die Engine-Version 1.0.3.0.R2 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informationen zu dieser Engine-Version finden Sie unter Neptune-Engine-Version 1.0.3.0.R2.	12. Oktober 2020
Engine-Version 1.0.2.2.R5	Ab dem 12.10.2020 wird die Engine-Version 1.0.2.2.R5 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informationen zu dieser Engine-Version finden Sie unter Neptune-Engine-Version 1.0.2.2.R5.	12. Oktober 2020

Blog-Beitrag zur Konfiguration Ihrer VPC für SPARQL-Verbundabfragen	Weitere Informationen finden Sie unter Konfiguration von Amazon VPC für SPARQL-1.1-Verbundabfragen mit Amazon Neptune von Charles Ivie.	12. Oktober 2020
Blog-Beitrag zum Schreiben eines kaskadierenden SPARQL-Löschvorgangs	Weitere Informationen finden Sie unter Schreiben eines kaskadierenden Löschvorgangs in SPARQL von Ora Lassila.	5. Oktober 2020
Blogbeitrag zur grafischen Darstellung von AWS Ressourcen mit Neptune	Siehe Graph Your AWS Resources with Amazon Neptune von Dave Bechberger.	28. September 2020
Blog-Beitrag zur Erstellung einer MedDRA-Terminologiezuordnung für Pharmakovigilanz und die Meldung von Nebenwirkungen mit Neptune	Weitere Informationen finden Sie unter Erstellung einer auf Amazon Neptune basierenden MedDRA-Terminologiezuordnung für Pharmakovigilanz und die Meldung von Nebenwirkungen von Vaijayanti Joshi, Deven Atnoor, Ph.D., und Sudhanshu Malhotra.	24. September 2020
Blog-Beitrag zur Erstellung eines Wissensgraphen aus einem Data Warehouse mithilfe von Neptune als Ergänzung zu kommerzieller Intelligenz	Weitere Informationen finden Sie unter Ergänzung kommerzieller Intelligenz durch Erstellung eines Wissensgraphen aus einem Data Warehouse mit Amazon Neptune von Shahria Hossain und Mikael Graindorge.	23. September 2020

Blog-Beitrag zum Load Balancing mit dem Neptune-Gremlin-Client	Weitere Informationen finden Sie unter Load Balance bei Graphabfragen mit dem Amazon-Neptune-Gremlin-Client von Ian Robinson.	16. September 2020
Blog-Beitrag zur digitalen Personalisierung mithilfe eines Identitätsgraphen bei Cox Automotive	Weitere Informationen finden Sie unter Cox Automotive skaliert die digitale Personalisierung mithilfe eines Identitätsgraphen mit Unterstützung durch Amazon Neptune von Carlos Rendon und Niraj Jetly.	16. September 2020
Blog-Beitrag über kollaboratives Filtern von Yelp-Daten	Weitere Informationen finden Sie unter Verwendung zusammenarbeitsorientierter Filterung von Yelp-Daten zum Aufbau eines Empfehlungssystems in Amazon Neptune von Chad Tindel.	8. September 2020
Blog-Beitrag zur Visualisierung von Abfrageergebnissen in Amazon Neptune	Weitere Informationen finden Sie unter Visualisieren von Abfrageergebnissen mithilfe der Amazon-Neptune-Workbench von Kelvin Lawrence.	2. September 2020

Neptune hat die Graphvisualisierung veröffentlicht	Amazon Neptune bietet jetzt umfangreiche Funktionen zur Visualisierung von Graphen in Jupyter-Notebooks in der Neptune Workbench sowie eine Reihe neuer Features, die die Verwendung von Notebooks vereinfachen. Weitere Informationen finden Sie unter Graphvisualisierung.	12. August 2020
Neptune-Start in Südamerika (São Paulo)	Amazon Neptune ist jetzt in Südamerika (São Paulo) (`sa-east-1`) verfügbar.	6. August 2020
Neptune-Start in Asien-Pazifik (Hongkong)	Amazon Neptune ist jetzt in Asien-Pazifik (Hongkong) (`ap-east-1`) verfügbar.	6. August 2020
Engine-Version 1.0.3.0	Ab dem 03.08.2023 wird die Engine-Version 1.0.3.0 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informationen zu dieser Engine-Version finden Sie unter Neptune-Engine-Version 1.0.3.0.	3. August 2020

Engine-Version 1.0.2.2.R4	Ab dem 23.07.2020 wird die Engine-Version 1.0.2.2.R 4 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informati onen zu dieser Engine-Version finden Sie unter Neptune-E ngine-Version 1.0.2.2.R4.	23. Juli 2020
Blog-Beitrag über die automatisierte Kontaktve rfolgung mit Amazon Neptune bei Zerobase	Weitere Informationen finden Sie unter Zerobase erstellt private, sichere und automatis ierte Kontaktverfolgung mithilfe von Amazon Neptune von David Harris und Aron Szanto.	13. Juli 2020
Neptune-Start in USA West (Nordkalifornien)	Amazon Neptune ist jetzt in USA West (Nordkalifornien) (`us-west-1`) verfügbar.	9. Juli 2020
Amazon Neptune unterstüt zt die tag-basierte Zugriffsk ontrolle	Sie können jetzt AWS Tags in IAM-Richtlinien verwenden , um den Zugriff auf Ihre Neptune-Datenbank zu kontrollieren. Siehe Tag-basie rte Zugriffskontrolle in Amazon Neptune.	7. Juli 2020

Ein Java-Stream-Poller ist jetzt verfügbar	Amazon Neptune unterstützt jetzt eine Java-Version des Lambda-Stream-Pollers für Neptune-Streams sowie die Python-Version. Weitere Informationen finden Sie unter Hinzufügen von Details zu dem von Ihnen erstellten Neptune-Streams-Consumer-Stack.	6. Juli 2020
Blogbeitrag über den AWS COVID-19-Knowledge-Graph	Weitere Informationen finden Sie unter Erstellung und Abfrage des AWS COVID-19-Wissensgraphen von Ninad Kulkarni, Colby Wise, George Price und Miguel Romero.	1. Juli 2020
Engine-Version 1.0.1.1	Ab dem 26.06.2020 wird die Engine-Version 1.0.1.1 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informationen zu dieser Engine-Version finden Sie unter Neptune-Engine-Version 1.0.1.1.	26. Juni 2020
Blog-Beitrag über die Migration von Blazegraph zu Amazon Neptune	Siehe Moving to the cloud: Migrating Blazegraph to Amazon Neptune von Dave Bechberger.	25. Juni 2020

Blog-Beitrag zum Ändern der Datenerfassung von Neo4j zu Amazon Neptune	Siehe Change data capture from Neo4j to Amazon Neptune using Amazon Managed Streaming for Apache Kafka von Sanjeet Sahay.	22. Juni 2020
Blog-Beitrag darüber, wie Waves Amazon Neptune verwendet	Siehe How Waves runs user queries and recommendations at scale with Amazon Neptune von Pavel Vasilyev.	16. Juni 2020
Engine-Version 1.0.1.2	Ab dem 10.06.2020 wird die Engine-Version 1.0.1.2 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informationen zu dieser Engine-Version finden Sie unter Neptune-Engine-Version 1.0.1.2.	10. Juni 2020
Blog-Beitrag zum Aufbau eines Kunden-Wissens-Repositorys	Siehe Building a customer 360 knowledge repository with Amazon Neptune and Amazon Redshift von Ram Bhandarkar.	9. Juni 2020
Blog-Beitrag darüber, wie Gunosy Amazon Neptune verwendet	Siehe How Gunosy built a comment feature in News Pass using Amazon Neptune von Yosuke Uchiyama.	08. Juni 2020

Blogbeitrag über den COVID-19-Knowledge-Graph AWS	Weitere Informationen findest du unter Erstellung und Abfrage des AWS COVID-19-Wissensgraphen von Ninad Kulkarni, Colby Wise, George Price und Miguel Romero.	2. Juni 2020
Blog-Beitrag über die Erkundung der COVID-19-Forschung mit Amazon Neptune	Siehe Erkundung der wissenschaftlichen Forschung zu COVID-19 mit Amazon Neptune, Amazon Comprehend Medical und dem Tom Sawyer Graph Database Browser von George Price, Colby Wise, Miguel Romero und Ninad Kulkarni.	2. Juni 2020
Sie können jetzt Daten in Neptune laden mit AWS DMS	Weitere Informationen finden Sie unter Verwenden des AWS Database Migration Service zum Laden von Daten aus einem anderen Datenspeicher in Amazon Neptune.	1. Juni 2020
Engine-Version 1.0.2.0 ist veraltet	Die Amazon-Neptune-Engine-Version 1.0.2.0 ist jetzt veraltet. Cluster, die auf dieser Engine-Version ausgeführt werden, werden ab dem 1. Juni 2020 während des ersten Wartungsfensters automatisch auf Version 1.0.2.1 aktualisiert.	19. Mai 2020
Blog-Beitrag zum Erstellen eines Kundenidentitätsgraphen mit Neptune	Siehe Erstellen eines Kundenidentitätsgraphen mit Amazon Neptune von Rajesh Wunnava und Taylor Riggan.	12. Mai 2020

Engine-Version 1.0.2.0.R3	Ab dem 05.05.2022 wird die Engine-Version 1.0.2.0.R3 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informationen zu dieser Engine-Version finden Sie unter Neptune-Engine-Version 1.0.2.0.R3.	5. Mai 2020
Engine-Version 1.0.2.1.R6	Ab dem 22.04.2020 wird die Engine-Version 1.0.2.1.R6 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informationen zu dieser Engine-Version finden Sie unter Neptune-Engine-Version 1.0.2.1.R6.	22. April 2020
Blog-Beitrag über die Migration von Daten von Neo4j zu Neptune	Siehe Migrating a Neo4j graph database to Amazon Neptune with a fully automated utility von Sanjeet Sahay.	13. April 2020
Blog-Beitrag zur Senkung der Kosten für das Erstellen von Graph-Anwendungen mit Neptune	Siehe Lower the cost of building graph apps by up to 76% with Amazon Neptune T3 instances von Karthik Bharathy und Brad Bebee.	09. April 2020

Neptune bietet eine T3-Burstable-Instance-Klasse	Sie können jetzt eine Amazon-Neptune-T3-Burstable-Instance für kostengünstige Entwicklungs- und Testzwecke erstellen. Siehe Neptune T3 Burstable Instance-Klasse.	8. April 2020
Engine-Version 1.0.2.2.R2	Ab dem 02.04.2020 wird die Engine-Version 1.0.2.2.R2 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informationen zu dieser Engine-Version finden Sie unter Neptune-Engine-Version 1.0.2.2.R2.	2. April 2020
Blog-Beitrag zur grafischen Darstellung der Investitionsabhängigkeit bei EDGAR	Siehe Graphing investment dependency with Amazon Neptune von Lawrence Verdi.	17. März 2020
Neptune-Start in Europa (Paris)	Amazon Neptune ist jetzt in Europa (Paris) (eu-west-3) verfügbar.	11. März 2020
Engine-Version 1.0.2.2	Ab dem 09.03.2020 wird die Engine-Version 1.0.2.2 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist. Weitere Informationen zu dieser Engine-Version finden Sie unter Neptune-Engine-Version 1.0.2.2.	9. März 2020

Beenden und Neustarten eines DB-Clusters	Sie können einen DB-Cluster jetzt für 7 Tage mithilfe der Neptun-Konsole anhalten und später erneut starten, wenn Sie ihn wieder benötigen. Solange Ihr DB-Cluster angehalten ist, werden Ihnen nur die Cluster-Speicherung, manuelle Snapshots und die automatisierte Backup-Speicherung berechnet, nicht jedoch die DB-Instance-Stunden. Siehe Anhalten und Starten eines DB-Clusters von Amazon Neptune.	19. Februar 2020
Video über einen sozialen Graphen bei Nike	Hören Sie zu, wie Todd Escalona mit Marc Wangenheim, Senior Engineering Manager bei Nike, darüber AWS spricht, wie das Unternehmen eine Reihe von Anwendungen über ein auf Amazon Neptune basierendes Social Graph unterstützt. Siehe Nike: A Social Graph at Scale with Amazon Neptune.	11. Februar 2020

Neptune-Cluster können jetzt so konfiguriert werden, dass SSL-Verbindungen erforderlich sind.	In Regionen, die weiterhin HTTP-Verbindungen unterstützen, ist SSL jetzt standardmäßig in allen neuen Parametergruppen aktiviert. An vorhandenen Parameter gruppen wurden zwar keine Änderungen vorgenommen, Sie können jedoch Clients zur Verwendung von SSL zwingen, indem Sie den `neptune_enforce_ssl` -Parameter auf 1 ändern. Weitere Informationen zum Aktivieren von HTTP-Verb indungen für einen Cluster in einer Region, die diese weiterhin unterstützt, finden Sie unter Verschlüsselung während der Übertragung: Herstellen einer Verbindung zu Neptune mittels SSL/HTTPS. Eine Beschreibung der Cluster- und Instance-Parameter finden Sie unter Zum Konfigurieren von Amazon Neptune verwendbare Parameter.	10. Februar 2020

Sie können jetzt die Engine-Version und den Löschschutz in der Neptune-Vorlage angeben CloudFormation	Amazon Neptune hat seine CloudFormation Vorlage aktualisiert und enthält nun einen `AWS::Neptune::DBCluster.EngineVersion` Parameter, mit dem Sie eine bestimmte Engine-Version für Ihren neuen DB-Cluster angeben können, sowie einen `AWS::Neptune::DBCluster.DeletionProtection` Parameter, mit dem Sie den Löschschutz für diesen Cluster aktivieren können.	9. Februar 2020
Löschschutz	Amazon Neptune hat einen Löschschutz für DB-Cluster und Instances bereitgestellt. Solange der Löschschutz für einen DB-Cluster oder eine Instance aktiviert ist, können Sie ihn/sie nicht löschen. Vgl. Sie können eine DB-Instance nicht löschen, wenn Löschschutz aktiviert ist.	20. Januar 2020
Neptune-Start in China (Ningxia)	Amazon Neptune ist jetzt in China (Ningxia) (cn-northwest-1) verfügbar.	15. Januar 2020

Engine-Version 1.0.2.1.R4	Patch R4 für die Engine-Version 1.0.2.1 ist allgemein verfügbar. Weitere Informationen finden Sie unter Neptune-Engine-Version 1.0.2.1.R4.	20. Dezember 2019
Engine-Version 1.0.2.1.R3	Patch R3 für die Engine-Version 1.0.2.1 ist allgemein verfügbar. Weitere Informationen finden Sie unter Neptune-Engine-Version 1.0.2.1.R3.	12. Dezember 2019
Blog-Beitrag über die Verwendung von Neptune zur Analyse von Social-Media-Feeds	Weitere Informationen unter Analyzing social media feeds using Amazon Neptune (Analysieren von Social-Media-Feeds mit Amazon Neptune).	27. November 2019
Engine-Version 1.0.2.1.R2	Patch R2 für die Engine-Version 1.0.2.1 ist allgemein verfügbar. Weitere Informationen finden Sie unter Neptune-Engine-Version 1.0.2.1.R2.	25. November 2019
Engine-Version 1.0.2.1.R1	Amazon-Neptune-Engine-Version 1.0.2.1.R1 ist allgemein verfügbar. Weitere Informationen finden Sie unter Neptune-Engine-Version 1.0.2.1.	22. November 2019

Engine-Version 1.0.2.0.R2	Patch R2 für die Engine-Version 1.0.2.0 ist allgemein verfügbar. Weitere Informationen finden Sie unter Neptune-Engine-Version 1.0.2.0.R2.	21. November 2019
Blog-Beitrag über Neptune-Sitzungen und Workshops auf der re:Invent 2019	Sehen Sie sich Ihren Leitfaden für Amazon Neptune Neptune-Sessions, Workshops und Chalk-Talks auf der re:Invent 2019 an AWS.	20. November 2019
Engine-Version 1.0.2.0.R1	Amazon-Neptune-Engine-Version 1.0.2.0.R1 ist allgemein verfügbar. Weitere Informationen finden Sie unter Neptune-Engine-Version 1.0.2.0.	8. November 2019
Blog-Beitrag zum Erfassen von Graph-Änderungen mit Neptune Streams	Siehe Capture Graph Changes using Neptune Streams.	6. November 2019
Engine-Version 1.0.1.0.200502.0	Die Amazon-Neptune-Engine-Version 1.0.1.0.200502.0 ist allgemein verfügbar. Weitere Informationen finden Sie unter Update 1.0.1.0.200502.0.	31. Oktober 2019
Neptune-Start im Nahen Osten (Bahrain)	Amazon Neptune ist jetzt in der Region Naher Osten (Bahrain) (me-south-1) verfügbar.	30. Oktober 2019
Neptune-Start in Kanada (Zentral)	Amazon Neptune ist jetzt in Kanada (Zentral) verfügbar (ca-central-1).	30. Oktober 2019

Blog-Beitrag über das neue Neptune-Feature SPARQL Streams und die Unterstützung von SPARQL-Verbundabfragen	Siehe Amazon Neptune releases Streams, SPARQL federated query for graphs and more.	17. Oktober 2019
Engine-Version 1.0.1.0.200463.0	Die Amazon-Neptune-Engine-Version 1.0.1.0.200463.0 ist allgemein verfügbar. Weitere Informationen finden Sie unter Update 1.0.1.0.200463.0.	15. Oktober 2019
Engine-Version 1.0.1.0.200457.0	Die Amazon-Neptune-Engine-Version 1.0.1.0.200457.0 ist allgemein verfügbar. Weitere Informationen finden Sie unter Update 1.0.1.0.200457.0.	19. September 2019
Blog-Beitrag zum neuen Feature SPARQL Explain von Neptune	Vgl. Verwendung von SPARQL Explain zur Erläuterung der Abfrageausführung in Amazon Neptune.	17. September 2019
Blogbeitrag über Neptune-Unterstützung für 3.4 TinkerPop	Siehe Amazon Neptune unterstützt jetzt TinkerPop 3.4-Funktionen.	6. September 2019
Blogbeitrag zur Verwendung von Neptune mit bei Amazon PyTorch SageMaker	Weitere Informationen finden Sie unter Ein personalisiertes Nutzererlebnis PyTorch bei Amazon SageMaker und Amazon Neptune. shop-by-style	22. August 2019

Blogbeitrag zur Verwendung von Neptune mit AWS AppSync und Amazon Elasticache	Weitere Informationen finden Sie unter Integration alternativer Datenquellen mit AWS AppSync: Amazon Neptune und Amazon. ElastiCache	22. August 2019
Neptune wurde in AWS GovCloud (US-Ost) gestartet	Amazon Neptune ist jetzt in AWS GovCloud (US-Ost) (us-gov-east-1) verfügbar.	21. August 2019
Neptune wurde in AWS GovCloud (US-West) gestartet	Amazon Neptune ist jetzt in AWS GovCloud (US-West) (us-gov-west-1) verfügbar.	14. August 2019
Engine-Version 1.0.1.0.200369.0	Die Amazon-Neptune-Engine-Version 1.0.1.0.200369.0 ist allgemein verfügbar. Weitere Informationen finden Sie unter Update 1.0.1.0.200369.0.	13. August 2019
Engine-Version 1.0.1.0.200366.0	Die Amazon-Neptune-Engine-Version 1.0.1.0.200366.0 ist allgemein verfügbar. Weitere Informationen finden Sie unter Update 1.0.1.0.200366.0.	26. Juli 2019
Blogbeitrag zur Verwendung von Neptune mit bei Amazon PyTorch SageMaker	Weitere Informationen finden Sie unter Ein personalisiertes Nutzererlebnis PyTorch bei Amazon SageMaker und Amazon Neptune. shop-by-style	3. Juli 2019
Engine-Version 1.0.1.0.200348.0	Die Amazon-Neptune-Engine-Version 1.0.1.0.200348.0 ist allgemein verfügbar. Weitere Informationen finden Sie unter Update 1.0.1.0.200348.0.	2. Juli 2019

Neptune-Start in Europa (Stockholm)	Amazon Neptune ist jetzt in Europa (Stockholm) (eu-north-1) verfügbar.	27. Juni 2019
Neptune kann jetzt Audit-Logs in Logs veröffentlichen CloudWatch	Weitere Informationen finden Sie unter Neptune Logs in Amazon CloudWatch Logs veröffentlichen.	18. Juni 2019
Engine-Version 1.0.1.0.200310.0	Die Amazon-Neptune-Engine-Version 1.0.1.0.200310.0 ist allgemein verfügbar. Weitere Informationen finden Sie unter Update 1.0.1.0.200310.0.	12. Juni 2019
Blogbeitrag über's LifeOmic JupiterOne	Erfahren Sie, wie LifeOmic Amazon Neptune JupiterOne die Sicherheits- und Compliance-Abläufe vereinfacht.	2. Mai 2019
Neptune-Start in Asien-Pazifik (Seoul)	Amazon Neptune ist jetzt in der Region Asien-Pazifik (Seoul) (ap-northeast-2) verfügbar.	1. Mai 2019
Engine-Version 1.0.1.0.200296.0	Die Amazon-Neptune-Engine-Version 1.0.1.0.200296.0 ist allgemein verfügbar. Weitere Informationen finden Sie unter Update 1.0.1.0.200296.0.	1. Mai 2019
Neptune-Start in Asien-Pazifik (Mumbai)	Amazon Neptune ist jetzt in der Region Asien-Pazifik (Mumbai) (ap-south-1) verfügbar.	6. März 2019

Blog-Beitrag zu Gremlin-Abfragehinweisen	Siehe Gremlin-Abfragehinweise für Amazon Neptune.	26. Februar 2019
Neptune-Start in Asien-Pazifik (Tokio)	Amazon Neptune ist jetzt in der Region Asien-Pazifik (Tokio) (ap-northeast-1) verfügbar.	23. Januar 2019
CloudFormation Vorlage für die Erstellung einer AWS Lambda Funktion für den Zugriff auf Neptune	Der Abschnitt „Erste Schritte“ wurde aktualisiert und eine CloudFormation Vorlage hinzugefügt, um eine Lambda-Funktion zur Verwendung mit Neptune zu erstellen. Weitere Informationen finden Sie unter Erste Schritte mit Neptune.	23. Januar 2019
Engine-Version 1.0.1.0.200267.0	Die Amazon-Neptune-Engine-Version 1.0.1.0.200267.0 ist allgemein verfügbar. Weitere Informationen finden Sie unter Update 1.0.1.0.200267.0.	21. Januar 2019
Neptune-Start in Asien-Pazifik (Sydney)	Amazon Neptune ist jetzt in der Region Asien-Pazifik (Sydney) (ap-southeast-2) verfügbar.	9. Januar 2019
Blog-Beitrag zur Verwendung von Metaphactory	Siehe Wissensdiagramme auf Amazon Neptune mit Metaphactory.	9. Januar 2019
Neptune-Start in Asien-Pazifik (Singapur)	Amazon Neptune ist jetzt in Asien-Pazifik (Singapur) (ap-southeast-1) verfügbar.	13. Dezember 2018

Engine-Version 1.0.1.0.200264.0	Die Amazon-Neptune-Engine-Version 1.0.1.0.200264.0 ist allgemein verfügbar. Weitere Informationen finden Sie unter Update 1.0.1.0.200264.0.	19. November 2018
Amazon-Neptune-SSL-Unterstützung	Neptune unterstützt jetzt SSL-Verbindungen.	19. November 2018
Konsolidierung der Fehler-Themen	Alle Fehlermeldungen und Code-Informationen sind nun in einem einzigen Thema zusammengefasst.	15. November 2018
Aktualisierung des Themas „Erste Schritte“	Das Thema „Erste Schritte“ wurde mit zusätzlichen Links aktualisiert und neu organisiert.	14. November 2018
Engine-Version 1.0.1.0.200258.0	Die Amazon-Neptune-Engine-Version 1.0.1.0.200258.0 ist allgemein verfügbar. Weitere Informationen finden Sie unter Update 1.0.1.0.200258.0.	8. November 2018
Neptune-Start in Europa (Frankfurt)	Amazon Neptune ist jetzt in Europa (Frankfurt) (eu-central-1) verfügbar.	7. November 2018
Blog-Beitrag Nr. 1 in einer Reihe	Siehe Lassen Sie mich einen Graphen dafür erstellen – Teil 1 – Flugstrecken.	7. November 2018

Blogbeitrag zur Verwendung von Amazon SageMaker Jupyter Notebooks	Weitere Informationen finden Sie unter Analysieren von Amazon Neptune Neptune-Diagrammen mit Amazon SageMaker Jupyter Notebooks .	1. November 2018
Engine-Version 1.0.1.0.200255.0	Die Amazon-Neptune-Engine-Version 1.0.1.0.200255.0 ist allgemein verfügbar. Weitere Informationen finden Sie unter Update 1.0.1.0.200255.0.	29. Oktober 2018
Neptune-Start in Europa (London)	Amazon Neptune ist jetzt in Europa (London) (eu-west-2) verfügbar.	3. Oktober 2018
Engine-Version 1.0.1.0.200237.0	Die Amazon-Neptune-Engine-Version 1.0.1.0.200237.0 ist allgemein verfügbar. Weitere Informationen finden Sie unter Update 1.0.1.0.200237.0.	6. September 2018
Engine-Version 1.0.1.0.200236.0	Die Amazon-Neptune-Engine-Version 1.0.1.0.200236.0 ist allgemein verfügbar. Weitere Informationen finden Sie unter Update 1.0.1.0.200236.0.	24. Juli 2018
Engine-Version 1.0.1.0.200233.0	Die Amazon-Neptune-Engine-Version 1.0.1.0.200233.0 ist allgemein verfügbar. Weitere Informationen finden Sie unter Update 1.0.1.0.200233.0.	22. Juni 2018

Neuer Neptune-Schnellstart	Der Schnellstart mit CloudFormation und das Tutorial zur Gremlin-Konsole wurden aktualisiert. Weitere Informationen finden Sie unter Amazon Neptune Quick Start Using. CloudFormation	19. Juni 2018
Erstveröffentlichung von Amazon Neptune	Dies ist die erste Version des Neptune Benutzerhandbuchs. Weitere Informationen finden Sie im Blogbeitrag anlässlich der Veröffentlichung: Amazon Neptune allgemein verfügbar.	30. Mai 2018
Blog-Beitrag mit einer Einführung zu Neptune	Siehe Amazon Neptune – ein vollständig verwalteter Graphdatenbank-Service.	29. November 2017

Erste Schritte mit Amazon Neptune

Amazon Neptune ist ein vollständig verwalteter Graphdatenbank-Service, der für Milliarden von Beziehungen skaliert werden kann und es Ihnen ermöglicht, diese mit einer Latenz von Millisekunden abzufragen, und dies zu geringen Kosten für diese Art von Kapazität.

Weitere Informationen zu Neptune finden Sie unter Übersicht über Amazon-Neptune-Features.

Wenn Sie sich bereits mit Diagrammen auskennen, fahren Sie mit oder fort. Schneller Start mit CloudShell Neptune mit Grafiknotizbüchern verwenden Wenn Sie sofort eine Neptune-Datenbank erstellen möchten, finden Sie weitere Informationen unter Erstellen eines Amazon Neptune Neptune-Clusters mit AWS CloudFormation.

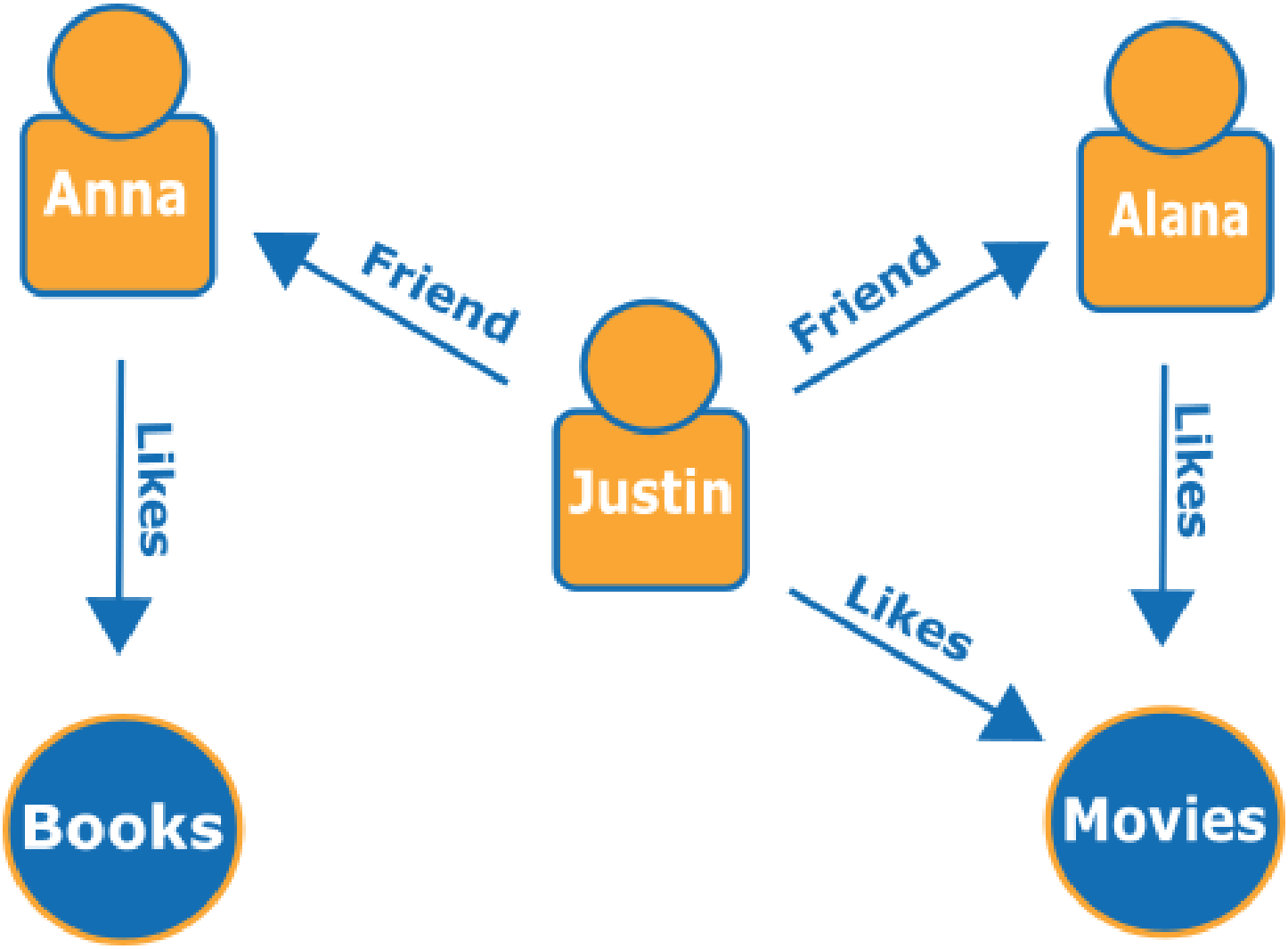
Andernfalls möchten Sie vielleicht etwas mehr über Graphdatenbanken wissen, bevor Sie beginnen.

Die wichtigsten Konzepte von Graphdatenbanken

Graphdatenbanken sind für das Speichern und Abfragen der Beziehungen zwischen Datenelementen optimiert.

Sie speichern Datenelemente selbst als Scheitelpunkte des Graphen und die Beziehungen zwischen ihnen als Edges. Jede Edge hat einen Typ und verläuft von einem Vertex (dem Anfang) zum anderen (dem Ende). Beziehungen können sowohl als Prädikate als auch als Edges bezeichnet werden und Scheitelpunkte werden manchmal auch als Knoten bezeichnet. In sogenannten Eigenschaftsgraphen können sowohl Scheitelpunkten als auch Edges zusätzliche Eigenschaften zugeordnet werden.

Hier ist ein kleiner Graph, der Freunde und Hobbys in einem sozialen Netzwerk darstellt:



Die Edges werden als benannte Pfeile dargestellt und die Scheitelpunkte stehen für bestimmte Personen und Hobbys, die sie miteinander verbinden.

Mit einer einfachen Verschiebung dieses Graphen erfahren Sie, was die Freunde von Justin mögen.

Warum sollte ich eine Graphdatenbank verwenden?

Wenn Verbindungen oder Beziehungen zwischen Entitäten zum Kern der Daten gehören, die Sie darstellen möchten, ist eine Graphdatenbank die logische Konsequenz.

Zum einen ist es einfach, Datenverbindungen als Graph zu modellieren und dann komplexe Abfragen zu schreiben, die reale Informationen aus dem Graphen extrahieren.

Um eine äquivalente Anwendung mithilfe einer relationalen Datenbank zu erstellen, müssen Sie zahlreiche Tabellen mit mehreren Fremdschlüsseln erstellen und dann verschachtelte SQL-Abfragen

und komplexe Verknüpfungen schreiben. Dieser Ansatz wird nicht nur aus Sicht der Programmierung schnell unhandlich, auch nimmt seine Leistung schnell ab, wenn die Datenmenge zunimmt.

Im Gegensatz dazu kann eine Graphdatenbank wie Neptune Beziehungen zwischen Milliarden von Scheitelpunkten abfragen, ohne dabei ins Stocken zu geraten.

Was können Sie mit einer Graphdatenbank tun?

Graphen können die Wechselbeziehungen zwischen realen Entitäten auf vielfältige Weise darstellen, was Handlungen, Eigentumsverhältnisse, Abstammung, Kaufentscheidungen, persönliche Verbindungen, familiäre Bindungen usw. angeht.

Im Folgenden sind einige der häufigsten Bereiche aufgeführt, in denen Graphdatenbanken verwendet werden:

- Wissensgraphen – Mit Wissensgraphen können Sie alle Arten von zusammenhängenden Informationen organisieren und abfragen, um allgemeine Fragen zu beantworten. Mithilfe eines Wissensgraphen können Sie aktuelle Informationen zu Produktkatalogen hinzufügen und vielfältige Informationen modellieren, wie sie beispielsweise in Wikidata enthalten sind.

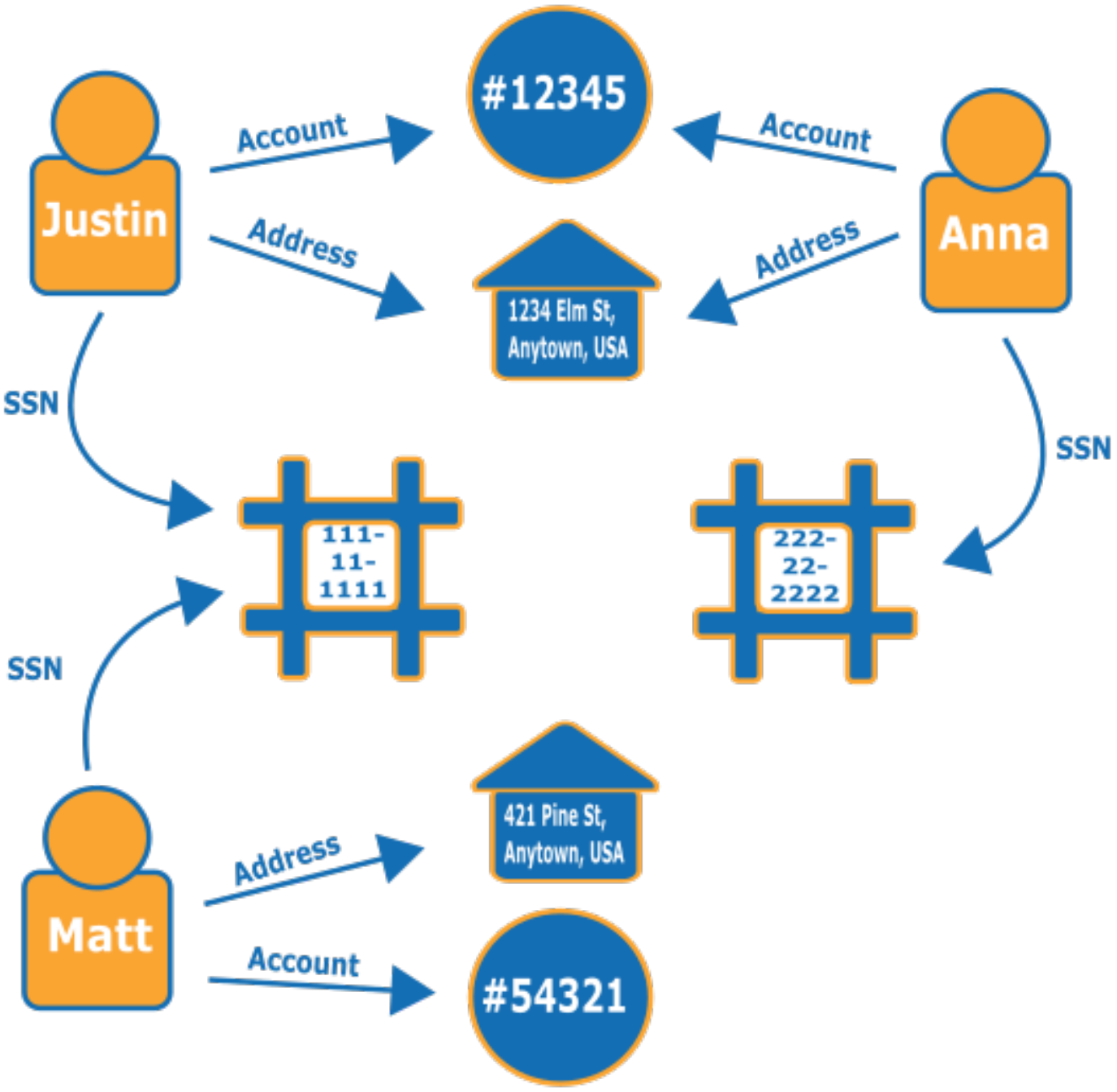
  Weitere Informationen zu Wissensdiagrammen und ihren Einsatzbereichen finden Sie unter Knowledge Graphs on AWS.
- Identitätsgraphen – In einer Graphdatenbank können Sie Beziehungen zwischen Informationskategorien wie Kundeninteressen, Freunden und Kaufhistorie speichern und diese Daten dann abfragen, um personalisierte und relevante Empfehlungen abzugeben.

  Zum Beispiel können Sie eine hochverfügbare Graphdatenbank verwenden, um einem Benutzer Produktempfehlungen basierend darauf zu unterbreiten, welche Produkte von anderen gekauft wurden, die denselben Sport betreiben und eine ähnliche Kaufhistorie aufweisen. Oder Sie können Personen identifizieren, die einen gemeinsamen Freund haben, sich aber noch nicht kennen, und eine Freundschaftsempfehlung abgeben.

  Graphen dieser Art werden als Identitätsgraphen bezeichnet und häufig zur Personalisierung von Interaktionen mit Benutzern verwendet. Weitere Informationen finden Sie unter Identitätsgraphen auf AWS. Um mit der Erstellung Ihres eigenen Identitätsgraphen zu beginnen, können Sie mit dem Beispiel Identitätsgraph unter Verwendung von Amazon Neptune beginnen.
- Betrugsgraphen – Dies ist eine häufige Anwendung von Graphdatenbanken. Sie können Ihnen dabei helfen, Kreditkartenkäufe und Einkaufsstandorte nachzuverfolgen, um ungewöhnliche Nutzungen zu erkennen oder um festzustellen, dass ein Käufer versucht, dieselbe E-Mail-Adresse

und Kreditkarte zu verwenden, die in einem bekannten Betrugsfall verwendet wurden. Damit können Sie nach mehreren Personen suchen, die mit einer persönlichen E-Mail-Adresse verknüpft sind, oder nach mehreren Personen an verschiedenen physischen Standorten, die dieselbe IP-Adresse nutzen.

Sehen Sie sich diesen Graphen an. Er zeigt die Beziehung zwischen drei Personen und ihren identitätsbezogenen Daten. Jede Person hat eine Adresse, ein Bankkonto und eine Sozialversicherungsnummer. Wir sehen aber, dass Matt und Justin dieselbe Sozialversicherungsnummer nutzen. Dies ist ungewöhnlich und weist auf einen möglichen Betrug durch eine dieser Personen hin. Eine Abfrage eines Betrugsdiagramms kann Verbindungen dieser Art aufdecken, so dass sie überprüft werden können.



Weitere Informationen zu Betrugsgrafiken und deren Verwendung finden Sie unter [Betrugsgrafiken auf AWS](#).

- Soziale Netzwerke – Einer der ersten und häufigsten Bereiche, in denen Graphdatenbanken verwendet wurden und werden, sind Anwendungen für soziale Netzwerke.

Angenommen, Sie möchten einen sozialen Feed in eine Website integrieren. Sie können ganz einfach eine Graphdatenbank im Backend verwenden, um Benutzern Ergebnisse zu bieten, die die neuesten Updates von ihren Familien und Freunden, von Personen, deren Updates ihnen „gefallen“, und von Personen, die ihnen nahe stehen, widerspiegeln.

- Wegbeschreibungen – Ein Graph kann dabei helfen, die optimale Route von einem Startpunkt zu einem Ziel zu finden, wobei der aktuelle Verkehr und typische Verkehrsmuster berücksichtigt werden.
- Logistik – Mithilfe von Graphen können Sie ermitteln, wie die verfügbaren Versand- und Vertriebsressourcen am effizientesten genutzt werden können, um Kundenanforderungen zu erfüllen.
- Diagnose – Graphen können komplexe Diagnosestrukturen darstellen, die abgefragt werden können, um die Ursache beobachteter Probleme und Ausfälle zu ermitteln.
- Wissenschaftliche Forschung – Mit einer Graphdatenbank können Sie Anwendungen erstellen, die wissenschaftliche Daten und sogar sensible medizinische Informationen mithilfe von Verschlüsselung im Ruhezustand speichern, und darin navigieren. So können Sie beispielsweise Modelle von Krankheits- und Gen-Interaktionen speichern. Sie können nach Graphmustern in Proteinwegen suchen, um andere Gene zu finden, die mit einer Krankheit assoziiert sein könnten. Sie können chemische Verbindungen als Graph modellieren und Muster in molekularen Strukturen abfragen. Sie können Patientendaten aus Krankenakten in verschiedenen Systemen korrelieren. Sie können veröffentlichte Forschungsergebnisse thematisch organisieren, um schnell relevante Informationen zu finden.
- Regulatorische Regeln – Sie können komplexe regulatorische Anforderungen als Graphen speichern und diese abfragen, um Situationen zu erkennen, in denen sie für Ihren täglichen Geschäftsbetrieb gelten könnten.
- Netzwerktopologie und Ereignisse – Eine Graphdatenbank kann Ihnen bei der Verwaltung und dem Schutz eines IT-Netzwerks helfen. Wenn Sie die Netzwerktopologie als Graph speichern, können Sie auch viele verschiedene Arten von Ereignissen im Netzwerk speichern und verarbeiten. Sie können beispielsweise Fragen beantworten, auf wie vielen Hosts eine bestimmte Anwendung ausgeführt wird. Sie können nach Mustern suchen, die darauf hinweisen könnten, dass ein bestimmter Host durch ein Schadprogramm kompromittiert wurde, und Verbindungsdaten abfragen, anhand derer das Programm bis zu dem ursprünglichen Host, der es heruntergeladen hat, zurückverfolgt werden kann.

Wie fragt man einen Graphen ab?

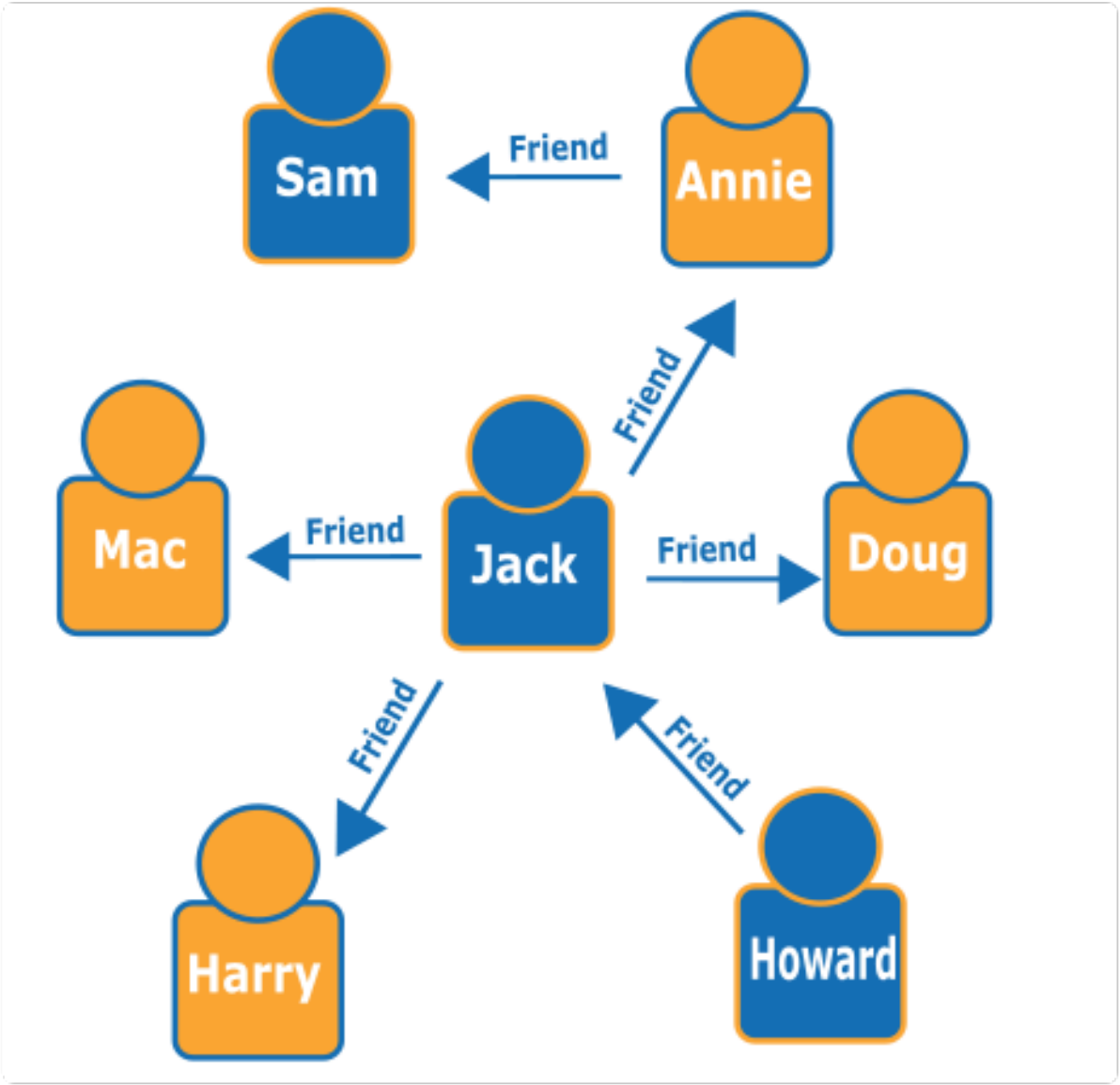
Neptune unterstützt drei spezielle Abfragesprachen, die für die Abfrage von Graphdaten verschiedener Art entwickelt wurden. Sie können diese Sprachen verwenden, um Daten in einer Neptune-Graphdatenbank hinzuzufügen, zu ändern, zu löschen und abzufragen:

- Gremlin ist eine Sprache für Graph-Transversalen für Eigenschaftsgraphen. Eine Abfrage in Gremlin ist eine Transversale, die aus verschiedenen Schritten besteht. Jeder Schritt folgt einem Edge zu einem Knoten. Weitere Informationen finden Sie in der Gremlin-Dokumentation TinkerPop bei Apache.

  Die Neptune-Implementierung von Gremlin weist einige Unterschiede zu anderen Implementierungen auf, insbesondere wenn Sie Gremlin-Groovy (Gremlin-Abfragen, die als serialisierter Text gesendet werden) verwenden. Weitere Informationen finden Sie unter Einhaltung der Gremlin-Standards in Amazon Neptune.
- openCypher – openCypher ist eine deklarative Abfragesprache für Eigenschaftsdiagramme. Ursprünglich von Neo4j entwickelt, wurde sie 2015 als Open-Source-Software veröffentlicht und ist unter einer Apache 2-Open-Source-Lizenz für das openCypher-Projekt verfügbar. Siehe die Referenz zur Cypher Query Language (Version 9) für die Sprachspezifikation sowie den Cypher Style Guide für weitere Informationen.
- SPARQL ist eine deklarative Abfragesprache für RDF-Daten, basierend auf dem durch das World Wide Web Consortium (W3C) standardisierten Graph-Musterabgleich. Eine Beschreibung finden Sie in der Spezifikation SPARQL 1.1 – Übersicht) und SPARQL 1.1 Abfragesprache. Spezifische Informationen zur Neptune-Implementierung von SPARQL finden Sie unter Einhaltung von SPARQL-Standards in Amazon Neptune.

Beispiele für passende Gremlin- und SPARQL-Abfragen

Anhand des folgenden Grafen von Personen (Knoten) und ihrer Beziehungen (Edges), können Sie feststellen, wer die Freunde von Freunden einer bestimmten Person sind, beispielsweise wer die Freunde von Howards Freunden sind.



Wenn Sie sich den Graphen ansehen, erkennen Sie, dass Howard einen Freund namens Jack hat und Jack mit den vier Personen Annie, Harry, Doug und Mac befreundet ist. Dies ist ein einfaches Beispiel für einen einfachen Graphen. Diese Arten von Abfragen können aber an Komplexität, Datensatzgröße und Ergebnismenge zunehmen.

Im Folgenden finden Sie eine Gremlin-Transversal-Abfrage, die die Namen der Freunde von Howards Freunden zurückgibt.

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

Im Folgenden finden Sie eine SPARQL-Abfrage, die die Namen der Freunde von Howards Freunden zurückgibt.

```
prefix : <#>

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

> **Note**
>
> Jedem Teil eines Resource Description Framework (RDF)-Tripels ist ein URI zugeordnet. In diesem Beispiel ist das URI-Präfix absichtlich kurz.

Nehmen Sie an einem Online-Kurs zur Nutzung von Amazon Neptune teil

Wenn Sie gerne mit Videos lernen, AWS bietet Ihnen die Online-Tech-Talks Online-Kurse an, die Ihnen den Einstieg erleichtern AWS . Ein Kurs, der in Graphdatenbanken einführt, ist:

> Einführung zu Graphdatenbanken, detaillierte Informationen und Demo mit Amazon Neptune.

Eingehendere Informationen zur Graph-Referenzarchitektur

Wenn Sie darüber nachdenken, welche Probleme eine Graphdatenbank für Sie lösen könnte und wie Sie sie angehen können, ist das Neptune Graph Reference GitHub Architectures Project einer der besten Ausgangspunkte.

Dort finden Sie detaillierte Beschreibungen der Graph-Workload-Typen sowie drei Abschnitte, die Ihnen beim Entwerfen einer effektiven Graphdatenbank helfen:

- Datenmodelle und Abfragesprachen – Dieser Abschnitt führt Sie durch die Unterschiede zwischen Gremlin und SPARQL und zeigt Ihnen, wie Sie zwischen ihnen wählen können.

- Graphdatenmodellierung – Dies ist eine gründliche Darlegung dazu, wie Entscheidungen bei der Graphdatenmodellierung getroffen werden können, einschließlich detaillierter Anleitungen zur Modellierung von Eigenschaftsgraphen mit Gremlin und zur RDF-Modellierung mit SPARQL.
- Konvertierung anderer Datenmodelle in ein Graphmodell – Hier erfahren Sie, wie Sie ein relationales Datenmodell in ein Graphmodell übersetzen können.

Es gibt auch drei Abschnitte, die Sie durch die spezifischen Schritte zur Verwendung von Neptune führen:

- Herstellen einer Verbindung zu Amazon Neptune von Clients außerhalb der Neptune-VPC aus – In diesem Abschnitt finden Sie verschiedene Optionen für die Verbindung mit Neptune von außerhalb der VPC aus, in der sich Ihr DB-Cluster befindet.
- Über AWS Lambda Functions auf Amazon Neptune zugreifen — Hier erfahren Sie, wie Sie über Lambda-Funktionen eine zuverlässige Verbindung zu Neptune herstellen können.
- Schreiben zu Amazon Neptune von einem Amazon Kinesis Data Stream aus – Dieser Abschnitt kann Ihnen helfen, Szenarien mit hohem Schreibdurchsatz mit Neptune zu bewältigen.

Schneller Einstieg in die Verwendung AWS CloudShell

Diese Anleitung hilft Ihnen beim schnellen Einstieg, indem Sie direkt von der AWS Management Console aus eine Verbindung AWS CloudShellzu Ihrem Neptune-Cluster herstellen und ihn abfragen können.

Zeit bis zur Fertigstellung: Ungefähr 2 Minuten

Kosten: CloudShell ist ohne zusätzliche Kosten erhältlich. Das Erstellen eines Neptun-Clusters ist mit Gebühren verbunden. Preisinformationen finden Sie unter Amazon Neptune Neptune-Preise.

Voraussetzungen

- Ein Konto AWS . Falls Sie noch keines haben, finden Sie weitere Informationen unter AWS Konto einrichten.
- Ein Neptun-Cluster mit dem Status Verfügbar. Falls Sie noch keinen haben, finden Sie weitere Informationen unter Starten eines Neptune-DB-Clusters mit dem AWS-Managementkonsole So erstellen Sie einen.

Schritt 1: Connect zu Ihrem Neptun-Cluster her

Stellen Sie mithilfe von eine Connect zu Ihrem Neptun-Cluster her. AWS CloudShell CloudShellstellt automatisch eine Shell-Umgebung bereit, die mit der VPC Ihres Clusters verbunden ist. Sie müssen keine Sicherheitsgruppen konfigurieren oder Notebooks einrichten, um mit der Datenbank zu interagieren.

Um eine Verbindung zu Ihrem Cluster herzustellen

1. Suchen Sie in der Neptune-Konsole unter Clusters den Cluster, den Sie erstellt haben.
2. Aktivieren Sie das Kontrollkästchen neben Ihrem Cluster.
3. Wählen Sie Mit Cluster verbinden aus.

   > **Note**
   >
   > Diese Schaltfläche wird aktiv, nachdem Sie das Kontrollkästchen neben Ihrem Cluster aktiviert haben und der Clusterstatus als Verfügbar angezeigt wird.

   Der Bildschirm CloudShell mit dem Befehl Ausführen wird angezeigt.
4. Wählen Sie Ausführen, um eine Verbindung zum Cluster herzustellen. Die Bereitstellung der VPC-Umgebung dauert einige Sekunden.

Nachdem Sie eine Verbindung hergestellt haben, erscheint die `graphsh` Konsole mit einer interaktiven Shell zum Ausführen von Graphabfragen.

Schritt 2: Daten einfügen und abfragen

Nachdem Sie mit Ihrem Cluster verbunden sind, führen Sie einige Abfragen aus, um sich mit Amazon Neptune vertraut zu machen. In den folgenden Beispielen wird eine der von Neptune unterstützten Abfragesprachen verwendetopenCypher.

Um Knoten und Beziehungen einzufügen

Führen Sie die folgende Abfrage aus, um zwei Personenknoten und eine Beziehung zu erstellen:

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

Die Ausgabe sieht folgendermaßen oder ähnlich aus:

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

Um Knoten zu finden

Führen Sie die folgende Abfrage aus, um alle Personennamen in alphabetischer Reihenfolge zurückzugeben:

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

Die Ausgabe sieht folgendermaßen oder ähnlich aus:

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

Nächste Schritte

Sie haben diesen Schnellstart abgeschlossen. Erkunden Sie die folgenden Funktionen, um weiter zu lernen:

- Übersicht über Amazon-Neptune-Features— Erfahren Sie mehr über die wichtigsten Funktionen von Amazon Neptune.
- Verwendung von Amazon Neptune Serverless— Skalieren Sie Ihre Cluster-Kapazität automatisch auf der Grundlage der Arbeitslast.
- Verwalten Ihrer Amazon-Neptune-Datenbank— Verwalten Sie Ihre Cluster, Instanzen und Konfigurationen.

Um laufende Gebühren zu vermeiden, löschen Sie den Neptun-Cluster, wenn Sie ihn nur für diesen Schnellstart erstellt haben. Detaillierte Anweisungen finden Sie unter Verwalten Ihrer Amazon-Neptune-Datenbank.

Amazon Neptune mit Grafiknotizbüchern verwenden

Um mit Neptun-Diagrammen zu arbeiten, können Sie ein Neptun-Graph-Notizbuch verwenden oder mithilfe einer Vorlage eine neue Neptun-Datenbank erstellen. CloudFormation

Ganz gleich, ob Sie mit Diagrammen noch nicht vertraut sind und lernen und experimentieren möchten, oder ob Sie Erfahrung haben und Ihre Abfragen verfeinern möchten, die Neptune Workbench bietet eine interaktive Entwicklungsumgebung (IDE), mit der Sie Ihre Produktivität beim Erstellen von Diagrammanwendungen steigern können. Die Workbench bietet eine benutzerfreundliche Oberfläche für die Interaktion mit Ihrer Neptune-Datenbank, das Schreiben von Abfragen und die Visualisierung Ihrer Daten.

Indem Sie die CloudFormation Vorlage für die Einrichtung Ihrer Neptune-Datenbank und die Workbench für die Entwicklung Ihrer Diagrammanwendungen verwenden, können Sie schnell und effizient mit Neptune beginnen, ohne dass zusätzliche Tools erforderlich sind. Auf diese Weise können Sie sich auf die Erstellung Ihrer Anwendungen konzentrieren, anstatt die zugrunde liegende Infrastruktur einzurichten.

> **Note**
>
> Neptune-Notizbücher, die über Amazon SageMaker AI verwaltet werden, sind derzeit in der Region Asien-Pazifik (Malaysia) (ap-Southeast-5) nicht verfügbar. Sie können Neptune-Notebooks jedoch weiterhin über alternative, nicht verwaltete Optionen bereitstellen. Informationen zur manuellen Bereitstellung von Manuelles Einrichten Neptune Neptune-Notebooks Notebooks finden Sie unter.

Neptune stellt Jupyter und JupyterLabNotebooks im Open-Source-Neptune Graph Notebook-Projekt auf und in der Neptune Workbench GitHub zur Verfügung. Diese Notebooks bieten Tutorials mit Beispielanwendungen und Codefragmente in einer interaktiven Codierungsumgebung, in der Sie mehr über Graphtechnologie und Neptune erfahren können. Sie können sie verwenden, um das Einrichten, Konfigurieren, Auffüllen und Abfragen von Graphen mithilfe verschiedener Abfragesprachen, verschiedener Datensätze und sogar verschiedener Datenbanken im Backend durchzugehen.

Sie können diese Notebooks auf verschiedene Arten hosten:

- Mit der Neptune Workbench können Sie Jupyter-Notebooks in einer vollständig verwalteten Umgebung ausführen, die in Amazon SageMaker AI gehostet wird, und lädt automatisch die

neueste Version des Neptune Graph Notebook-Projekts für Sie. Es ist einfach, die Workbench in der Neptune-Konsole einzurichten, wenn Sie eine neue Neptune-Datenbank erstellen.

> **Note**
>
> Wenn Sie eine Neptune-Notebook-Instance erstellen, stehen Ihnen zwei Optionen für den Netzwerkzugriff zur Verfügung: Direkter Zugriff über Amazon SageMaker AI (Standard) und Zugriff über eine VPC. Bei beiden Optionen benötigt das Notebook Zugriff auf das Internet, um Paketabhängigkeiten für die Installation der Neptune-Workbench abzurufen. Mangelnder Internetzugang führt dazu, dass die Erstellung einer Neptune-Notebook-Instanz fehlschlägt.

- Sie können Jupyter auch lokal installieren. Auf diese Weise können Sie die Notebooks von Ihrem Laptop aus ausführen, der entweder mit Neptune oder mit einer lokalen Instance einer der Open-Source-Graphdatenbanken verbunden ist. Im letzteren Fall können Sie so viel mit der Graph-Technologie experimentieren, wie Sie möchten, bevor Sie auch nur einen Cent ausgeben. Wenn Sie dann bereit sind, können Sie problemlos zur verwalteten Produktionsumgebung wechseln, die Neptune bietet.

Verwenden der Neptune-Workbench zum Hosten von Neptune-Notebooks

Neptune bietet die Instance-Typen T3 und T4g, mit denen Sie für weniger als 0,10 USD pro Stunde beginnen können. Workbench-Ressourcen werden Ihnen über Amazon SageMaker AI separat von Ihrer Neptune-Abrechnung in Rechnung gestellt. Weitere Informationen finden Sie auf der Seite mit Neptune-Preisen. Jupyter und JupyterLab Notebooks, die auf der Neptune Workbench erstellt wurden, verwenden alle eine Amazon Linux 2- und 4-Umgebung. JupyterLab Weitere Informationen zur JupyterLab Notebookunterstützung finden Sie in der Amazon SageMaker AI-Dokumentation.

Sie können einen Jupyter oder ein JupyterLab Notizbuch mit der Neptune-Workbench auf zwei Arten erstellen: AWS-Managementkonsole

- Verwenden Sie das Menü Notebook-Konfiguration, wenn Sie einen neuen Neptune-DB-Cluster erstellen. Befolgen Sie dafür die unter Starten eines Neptune-DB-Clusters mit dem AWS-Managementkonsole beschriebenen Schritte.
- Verwenden Sie das Notebook-Menü im linken Navigationsbereich, wenn Ihr DB-Cluster bereits erstellt wurde. Führen Sie dazu die folgenden Schritte aus.

So erstellen Sie einen Jupyter oder ein Notizbuch mithilfe des Notebook-Menüs JupyterLab

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die Amazon Neptune Neptune-Konsole zu Hausehttps://console.aws.amazon.com/neptune/.
2. Wählen Sie im Navigationsbereich auf der linken Seite Notebooks (Notizbücher) aus.
3. Klicken Sie auf Create Notebook (Notebook erstellen).
4. Wählen Sie Database als Neptune-Dienst.
5. Wählen Sie in der Liste Cluster Ihren Neptune-DB-Cluster aus. Wenn Sie noch keinen DB-Cluster haben, wählen Sie Create cluster (Cluster erstellen) aus, um einen zu erstellen.
6. Wählen Sie einen Notebook-Instance-Typ aus.
7. Geben Sie einen Namen und optional eine Beschreibung für Ihr Notebook ein.
8. Sofern Sie nicht bereits eine AWS Identity and Access Management (IAM-) Rolle für Ihre Notebooks erstellt haben, wählen Sie Create an IAM-Rolle aus und geben Sie einen IAM-Rollennamen ein.

   > **Note**
   >
   > Wenn Sie sich dafür entscheiden, eine IAM-Rolle wiederzuverwenden, die für ein früheres Notebook erstellt wurde, muss die Rollenrichtlinie die richtigen Berechtigungen für den Zugriff auf den Neptune-DB-Cluster enthalten, den Sie verwenden. Sie können dies sicherstellen, indem Sie überprüfen, ob die Komponenten im Ressourcen-ARN unter der `neptune-db:*`-Aktion mit diesem Cluster übereinstimmen. Nicht korrekt konfigurierte Berechtigungen führen zu Verbindungsfehlern, wenn Sie versuchen, Notebook-Magic-Befehle auszuführen.

9. Klicken Sie auf Create Notebook (Notebook erstellen). Der Erstellungsvorgang kann 5 bis 10 Minuten dauern, bis alles bereit ist.
10. Nachdem Ihr Notizbuch erstellt wurde, wählen Sie es aus und wählen Sie dann Jupyter öffnen oder Öffnen. JupyterLab

Die Konsole kann eine AWS Identity and Access Management (IAM)-Rolle für Ihre Notebooks erstellen oder Sie können selbst eine Rolle erstellen. Die Richtlinie für diese Rolle sollte Folgendes umfassen:

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/*"
      ]
    }
  ]
}
```

Beachten Sie, dass die zweite Anweisung in der obigen Richtlinie eine oder mehrere Neptune-Cluster-Ressourcen auflistet. IDs

Außerdem sollte die Rolle die folgende Vertrauensstellung aufbauen:

JSON

```
{
  "Version":"2012-10-17",
```

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

Auch hier kann es 5 bis 10 Minuten dauern, bis alles einsatzbereit ist.

Sie können Ihr neues Notebook so konfigurieren, dass es mit Neptune ML funktioniert, wie unter Manuelles Konfigurieren eines Neptune-Notebooks für Neptune ML erläutert.

Verwenden von Python, um ein generisches SageMaker KI-Notizbuch mit Neptune zu verbinden

Das Anschließen eines Notebooks an Neptune ist einfach, wenn Sie Neptune Magics installiert haben. Es ist jedoch auch möglich, ein SageMaker KI-Notebook mithilfe von Python mit Neptune zu verbinden, auch wenn Sie kein Neptune-Notebook verwenden.

Schritte, die unternommen werden müssen, um in einer SageMaker KI-Notebookzelle eine Verbindung zu Neptune herzustellen

1. Installieren des Gremlin-Python-Clients:

   ```
   !pip install gremlinpython
   ```

   Neptune-Notebooks installieren den Gremlin Python-Client für Sie, sodass dieser Schritt nur erforderlich ist, wenn Sie ein einfaches KI-Notizbuch verwenden. SageMaker
2. Schreiben Sie Code wie den folgenden, um eine Verbindung herzustellen und eine Gremlin-Abfrage auszuführen:

   ```
   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

Falls Sie eine Version des Gremlin Python-Clients verwenden sollten, die älter als 3.5.0 ist, diese Zeile:

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

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

Wäre einfach:

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

Starten von Graph-Notebook als Amazon Neptune Workbench über CloudFormation

Sie können Amazon Neptune Workbench-Notebooks mithilfe von bereitstellen. CloudFormation Die CloudFormation Vorlage stellt Neptune Workbench-Notizbücher als Ressourcen bereit und beinhaltet die Basisnotizbücher für „Erste Schritte“. Mit der Workbench können Sie mit Ihrem Amazon Neptune Database-Cluster mithilfe von Jupyter-Notebooks arbeiten, die von Amazon gehostet werden. SageMaker Workbench-Ressourcen werden Ihnen über Amazon separat von Ihrer SageMaker Neptune-Abrechnung in Rechnung gestellt.

Die CloudFormation Vorlage und detaillierte Anweisungen, einschließlich der Anforderungen und Berechtigungen für die IAM-Rollen, sind im Graph-Notebook-Repository verfügbar. GitHub

CloudWatch Logs auf Neptune Notebooks aktivieren

CloudWatch Logs sind jetzt standardmäßig für Neptune Notebooks aktiviert. Wenn Sie ein älteres Notizbuch haben, das keine CloudWatch Protokolle erzeugt, gehen Sie wie folgt vor, um sie manuell zu aktivieren:

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die SageMaker AI-Konsole.
2. Wählen Sie im Navigationsbereich auf der linken Seite Notebook und dann Notebook-Instances aus. Suchen Sie nach dem Namen des Neptune-Notebooks, für das Sie Protokolle aktivieren möchten.
3. Gehen Sie zur Detailseite, indem Sie den Namen dieser Notebook-Instance auswählen.
4. Wenn die Notebook-Instance ausgeführt wird, klicken Sie oben rechts auf der Notebook-Detailseite auf die Schaltfläche Stopp.
5. Unter Berechtigungen und Verschlüsselung gibt es ein Feld für den ARD der IAM-Rolle. Wählen Sie den Link in diesem Feld aus, um zu der IAM-Rolle zu gelangen, mit der diese Notebook-Instance ausgeführt wird.
6. Erstellen Sie die folgende Richtlinie:

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. Speichern Sie diese neue Richtlinie und fügen Sie sie an die IAM-Rolle aus Schritt 4 an.
8. Klicken Sie oben rechts auf der Detailseite der SageMaker AI-Notebook-Instanz auf Start.
9. Sobald Protokolle erstellt werden, sollte der Link Protokolle anzeigen unterhalb des Felds Lifecycle-Konfiguration unten links auf der Detailseite im Abschnitt Notebook-Instance-Einstellungen angezeigt werden.

Wenn ein Notebook nicht gestartet werden kann, erscheint auf der Seite mit den Notebook-Details auf der SageMaker AI-Konsole eine Meldung, dass der Start der Notebook-Instanz über 5 Minuten gedauert hat. CloudWatch Protokolle, die für dieses Problem relevant sind, finden Sie unter diesem Namen:

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

Einrichten von Graph-Notebooks auf Ihrem lokalen Computer

Das Graph-Notebook-Projekt enthält Anweisungen zum Einrichten von Neptune-Notebooks auf Ihrem lokalen Computer:

- Voraussetzungen
- Jupyter und Installation JupyterLab
- Verbinden mit einer Graphdatenbank.

Sie können Ihre lokalen Notebooks entweder mit einem Neptune-DB-Cluster oder mit einer lokalen oder Remote-Instance einer Open-Source-Graphdatenbank verbinden.

Verwenden von Neptune-Notebooks mit Neptune-Clustern

Wenn Sie eine Verbindung zu einem Neptune-Cluster am Backend herstellen, möchten Sie die Notebooks möglicherweise in Amazon SageMaker AI ausführen. Die Verbindung zu Neptune über SageMaker KI kann bequemer sein als über eine lokale Installation der Notebooks, und Sie können so einfacher mit Neptune ML arbeiten.

Anweisungen zum Einrichten von Notebooks in SageMaker AI finden Sie unter Graph-Notebook mit Amazon starten. SageMaker

Anweisungen zum Einrichten und Konfigurieren von Neptune selbst finden Sie unter Amazon Neptune einrichten.

Sie können auch eine lokale Installation der Neptune-Notebooks mit einem Neptune-DB-Cluster verbinden. Dies kann etwas komplizierter sein, da Amazon-Neptune-DB-Cluster nur in einer Amazon Virtual Private Cloud (VPC) erstellt werden können, die ihrer Konstruktion nach von der Außenwelt isoliert ist. Es gibt eine Reihe von Möglichkeiten, von außen eine Verbindung zu einer VPC herzustellen. Eine Möglichkeit besteht darin, einen Load Balancer zu verwenden. Eine andere Möglichkeit ist die Verwendung von VPC Peering (siehe Handbuch zu Amazon Virtual Private Cloud Peering).

Für die meisten Benutzer ist es jedoch am bequemsten, eine Verbindung herzustellen, indem sie einen Amazon-EC2-Proxyserver innerhalb der VPC einrichten und dann SSH-Tunneling (auch „Portweiterleitung“ genannt) zu verwenden, um eine Verbindung dazu herzustellen. Anweisungen zur Einrichtung finden Sie unter Graph-Notebook lokal mit Amazon Neptune verbinden im `additional-databases/neptune` Ordner des GitHub Graph-Notebook-Projekts.

Verwendung von Neptune-Notebooks mit Open-Source-Graphdatenbanken

Um kostenlos mit der Graphtechnologie zu beginnen, können Sie auch Neptune-Notebooks mit verschiedenen Open-Source-Datenbanken im Backend verwenden. Beispiele hierfür sind der TinkerPop Gremlin-Server und die Blazegraph-Datenbank.

Gehen Sie folgendermaßen vor, um Gremlin Server als Back-End-Datenbank zu verwenden:

- Das Graph-Notizbuch wird mit einem Gremlin Server-Ordner verbunden. GitHub
- Der Gremlin-Konfigurationsordner von graph-notebook. GitHub

Gehen Sie wie folgt vor, um eine lokale Blazegraph-Instanz als Backend-Datenbank zu verwenden:

- Lesen Sie die Blazegraph-Schnellstartanweisungen, um die grundlegende Einrichtung und Konfiguration zu verstehen, die für den Betrieb einer Blazegraph-Instanz erforderlich sind.
- Greifen Sie auf den GitHub Blazegraph-Konfigurationsordner von graph-notebook zu, der die erforderlichen Dateien und Anweisungen zum Einrichten einer lokalen Blazegraph-Instanz enthält.
- Navigieren Sie innerhalb des GitHub Repositorys zum Verzeichnis „Blazegraph“ und folgen Sie den Anweisungen, um Ihre lokale Blazegraph-Instanz einzurichten. Dies beinhaltet Schritte zum Herunterladen der Blazegraph-Software, zum Konfigurieren der erforderlichen Dateien und zum Starten des Blazegraph-Servers.

Sobald Sie eine lokale Blazegraph-Instanz ausgeführt haben, können Sie sie als Backend-Datenbank für Ihre graphbasierten Daten und Abfragen in Ihre Anwendung integrieren. In der Dokumentation und im Beispielcode im Graph-Notebook-Repository erfahren Sie, wie Sie Ihre Anwendung mit der Blazegraph-Instanz verbinden.

Migration von Neptune-Notebooks auf 4.x JupyterLab

In diesem Abschnitt werden verschiedene Ansätze für die Migration Ihrer Neptune-Notebooks auf Amazon Linux-Umgebungen der Version JupyterLab 4.x und neuer beschrieben. Ausführliche Informationen zur JupyterLab Versionierung finden Sie unter Amazon SageMaker AI JupyterLab Versioning.

Migrationsansätze

Neuinstallation

Wenn Sie vorhandene Workspace-Dateien oder -Konfigurationen nicht beibehalten müssen, können Sie:

1. Erstellen Sie eine neue Notebook-Instanz, auf der JupyterLab 4.x (notebook-al2-v3) ausgeführt wird
2. Stellen Sie sicher, dass das neue Setup wie erwartet funktioniert
3. Stoppen und löschen Sie Ihre alte Notebook-Instanz

Migration von Dateiübertragungen

Diese Methode verwendet Ihr lokales System oder Amazon S3 als Zwischenspeicher.

Am besten geeignet für

- Direkter Internetzugang über die Amazon SageMaker AI-Netzwerkkonfiguration.
- Ein moderates Datenvolumen für die Migration
- Bestimmte Dateien, die beibehalten werden sollen, und nicht ganze Workspace-Konfigurationen.

Methode 1: Verwenden der JupyterLab Benutzeroberfläche

Am besten geeignet für

- Kleine Anzahl von Dateien
- Selektive Dateimigration
- Bevorzugen Sie einfache drag-and-drop Operationen

Schritte

1. Laden Sie Dateien von der JupyterLab Quellinstanz herunter:
   - Navigieren Sie zu den Dateien, die Sie auf die neue Instanz migrieren möchten, und wählen Sie sie aus JupyterLab

- Rechtsklicken Sie und wählen Sie Herunterladen

2. Auf eine neue JupyterLab Instanz hochladen:
   - Verwenden Sie die Upload-Schaltfläche in JupyterLab und wählen Sie alle Dateien aus, die Sie in die neue Instanz kopieren möchten
   - (Oder) Dateien direkt per Drag-and-Drop ziehen

Methode 2: Amazon S3 verwenden

Am besten geeignet für

- Große Anzahl von Dateien
- Beibehaltung Ihrer Ordnerstrukturen
- Massenmigrationen

Voraussetzungen

Stellen Sie sicher, dass die dem Notizbuch zugeordnete Rolle über die entsprechenden Berechtigungen verfügt, um den Amazon S3 S3-Bucket hochzuladen und darauf zuzugreifen:

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

> **Note**
>
> AWS CLI sollte auf SageMaker AI-Notebooks vorinstalliert sein.

Schritte

1. Öffnen Sie ein Terminal in JupyterLab oder geben Sie die Terminalbefehle in eine Notebookzelle mit ! Präfix ein.
2. Kopieren Sie Dateien mit den CLI-Befehlen Amazon S3 cp oder Amazon S3 sync von Ihrer alten JupyterLab Instance nach S3:

```
# 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. Kopieren Sie Dateien von S3 auf Ihre neue 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**
>
> Wird `sync` zur Pflege von Ordnerstrukturen und für inkrementelle Updates sowie `cp` für einmalige Übertragungen verwendet.

Amazon EFS-Migration

Am besten geeignet für

- Netzwerkkonfiguration nur für VPC-Geräte
- Große Datenmengen

Schritte

Folgen Sie dem Blog Ein EFS-Dateisystem in ein Amazon SageMaker AI-Notizbuch einbinden, um ein Amazon EFS-Dateisystem mit Ihren Notebook-Instances zu verwenden.

Darüber hinaus gibt es noch einige weitere Schritte, die speziell für die Migration von Neptune-Notebooks in die neue Umgebung gelten:

1. Wählen Sie bei der Erstellung eines Neptune-Notebooks in der Konsole unter Lebenszykluskonfiguration die Option Neue Lebenszykluskonfiguration erstellen aus

2. Fügen Sie in der Template-Lifecycle-Konfiguration Ihren Amazon EFS-Mount-Befehl (`sudo mount -t nfs ...`) nach dem Skript install.sh an

Dadurch wird sichergestellt, dass Ihr Amazon EFS-Dateisystem bei jedem Start oder Neustart Ihrer Notebook-Instance automatisch bereitgestellt wird. Informationen zur Behebung von Bereitstellungsproblemen finden Sie im Amazon EFS-Problembehandlungsdokument.

Vorteile

- Nahtloser instanzübergreifender Zugriff auf Dateien
- Direkter Dateizugriff ohne Zwischenübertragungen
- Effizienter Umgang mit großen Datensätzen

Amazon EBS-Volumenmigration

Am besten, wenn Sie etwas aufbewahren müssen

- Vollständige Workspace-Konfigurationen
- Versteckte Dateien
- Systemeinstellungen
- Beibehaltung vollständiger Workspace-Konfigurationen, versteckter Dateien und Systemeinstellungen

Folgen Sie dem AWS SageMaker KI-Migrationsleitfaden für Amazon EBS-Volumes, um Dateien von dem Amazon EBS-Volume zu übertragen, das den Notebook-Instances zugeordnet ist.

Darüber hinaus gibt es noch einige weitere Schritte, die speziell für die Migration von Neptune-Notebooks in die neue Umgebung gelten.

Neptune-spezifische Voraussetzungen

Fügen Sie der IAM-Rolle des Quell-Neptune-Notebooks alle folgenden Berechtigungen hinzu:

```
{
  "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": [
    "*"
  ]
}
```

Stellen Sie sicher, dass Sie den korrekten ARN für den S3-Bucket angeben, den Sie für die Sicherung verwenden werden.

Neptune-spezifische Lebenszyklus-Konfiguration

Bei der Erstellung des zweiten Lifecycle-Konfigurationsskripts für die Wiederherstellung des Backups (von `on-create.sh`), wie im Blog-Beitrag beschrieben, muss der Lifecycle-Name dem `aws-neptune-*`-Format entsprechen, wie `aws-neptune-sync-from-s3`. Dadurch wird sichergestellt, dass das LCC während der Notebook-Erstellung in der Neptune-Konsole ausgewählt werden kann.

Neptune-spezifische Synchronisation von einem Snapshot zu einer neuen Instance

In den Schritten, die im Blog-Beitrag zur Synchronisation von einem Snapshot zu einer neuen Instance beschrieben werden, sind hier die für Neptune spezifischen Änderungen aufgeführt:

- Wählen Sie in Schritt 4 notebook-al2-v3.
- Verwenden Sie in Schritt 5 die IAM-Rolle aus dem Neptune-Quell-Notebook erneut.
- Zwischen den Schritten 7 und 8:

- Geben Sie in den Notebook-Instance-Einstellungen einen Namen ein, der das `aws-neptune-*`-Format verwendet.
- Öffnen Sie das Einstellungsakkordeon Netzwerk und wählen Sie dieselbe VPC-, Subnetz- und Sicherheitsgruppe wie im Quell-Notebook aus.

Für Neptune spezifische Schritte nach der Erstellung des neuen Notebooks

1. Wählen Sie die Schaltfläche Jupyter öffnen für das Notebook. Fahren Sie nach der Anzeige der `SYNC_COMPLETE`-Datei im Hauptverzeichnis mit dem nächsten Schritt fort.
2. Gehen Sie in der AI-Konsole zur Notebook-Instanzseite. SageMaker
3. Halten Sie das Notebook an.
4. Wählen Sie Bearbeiten aus.
5. Bearbeiten Sie in den Notebook-Instance-Einstellungen das Feld Lifecycle-Konfiguration, indem Sie den ursprünglichen Lebenszyklus des Neptune-Quell-Notebooks auswählen. Beachten Sie, dass dies nicht der EBS-Backup-Lebenszyklus ist.
6. Wählen Sie Notebook-Einstellungen aktualisieren aus.
7. Starten Sie das Notebook erneut.

Mit den hier beschriebenen Änderungen an den im Blogbeitrag beschriebenen Schritten sollten Ihre Graph-Notebooks jetzt auf eine neue Neptune-Notebook-Instance migriert werden, die die Amazon Linux 2- und JupyterLab 4-Umgebung verwendet. Sie werden für den Zugriff und die Verwaltung auf der Neptun-Seite im angezeigt AWS-Managementkonsole, und Sie können Ihre Arbeit jetzt dort fortsetzen, wo Sie aufgehört haben, indem Sie entweder Jupyter öffnen oder Öffnen auswählen. JupyterLab

Erstellen Sie ein Neptune-Notizbuch in Amazon AI-Instances SageMaker

1. Öffnen Sie die Amazon SageMaker AI-Konsole unter https://console.aws.amazon.com/sagemaker/.
2. Erweitern Sie im Navigationsbereich Notebook und wählen Sie dann Notebook-Instances aus.
3. Wählen Sie Create notebook instance (Notebook-Instance erstellen) aus.
4. Geben Sie in den Notebook-Instanzeinstellungen unter Notebook-Instanzname dem Notizbuch einen Namen mit dem Präfix `aws-neptune-` (z. B.`aws-neptune-my-test-notebook`).

5. Wählen Sie unter Plattform-ID Amazon Linux 2, JupyterLab 4 aus.
6. Wählen Sie Zusätzliche Konfiguration aus.
7. Wählen Sie unter Lebenszykluskonfiguration die Option Neue Lebenszykluskonfiguration erstellen aus.
8. Geben Sie unter Konfiguration unter Name den Namen der Notebook-Instanz aus Schritt 4 ein.
9. Ersetzen Sie unter Skripts unter Notebook starten das vorhandene Skript durch dieses:

```
#!/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. Wählen Sie Create configuration (Konfiguration erstellen) aus.
11. Wählen Sie unter Berechtigungen und Verschlüsselung unter IAM-Rolle die Rolle aus, die Sie oben erstellt haben.
12. Wenn Sie im Netzwerk einen privaten Graph-Endpunkt verwenden, gehen Sie wie folgt vor:

    a. Wählen Sie unter VPC die VPC aus, in der sich das Neptune Analytics-Diagramm befindet.

    b. Wählen Sie unter Subnetz ein Subnetz aus, das dem Neptune Analytics-Diagramm zugeordnet ist.

    c. Wählen Sie unter Sicherheitsgruppe (n) alle Sicherheitsgruppen aus, die dem Neptune Analytics-Diagramm zugeordnet sind.

13. Wählen Sie Create notebook instance (Notebook-Instance erstellen) aus.
14. Wenn Ihr neues Notizbuch nach 5 oder 10 Minuten den `Ready` Status erreicht hat, wählen Sie es aus. Wählen Sie „Jupyter öffnen“ oder „Öffnen“. JupyterLab

Manuelles Einrichten Neptune Neptune-Notebooks

Sie können auch die für graph-notebook und graph-explorer verfügbaren AWS Open-Source-Pakete verwenden, um eine Neptune-Notebook-Umgebung einzurichten. Es gibt zwar mehrere Möglichkeiten, ein Notebook mithilfe von Open-Source-Paketen einzurichten, aber der empfohlene Ansatz ist:

- Richten Sie Graph-Notebook auf Ihrem lokalen Computer und einen Amazon EC2 EC2-SSH-Tunnel ein, der Ihren lokalen Computer mit einem Neptune-Cluster auf derselben VPC wie die Amazon EC2 EC2-Instance verbindet.
- Richten Sie Graph-Explorer auf einer Amazon EC2 EC2-Instance in Ihrer VPC ein.

Detaillierte Anweisungen zur Einrichtung der Open-Source-Pakete Graph-Notebooks und Graph-Explorer finden Sie in der folgenden offiziellen Dokumentation und den folgenden Repositorys: 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

Security configuration (Sicherheitskonfiguration)

Achten Sie bei der Einrichtung Ihrer Umgebung darauf, dass die folgenden Sicherheitskonfigurationen entsprechend eingestellt sind:

- Neptune-Cluster-Sicherheitsgruppe — Erlaubt eingehenden TCP-Verkehr auf Port 8182 von der Sicherheitsgruppe Ihrer Amazon EC2 EC2-Instance.
- Amazon EC2 EC2-Instance-Sicherheitsgruppe — Konfigurieren Sie HTTPS-Regeln für eingehenden Datenverkehr, um den Graph Explorer-Zugriff zu ermöglichen.

Wir empfehlen, eine einzige Sicherheitsgruppe sowohl für Ihren Neptune-Cluster als auch für Ihre Amazon EC2 EC2-Instance zu verwenden. Dies vereinfacht die Konfiguration und reduziert potenzielle Verbindungsprobleme. Denken Sie jedoch daran, eine Regel zur Selbstreferenzierung hinzuzufügen, um die Kommunikation zwischen Ressourcen innerhalb der Gruppe zu ermöglichen.

Verwendung von Magie in Amazon Neptune Neptune-Notizbüchern

Die Neptune-Workbench stellt in den Notebooks mehrere so genannte Magics-Befehle bereit, die viel Zeit und Aufwand sparen. Sie können in zwei Kategorien eingeteilt werden: Zeilen-Magics und Zellen-Magics.

Zeilen-Magics sind Befehle, denen ein einzelnes Prozentzeichen (%) vorangestellt ist. Sie akzeptieren nur Zeileneingaben, keine Eingaben vom Rest der Zelle. Die Neptune-Workbench stellt die folgenden Zeilen-Magics bereit:

- %seed
- %load
- %load_ids
- %load_status
- %cancel_load
- %status
- %gremlin_status
- %opencypher_status oder %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
- Verteilung in%

Zellen-Magics sind Befehle, denen zwei (%%) vorangestellt sind. Sie verwenden den Zelleninhalt als Eingabe, können jedoch auch Zeileninhalte als Eingabe verwenden. Die Neptune-Workbench stellt die folgenden Zellen-Magics bereit:

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

Es gibt auch zwei Magics, eine Zeilen-Magics und eine Zellen-Magics für Neptune Machine Learning.

- %neptune_ml
- %%neptune_ml

> **Note**
>
> Beim Arbeiten mit Neptune-Magics können Sie in der Regel über den Parameter `--help` oder `-h` einen Hilfetext abrufen. Der Text einer Zellen-Magics darf nicht leer sein. Wenn Sie Hilfe abrufen, müssen Sie daher einen Text eingeben, auch wenn es nur ein einzelnes Zeichen ist. Beispiel:
>
> ```
> %%gremlin --help
>   x
> ```

Variable Injektion in Zellen- oder Zeilen-Magics

In einem Notebook definierte Variablen können in jeder Zellen- oder Zeilen-Magics im Notebook im folgenden Format referenziert werden: `${VAR_NAME}`.

Angenommen, Sie definieren die folgenden Variablen:

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

Dann ist diese Gremlin-Abfrage in einer Zellen-Magics:

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

Dies entspricht:

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

Abfrageargumente, die mit allen Abfragesprachen funktionieren

Die folgenden Abfrageargumente funktionieren mit `%%gremlin`-, `%%opencypher`- und `%%sparql`-Magics in der Neptune-Workbench:

Allgemeine Abfrageargumente

- **`--store-to`** (oder **`-s`**) – Gibt den Namen der Variablen an, in der die Abfrageergebnisse gespeichert werden sollen.
- **`--silent`** – Wenn vorhanden, wird nach Abschluss der Abfrage keine Ausgabe angezeigt.
- **`--group-by`** (oder **`-g`**) – Gibt die Eigenschaft an, die zum Gruppieren von Knoten verwendet wird (z. B. `code` oder `T.region`). Die Farbe der Eckpunkte ist von der Gruppe abhängig, der sie zugewiesen sind.
- **`--ignore-groups`** – Wenn vorhanden, werden alle Gruppierungsoptionen ignoriert.
- **`--display-property`** (oder **`-d`**) – Gibt die Eigenschaft an, deren Wert für jeden Eckpunkt angezeigt werden soll.

  Der Standardwert für jede Abfragesprache ist wie folgt:

  - Für Gremlin: `T.label`.
  - Für openCypher: `~labels`.
  - Für SPARQL: `type`.
- **`--edge-display-property`** (or **`-t`**) – Gibt die Eigenschaft an, deren Wert für jede Kante angezeigt werden soll.

Der Standardwert für jede Abfragesprache ist wie folgt:

- Für Gremlin: `T.label`.
- Für openCypher: `~labels`.
- Für SPARQL: `type`.

- **`--tooltip-property`** (oder **`-de`**) – Gibt die Eigenschaft an, deren Wert als QuickInfo für jeden Knoten angezeigt werden soll.

  Der Standardwert für jede Abfragesprache ist wie folgt:

  - Für Gremlin: `T.label`.
  - Für openCypher: `~labels`.
  - Für SPARQL: `type`.

- **`--edge-tooltip-property`** (oder **`-te`**) – Gibt eine Eigenschaft an, deren Wert als QuickInfo für jede Kante angezeigt werden soll.

  Der Standardwert für jede Abfragesprache ist wie folgt:

  - Für Gremlin: `T.label`.
  - Für openCypher: `~labels`.
  - Für SPARQL: `type`.

- **`--label-max-length`** (oder **`-l`**) – Gibt die maximale Zeichenlänge einer Eckpunktbezeichnung an. Der Standardwert ist 10.
- **`--edge-label-max-length`** (oder **`-le`**) – Gibt die maximale Zeichenlänge einer Kantenbezeichnung an. Der Standardwert ist 10.

  Nur im Fall von openCypher ist dies stattdessen `--rel-label-max-length` oder `-rel`.

- **`--simulation-duration`** (oder **`-sd`**) – Gibt die maximale Dauer der Visualisierungsphysik-Simulation an. Der Standardwert ist 1 500 ms.
- **`--stop-physics`** (oder **`-sp`**) – Deaktiviert die Visualisierungsphysik nach Stabilisierung der ursprünglichen Simulation.

Eigenschaftswerte für diese Argumente können entweder aus einem einzelnen Eigenschaftsschlüssel oder aus einer JSON-Zeichenfolge bestehen, die für jeden Labeltyp eine andere Eigenschaft angeben kann. Eine JSON-Zeichenfolge kann nur durch Variableneinfügung angegeben werden.

Die Zeilen-Magics `%seed`

Die Zeilen-Magics `%seed` ist eine bequeme Methode, wie Sie Ihrem Neptune-Endpunkt Daten hinzufügen können, um Gremlin-, openCypher- oder SPARQL-Abfragen zu untersuchen und mit ihnen zu experimentieren. Sie stellt ein Formular bereit, in dem Sie das Datenmodell auswählen können, das Sie untersuchen möchten (Eigenschaftsdiagramm oder RDF). Anschließend können Sie aus verschiedenen Beispieldatensätzen auswählen, die Neptune bereitstellt.

Die Zeilen-Magics `%load`

Die Zeilen-Magics `%load` generiert ein Formular, mit dem Sie eine Massenladeanforderung an Neptune senden können (siehe Neptune-Loader-Befehl). Die Quelle muss ein Amazon-S3-Pfad in derselben Region wie der Neptune-Cluster sein.

Die Zeilen-Magics `%load_ids`

Die Zeilen-Magics `%load_ids` ruft die Lade-IDs ab, die an den Host-Endpunkt des Notebooks gesendet wurden (siehe Neptune Loader-Anforderungsparameter Get-Status). Die Anforderung hat folgendes Format:

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

Die Zeilen-Magics `%load_status`

Die Zeilen-Magics `%load_status` ruft den Ladestatus eines bestimmten Ladeauftrags ab, der an den Host-Endpunkt des Notebooks gesendet wurde, angegeben durch die Zeileneingabe (siehe Neptune Loader-Anforderungsparameter Get-Status). Die Anforderung hat folgendes Format:

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

Die Zeilen-Magics sieht wie folgt aus:

```
%load_status load id
```

Die Zeilen-Magics `%cancel_load`

Die Zeilen-Magics `%cancel_load` bricht einen bestimmten Ladeauftrag ab (siehe Neptune-Loader – Auftrag abbrechen). Die Anforderung hat folgendes Format:

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

Die Zeilen-Magics sieht wie folgt aus:

```
%cancel_load load id
```

Die Zeilen-Magics `%status`

Ruft Statusinformationen vom Host-Endpunkt des Notebooks ab (%graph_notebook_configzeigt den Host-Endpunkt an).

Für Neptune DB-Hosts werden die Statusinformationen vom Integritätsstatus-Endpunkt abgerufen. Für Neptune Analytics-Hosts wird der Status über die GetGraph API abgerufen. Weitere Informationen finden Sie unter %get_graph.

Die Zeilen-Magics `%gremlin_status`

Ruft Informationen zum Status einer Gremlin-Abfrage ab.

Die Zeilen-Magics `%opencypher_status` (auch `%oc_status`)

Ruft den Abfragestatus für eine opencypher-Abfrage ab. Die Zeilen-Magics verwendet die folgenden optionalen Argumente:

- **`--queryId`** oder **`-q`** – Gibt die ID einer bestimmten ausgeführten Abfrage an, deren Status angezeigt werden soll.
- **`--cancelQuery`** oder **`-c`** – Bricht eine ausgeführte Abfrage ab. Nimmt keinen Wert an.
- **`--silent-cancel`** or **`-s`** – Wenn `--silent` beim Abbrechen einer Abfrage auf `true`festgelegt ist, wird die ausgeführte Abfrage mit dem HTTP-Antwortcode `200` abgebrochen. Andernfalls wäre der HTTP-Antwortcode `500`.
- **`--store-to`** – Gibt den Namen der Variablen an, in der die Abfrageergebnisse gespeichert werden sollen.
- **`-w/--includeWaiting`**— Nur Neptune DB. Wenn der Wert auf true gesetzt ist und keine anderen Parameter vorhanden sind, werden Statusinformationen für wartende Abfragen sowie für laufende Abfragen zurückgegeben. Dieser Parameter akzeptiert keinen Wert.
- **`--state`**— Nur Neptune Analytics. Gibt an, für welche Teilmenge von Abfragestatus der Status abgerufen werden soll.

- **`-m/--maxResults`**— Nur Neptune Analytics. Legt eine Obergrenze für die Menge der zurückgegebenen Abfragen fest, die dem Wert von entsprechen. `--state`
- **`--silent`** – Wenn vorhanden, wird nach Abschluss der Abfrage keine Ausgabe angezeigt.

Die Zeilen-Magics `%stream_viewer`

Die Zeilen-Magics `%stream_viewer` zeigt eine Schnittstelle an, die eine interaktive Untersuchung der Einträge ermöglicht, die in Neptune-Streams protokolliert sind, wenn für den Neptune-Cluster Streams aktiviert sind. Der Befehl akzeptiert die folgenden optionalen Argumente:

- **`language`** – Die Abfragesprache der Stream-Daten: `gremlin` oder `sparql`. Wenn Sie dieses Argument nicht angeben, ist der Standardwert `gremlin`.
- **`--limit`** – Gibt die maximale Anzahl von Stream-Einträgen an, die pro Seite angezeigt werden sollen. Wenn Sie dieses Argument nicht angeben, ist der Standardwert `10`.

> **Note**
>
> Die Zeilen-Magics `%stream_viewer` wird nur in den Engine-Versionen 1.0.5.1 und früher vollständig unterstützt.

Die Zeilen-Magics `%sparql_status`

Ruft Informationen zum Status einer SPARQL-Abfrage ab.

Die Zeilen-Magics `%graph_notebook_config`

Diese Zeilen-Magics zeigt das JSON-Objekt an, das die Konfiguration enthält, die das Notebook für die Kommunikation mit Neptune verwendet. Die Konfiguration enthält Folgendes:

- `host`: Der Endpunkt, mit dem die Verbindung hergestellt und an den Befehle ausgegeben werden sollen.
- `port`: Der Port, der für die Ausgabe von Befehlen an Neptune verwendet wird. Der Standardwert ist `8182`.

- `auth_mode`: Der Authentifizierungsmodus, der verwendet werden soll, wenn Befehle an Neptune ausgegeben werden. Dieser muss `IAM` sein, wenn eine Verbindung mit einem Cluster hergestellt wird, für den die IAM-Authentifizierung aktiviert ist, andernfalls `DEFAULT`.
- `load_from_s3_arn`: Gibt den Amazon-S3-ARN an, den die Magics `%load` verwenden soll. Wenn dieser Wert leer ist, muss der ARN im Befehl `%load` angegeben werden.
- `ssl`: Ein boolescher Wert, der angibt, ob die Verbindung mit Neptune über TLS hergestellt werden soll oder nicht. Der Standardwert ist `true`.
- `aws_region`: Die Region, in der dieses Notebook bereitgestellt wird. Diese Informationen werden für die IAM-Authentifizierung und für `%load`-Anforderungen verwendet.

Sie können die Konfiguration ändern, indem Sie die Ausgabe für `%graph_notebook_config` in eine neue Zelle kopieren und dort Änderungen ausführen. Wenn Sie dann die Zellen-Magics `%%graph_notebook_config` für die neue Zelle ausführen, wird die Konfiguration entsprechend geändert.

Die Zeilen-Magics `%graph_notebook_host`

Legt den Host des Notebooks als Zeileneingabe fest.

Die Zeilen-Magics `%graph_notebook_version`

Die Zeilen-Magics `%graph_notebook_version` gibt die Versionsnummer des Neptune-Workbench-Notebooks zurück. In Version `1.27` wurde beispielsweise die Diagrammvisualisierung eingeführt.

Die Zeilen-Magics `%graph_notebook_service`

Die `%graph_notebook_service` Linienmagie legt den Leitungseingang als Servicenamen fest, der für Neptune-Anfragen verwendet wird.

Die Zeilen-Magics `%graph_notebook_vis_options`

Die Zeilen-Magics `%graph_notebook_vis_options` zeigt die aktuellen Visualisierungseinstellungen an, die das Notebook verwendet. Diese Optionen werden in der Dokumentation für vis.js beschrieben.

Sie können diese Einstellungen ändern, indem Sie die Ausgabe in eine neue Zelle kopieren, die gewünschten Änderungen ausführen und dann die Zellen-Magics `%%graph_notebook_vis_options` für die Zelle anwenden.

Um die Visualisierungseinstellungen auf die Standardwerte zurückzusetzen, können Sie die Zeilen-Magics `%graph_notebook_vis_options` mit dem Parameter `reset` ausführen. Hierdurch werden alle Visualisierungseinstellungen zurückgesetzt:

```
%graph_notebook_vis_options reset
```

Die Zeilen-Magics `%statistics`

Die Zeilen-Magics `%statistics` wird verwendet, um DFE-Engine-Statistiken abzurufen oder zu verwalten (siehe Verwalten von Statistiken, die die Neptune-DFE-Engine verwenden soll). Diese Magics kann auch verwendet werden, um eine Diagrammübersicht abzurufen.

Sie akzeptiert die folgenden Parameter:

- **`--language`** – Die Abfragesprache des Statistikendpunkts: `propertygraph` (oder `pg`) oder `rdf`.

  Wenn nicht angegeben, ist der Standardwert `propertygraph`.
- **`--mode`** (oder **`-m`**) – Gibt den Typ der Anforderung oder Aktion an, die gesendet werden soll: `status`, `disableAutoCompute`, `enableAutoCompute`, `refresh`, `delete`, `detailed` oder `basic`).

  Wenn nicht angegeben, ist der Standardwert `status`. Wenn jedoch `--summary` angegeben ist, ist der Standardwert `basic`.
- **`--summary`** – Ruft die Diagrammübersicht vom Statistikübersichtendpunkt der ausgewählten Sprache ab.
- **`--silent`** – Wenn vorhanden, wird nach Abschluss der Abfrage keine Ausgabe angezeigt.
- **`--store-to`** – Wird verwendet, um eine Variable anzugeben, in der die Abfrageergebnisse gespeichert werden sollen.

Die Zeilen-Magics `%summary`

Die Zeilen-Magics `%summary` wird verwendet, um Informationen zur Diagrammübersicht abzurufen. Sie ist ab Neptune-Engine-Version `1.2.1.0` verfügbar.

Sie akzeptiert die folgenden Parameter:

- **`--language`** – Die Abfragesprache des Statistikendpunkts: `propertygraph` (oder `pg`) oder `rdf`.

  Wenn nicht angegeben, ist der Standardwert `propertygraph`.
- **`--detailed`** – Schaltet die Anzeige von Strukturfeldern in der Ausgabe ein oder aus.

  Wenn nicht angegeben, ist der Übersichtsanzeigemodus `basic` der Standardwert.
- **`--silent`** – Wenn vorhanden, wird nach Abschluss der Abfrage keine Ausgabe angezeigt.
- **`--store-to`** – Wird verwendet, um eine Variable anzugeben, in der die Abfrageergebnisse gespeichert werden sollen.

Die Zeilen-Magics **`%reset_graph`**

Die `%reset_graph` (oder`%_graph_reset`) Line Magic führt einen ResetGraphAufruf gegen den Neptune Analytics-Endpunkt aus. Sie akzeptiert die folgenden optionalen Zeileneingaben:

- -ns oder -- no-skip-snapshot - Falls vorhanden, wird ein letzter Graph-Snapshot erstellt, bevor die Graphdaten gelöscht werden.
- --silent — Falls vorhanden, wird nach dem Absenden des Reset-Aufrufs keine Ausgabe angezeigt.
- --store-to — Wird verwendet, um eine Variable anzugeben, in der die Antwort gespeichert werden soll. ResetGraph

Die Zeilen-Magics **`%get_graph`**

Die `%get_graph` Linienmagie ruft über die API Informationen über einen Graphen ab. GetGraph
Diese Magie ist funktionell identisch mit der %status Verwendung mit Neptune Analytics.

Die Zeilen-Magics **`%degreeDistribution`**

Die `%degreeDistribution` Linienmagie erzeugt ein interaktives Histogramm, das die Gradverteilung der Scheitelpunkte in einem Graphen visualisiert. Das Histogramm zeigt die Anzahl der Scheitelpunkte (Y-Achse) an, die einen bestimmten Grad (X-Achse) haben. Es ist nur mit Neptune Analytics verfügbar.

Es akzeptiert die folgenden Parameter, die auch im Drop-down-Menü ausgewählt werden können:

- **`--traversalDirection`**— Gibt an, welcher Grad analysiert werden soll. Muss einer der folgenden Werte sein: `"both"` (Standard)`"inbound"`, oder`"outbound"`.
- **`--vertexLabels`**— Filtert Scheitelpunkte nach den bereitgestellten Beschriftungen (z. B.`airport country`). Die Standardeinstellung ist eine leere Liste.
- **`--edgeLabels`**— Filtert Kanten nach den bereitgestellten Beschriftungen (z. B.`route`). Die Standardeinstellung ist eine leere Liste.

Die Zellen-Magics **`%%sparql`**

Die Zellen-Magics `%%sparql` gibt eine SPARQL-Abfrage an den Neptune-Endpunkt aus. Sie akzeptiert die folgenden optionalen Zeileneingaben:

- **`-h`** oder **`--help`** – Gibt Hilfetext zu diesen Parametern zurück.
- **`--path`** – Stellt dem SPARQL-Endpunkt einen Pfad voran. Wenn Sie zum Beispiel `--path "abc/def"` angeben, wäre *`host`*`:`*`port`*`/abc/def` der aufgerufene Endpunkt.
- **`--expand-all`** – Dies ist ein Abfragevisualisierungshinweis, der den Visualizer anweist, alle Ergebnisse für `?s ?p ?o` unabhängig vom Bindungstyp in das Diagramm aufzunehmen.

  Standardmäßig enthält eine SPARQL-Visualisierung nur Tripelmuster, bei denen o? entweder `uri` oder bnode (ein leerer Knoten) ist. Alle anderen ?o-Bindungstypen, wie Literalzeichenfolgen oder Ganzzahlen, werden als Eigenschaften des ?s-Knotens behandelt und können im Bereich Details auf der Registerkarte Diagramm angezeigt werden.

  Wenn Sie Literalwerte als Eckpunkte in die Visualisierung einfügen möchten, verwenden Sie stattdessen den Abfragehinweis `--expand-all`.

  Sie dürfen diesen Visualisierungshinweis nicht mit explain-Parametern kombinieren, da explain-Abfragen nicht visualisiert werden.
- **`--explain-type`** – Wird verwendet, um den explain-Modus anzugeben, der verwendet werden soll (`dynamic`, `static` oder `details`).
- **`--explain-format`** – Wird verwendet, um das Antwortformat für eine explain-Abfrage anzugeben (`text/csv` oder `text/html`).
- **`--store-to`** – Wird verwendet, um eine Variable anzugeben, in der die Abfrageergebnisse gespeichert werden sollen.

Beispiel für eine `explain`-Abfrage:

```
%%sparql explain

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

Beispiel für eine Visualisierungsabfrage mit dem Visualisierungshinweisparameter `--expand-all` (siehe SPARQL-Visualisierung):

```
%%sparql --expand-all

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

Die Zellen-Magics **`%%gremlin`**

Die `%%gremlin` Zellmagie sendet eine Gremlin-Abfrage an den Neptun-Endpunkt unter Verwendung von. WebSocket Sie akzeptiert eine optionale Zeileneingabe, um in den Modus Gremlin `explain` /> oder Gremlin-`profile`-API zu wechseln, und die getrennte optionale Eingabe eines Visualisierungshinweises, um das Verhalten der Visualisierungsausgabe zu ändern (siehe Gremlin-Visualisierung).

Beispiel für eine `explain`-Abfrage:

```
%%gremlin explain

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

Beispiel für eine `profile`-Abfrage:

```
%%gremlin profile

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

Beispiel für eine Visualisierungsabfrage mit einem Visualisierungshinweisparameter:

```
%%gremlin -p v,outv

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

Optionale Parameter für **`%%gremlin profile`**-Abfragen

- **`--profile-chop`** – Gibt die maximale Länge der Profilergebnis-Zeichenfolge an. Wenn Sie dieses Argument nicht angeben, ist der Standardwert 250.
- **`--profile-serializer`** – Gibt den Serializer an, der für die Ergebnisse verwendet werden soll. Zulässige Werte sind alle gültigen Enumerationswerte des MIME-Typs oder des TinkerPop Treibers „Serializers“. Wenn Sie dieses Argument nicht angeben, ist der Standardwert `application.json`.
- **`--profile-no-results`** – Zeigt nur die Ergebniszahl an. Wenn dieser Parameter nicht verwendet wird, werden standardmäßig alle Abfrageergebnisse im Profilbericht angezeigt.
- **`--profile-indexOps`** – Zeigt einen detaillierten Bericht zu allen Indexoperationen an.

Die Zellen-Magics **`%%opencypher`** (auch **`%%oc`**)

Die Zellen-Magics `%%opencypher` (auch als `%%oc` abgekürzt) gibt eine openCypher-Abfrage an den Neptune-Endpunkt aus. Sie akzeptiert die folgenden optionalen Zeileneingabenargumente:

- **`--mode`**— Der Abfragemodus: entweder oder. `query` `bolt` Wenn Sie dieses Argument nicht angeben, ist der Standardwert `query`.
- **`--group-by`** oder **`-g`** – Gibt die Eigenschaft an, die zum Gruppieren von Knoten verwendet wird. Beispiel, `code, ~id`. Wenn Sie dieses Argument nicht angeben, ist der Standardwert `~labels`.
- **`--ignore-groups`** – Wenn vorhanden, werden alle Gruppierungsoptionen ignoriert.
- **`--display-property`** oder **`-d`** – Gibt die Eigenschaft an, deren Wert für jeden Eckpunkt angezeigt werden soll. Wenn Sie dieses Argument nicht angeben, ist der Standardwert `~labels`.
- **`--edge-display-property`** oder **`-de`** – Gibt die Eigenschaft an, deren Wert für jede Kante angezeigt werden soll. Wenn Sie dieses Argument nicht angeben, ist der Standardwert `~labels`.
- **`--explain-type`**— Wird verwendet, um den zu verwendenden Erklärungsmodus anzugeben (einer der folgenden Modi: dynamisch, statisch, Debug oder Details).
- **`--label-max-length`** oder **`-l`** – Gibt die maximale Anzahl der Zeichen einer Eckpunktbezeichnung an, die angezeigt werden können. Wenn Sie dieses Argument nicht angeben, ist der Standardwert `10`.
- **`--store-to`** oder **`-s`**) – Gibt den Namen der Variablen an, in der die Abfrageergebnisse gespeichert werden sollen.

- **`--plan-cache`** oder **`-pc`**: Gibt den zu verwendenden Plan-Cache-Modus an. Der Standardwert ist `auto`.
- **`--query-timeout`** oder **`-qt`**: Gibt das maximale Abfrage-Timeout in Millisekunden an. Der Standardwert ist `1800000`.
- **`--query-parameters`** oder **`qp`**: Parameterdefinitionen, die auf die Abfrage angewendet werden sollen. Diese Option kann entweder einen einzelnen Variablennamen oder eine Zeichenfolgendarstellung der Zuordnung annehmen.

  Beispiel für die Verwendung von **`--query-parameters`**

  1. Definieren Sie eine Zuordnung von OpenCypher-Parametern in einer Notebook-Zelle.

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

  2. Übergeben Sie die Parameter an `--query-parameters` in einer anderen Zelle mit `%%oc`.

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

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

Die Zellen-Magics **`%%graph_notebook_config`**

Die Zellen-Magics `%%graph_notebook_config` verwendet ein JSON-Objekt mit Informationen zur Konfiguration, um die Einstellungen zu ändern, die das Notebook für die Kommunikation mit Neptune verwendet, wenn möglich. Die Konfiguration hat das Format, das von der Zeilen-Magics %graph_notebook_config zurückgegeben wird.

Beispiel:

```
%%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"
}
```

Die Zellen-Magics `%%graph_notebook_vis_options`

Mit der Zellen-Magics `%%graph_notebook_vis_options` können Sie Visualisierungsoptionen für das Notebook festlegen. Sie können die von der Zeilen-Magics `%graph-notebook-vis-options` zurückgegebenen Einstellungen kopieren, in eine neue Zelle kopieren, Änderungen ausführen und die Zellen-Magics `%%graph_notebook_vis_options` verwenden, um die neuen Werte festzulegen.

Diese Optionen werden in der Dokumentation für vis.js beschrieben.

Um die Visualisierungseinstellungen auf die Standardwerte zurückzusetzen, können Sie die Zeilen-Magics `%graph_notebook_vis_options` mit dem Parameter `reset` ausführen. Hierdurch werden alle Visualisierungseinstellungen zurückgesetzt:

```
%graph_notebook_vis_options reset
```

Die Zeilen-Magics `%neptune_ml`

Sie können die Zeilen-Magics `%neptune_ml` verwenden, um verschiedene Neptune-ML-Operationen zu initiieren und zu verwalten.

> **Note**
>
> Sie können auch die Zellen-Magics %%neptune_ml verwenden, um einige Neptune-ML-Operationen zu initiieren und zu verwalten.

- **`%neptune_ml export start`** – Startet einen neuen Exportauftrag.

  Parameters

  - **`--export-url`***exporter-endpoint*— (optional) Der Amazon API Gateway Gateway-Endpunkt, über den der Exporteur aufgerufen werden kann.
  - **`--export-iam`** – (optional) Flag, das anzeigt, dass Anforderungen an die Export-URL mit SigV4 signiert werden müssen.

- **`--export-no-ssl`** – (optional) Flag, das anzeigt, dass SSL nicht verwendet werden sollte, wenn eine Verbindung zum Exporter hergestellt wird.
- **`--wait`** – (optional) Flag, das anzeigt, dass die Operation warten soll, bis der Export abgeschlossen ist.
- **`--wait-interval`***interval-to-wait*— (optional) Legt die Zeit in Sekunden zwischen den Exportstatusprüfungen fest (Standard: 60).
- **`--wait-timeout`***timeout-seconds*— (optional) Legt die Wartezeit in Sekunden fest, bis der Exportauftrag abgeschlossen ist, bevor der neueste Status zurückgegeben wird (Standard: 3.600).
- **`--store-to`***location-to-store-result*— (optional) Die Variable, in der das Exportergebnis gespeichert werden soll. Wenn `--wait` angegeben ist, wird der endgültige Status dort gespeichert.

- **`%neptune_ml export status`** – Ruft den Status eines Exportauftrags ab.

  Parameters

  - **`--job-id`***export job ID*— Die ID des Exportjobs, für den der Status abgerufen werden soll.
  - **`--export-url`***exporter-endpoint*— (optional) Der Amazon API Gateway Gateway-Endpunkt, über den der Exporteur aufgerufen werden kann.
  - **`--export-iam`** – (optional) Flag, das anzeigt, dass Anforderungen an die Export-URL mit SigV4 signiert werden müssen.
  - **`--export-no-ssl`** – (optional) Flag, das anzeigt, dass SSL nicht verwendet werden sollte, wenn eine Verbindung zum Exporter hergestellt wird.
  - **`--wait`** – (optional) Flag, das anzeigt, dass die Operation warten soll, bis der Export abgeschlossen ist.
  - **`--wait-interval`***interval-to-wait*— (optional) Legt die Zeit in Sekunden zwischen den Exportstatusprüfungen fest (Standard: 60).
  - **`--wait-timeout`***timeout-seconds*— (optional) Legt die Wartezeit in Sekunden fest, bis der Exportauftrag abgeschlossen ist, bevor der neueste Status zurückgegeben wird (Standard: 3.600).
  - **`--store-to`***location-to-store-result*— (optional) Die Variable, in der das Exportergebnis gespeichert werden soll. Wenn `--wait` angegeben ist, wird der endgültige Status dort gespeichert.

- **`%neptune_ml dataprocessing start`** – Startet den Neptune-ML-Datenverarbeitungsschritt.

  Parameters

  - **`--job-id`***`ID for this job`*— (optional) ID, die diesem Job zugewiesen werden soll.
  - **`--s3-input-uri`***`S3 URI`*— (optional) Die S3-URI, unter der die Eingabe für diesen Datenverarbeitungsjob gefunden werden soll.
  - **`--config-file-name`***`file name`*— (optional) Name der Konfigurationsdatei für diesen Datenverarbeitungsjob.
  - **`--store-to`***`location-to-store-result`*— (optional) Die Variable, in der das Datenverarbeitungsergebnis gespeichert werden soll.
  - **`--instance-type`***`(instance type)`*— (optional) Die Instanzgröße, die für diesen Datenverarbeitungsjob verwendet werden soll.
  - **`--wait`** – (optional) Flag, das anzeigt, dass die Operation warten soll, bis die Datenverarbeitung abgeschlossen ist.
  - **`--wait-interval`***`interval-to-wait`*— (optional) Legt die Zeit in Sekunden zwischen den Datenverarbeitungsstatusprüfungen fest (Standard: 60).
  - **`--wait-timeout`***`timeout-seconds`*— (optional) Legt die Wartezeit in Sekunden fest, bis der Datenverarbeitungsauftrag abgeschlossen ist, bevor der neueste Status zurückgegeben wird (Standard: 3.600).
- **`%neptune_ml dataprocessing status`** – Ruft den Status eines Datenverarbeitungsauftrags ab.

  Parameters

  - **`--job-id`***`ID of the job`*— ID des Jobs, für den der Status abgerufen werden soll.
  - **`--store-to`***`instance type`*— (optional) Die Variable, in der das Ergebnis des Modelltrainings gespeichert werden soll.
  - **`--wait`** – (optional) Flag, das anzeigt, dass die Operation warten soll, bis das Modelltraining abgeschlossen ist.
  - **`--wait-interval`***`interval-to-wait`*— (optional) Legt die Zeit in Sekunden zwischen den Statusprüfungen des Modelltrainings fest (Standard: 60).
  - **`--wait-timeout`***`timeout-seconds`*— (optional) Legt die Wartezeit in Sekunden fest, bis der Datenverarbeitungsauftrag abgeschlossen ist, bevor der neueste Status zurückgegeben wird (Standard: 3.600).

- **`%neptune_ml training start`** – Startet den Neptune-ML-Modelltrainingsprozess.

  Parameters

  - **`--job-id`***`ID for this job`*— (optional) ID, die diesem Job zugewiesen werden soll.
  - **`--data-processing-id`***`dataprocessing job ID`*— (optional) ID des Datenverarbeitungsjobs, der die Artefakte erstellt hat, die für das Training verwendet werden sollen.
  - **`--s3-output-uri`***`S3 URI`*— (optional) Die S3-URI, unter der die Ausgabe dieses Modelltrainingsjobs gespeichert werden soll.
  - **`--instance-type`***`(instance type)`*— (optional) Die Instanzgröße, die für diesen Modelltraining-Job verwendet werden soll.
  - **`--store-to`***`location-to-store-result`*— (optional) Die Variable, in der das Ergebnis des Modelltrainings gespeichert werden soll.
  - **`--wait`** – (optional) Flag, das anzeigt, dass die Operation warten soll, bis das Modelltraining abgeschlossen ist.
  - **`--wait-interval`***`interval-to-wait`*— (optional) Legt die Zeit in Sekunden zwischen den Statusprüfungen des Modelltrainings fest (Standard: 60).
  - **`--wait-timeout`***`timeout-seconds`*— (optional) Legt die Wartezeit in Sekunden fest, bis der Modelltrainingsjob abgeschlossen ist, bevor der neueste Status zurückgegeben wird (Standard: 3.600).
- **`%neptune_ml training status`** – Ruft den Status eines Neptune-ML-Modelltrainingsauftrags ab.

  Parameters

  - **`--job-id`***`ID of the job`*— ID des Jobs, für den der Status abgerufen werden soll.
  - **`--store-to`***`instance type`*— (optional) Die Variable, in der das Statusergebnis gespeichert werden soll.
  - **`--wait`** – (optional) Flag, das anzeigt, dass die Operation warten soll, bis das Modelltraining abgeschlossen ist.
  - **`--wait-interval`***`interval-to-wait`*— (optional) Legt die Zeit in Sekunden zwischen den Statusprüfungen des Modelltrainings fest (Standard: 60).
  - **`--wait-timeout`***`timeout-seconds`*— (optional) Legt die Wartezeit in Sekunden fest, bis der Datenverarbeitungsauftrag abgeschlossen ist, bevor der neueste Status zurückgegeben wird (Standard: 3.600).

- **`%neptune_ml endpoint create`** – Erstellt einen Abfrageendpunkt für ein Neptune-ML-Modell.

  Parameters

  - **`--job-id`***`ID for this job`*— (optional) ID, die diesem Job zugewiesen werden soll.
  - **`--model-job-id`***`model-training job ID`*— (optional) ID des Modelltrainingsjobs, für den ein Abfrageendpunkt erstellt werden soll.
  - **`--instance-type`***`(instance type)`*— (optional) Die Instanzgröße, für die der Abfrageendpunkt verwendet werden soll..
  - **`--store-to`***`location-to-store-result`*— (optional) Die Variable, in der das Ergebnis der Endpunkterstellung gespeichert werden soll.
  - **`--wait`** – (optional) Flag, das anzeigt, dass die Operation warten soll, bis die Endpunkterstellung abgeschlossen ist.
  - **`--wait-interval`***`interval-to-wait`*— (optional) Legt die Zeit in Sekunden zwischen Statusprüfungen fest (Standard: 60).
  - **`--wait-timeout`***`timeout-seconds`*— (optional) Legt die Wartezeit in Sekunden fest, bis der Job zur Endpunkterstellung abgeschlossen ist, bevor der neueste Status zurückgegeben wird (Standard: 3.600).
- **`%neptune_ml endpoint status`** – Ruft den Status eines Neptune-ML-Abfrageendpunkts ab.

  Parameters

  - **`--job-id`***`endpoint creation ID`*— (optional) ID eines Auftrags zur Endpunkterstellung, für den der Status gemeldet werden soll.
  - **`--store-to`***`location-to-store-result`*— (optional) Die Variable, in der das Statusergebnis gespeichert werden soll.
  - **`--wait`** – (optional) Flag, das anzeigt, dass die Operation warten soll, bis die Endpunkterstellung abgeschlossen ist.
  - **`--wait-interval`***`interval-to-wait`*— (optional) Legt die Zeit in Sekunden zwischen Statusprüfungen fest (Standard: 60).
  - **`--wait-timeout`***`timeout-seconds`*— (optional) Legt die Wartezeit in Sekunden fest, bis der Job zur Endpunkterstellung abgeschlossen ist, bevor der neueste Status zurückgegeben wird (Standard: 3.600).

Die Zellen-Magics `%%neptune_ml`

Die Zellen-Magics `%%neptune_ml` ignoriert Zeileneingaben wie `--job-id` oder `--export-url`. Stattdessen können Sie diese und andere Eingaben im Zelltext eingeben.

Sie können diese Eingaben auch in einer anderen Zelle speichern, die einer Jupyter-Variablen zugewiesen ist, und sie dann mithilfe dieser Variablen in den Zelltext einfügen. So können Sie diese Eingaben immer wieder verwenden, ohne sie jedes Mal erneut eingeben zu müssen.

Dies funktioniert nur, wenn die einfügende Variable der einzige Inhalt der Zelle ist. Sie können in derselben Zelle nicht mehrere Variablen oder eine Kombination aus Text und einer Variablen verwenden.

Die Zellen-Magics `%%neptune_ml export start` kann beispielsweise ein JSON-Dokument im Zelltext verwenden, das alle Parameter enthält, die in Parameter, die zur Steuerung des Neptun-Exportprozesses verwendet werden beschrieben werden.

Im Notebook Neptune-ML-01-Introduction-to-Node-Classification-Gremlin sehen Sie unter Konfigurieren von Features im Abschnitt Exportieren der Daten- und Modellkonfiguration, wie die folgende Zelle Exportparameter in einem Dokument enthält, das einer Jupyter-Variablen namens `export-params` zugewiesen ist:

```
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"}
```

Wenn Sie diese Zelle ausführen, speichert Jupyter das Parameterdokument unter diesem Namen. Anschließend können Sie mit `${export_params}` das JSON-Dokument wie folgt in den Text einer `%%neptune_ml export start cell` einfügen:

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

${export_params}
```

Verfügbare Formulare für die Zellen-Magics **`%%neptune_ml`**

Die Zellen-Magics `%%neptune_ml` kann in den folgenden Formularen verwendet werden:

- **`%%neptune_ml export start`** – Startet einen Neptune-ML-Exportvorgang.
- **`%%neptune_ml dataprocessing start`** – Startet einen Neptune-ML-Datenverarbeitungsauftrag.
- **`%%neptune_ml training start`** – Startet einen Neptune-ML-Modelltrainingsauftrag.
- **`%%neptune_ml endpoint create`** – Erstellt einen Neptune-ML-Abfrageendpunkt für ein Modell.

Visualisieren von Grafikdaten in Amazon Neptune Neptune-Notizbüchern

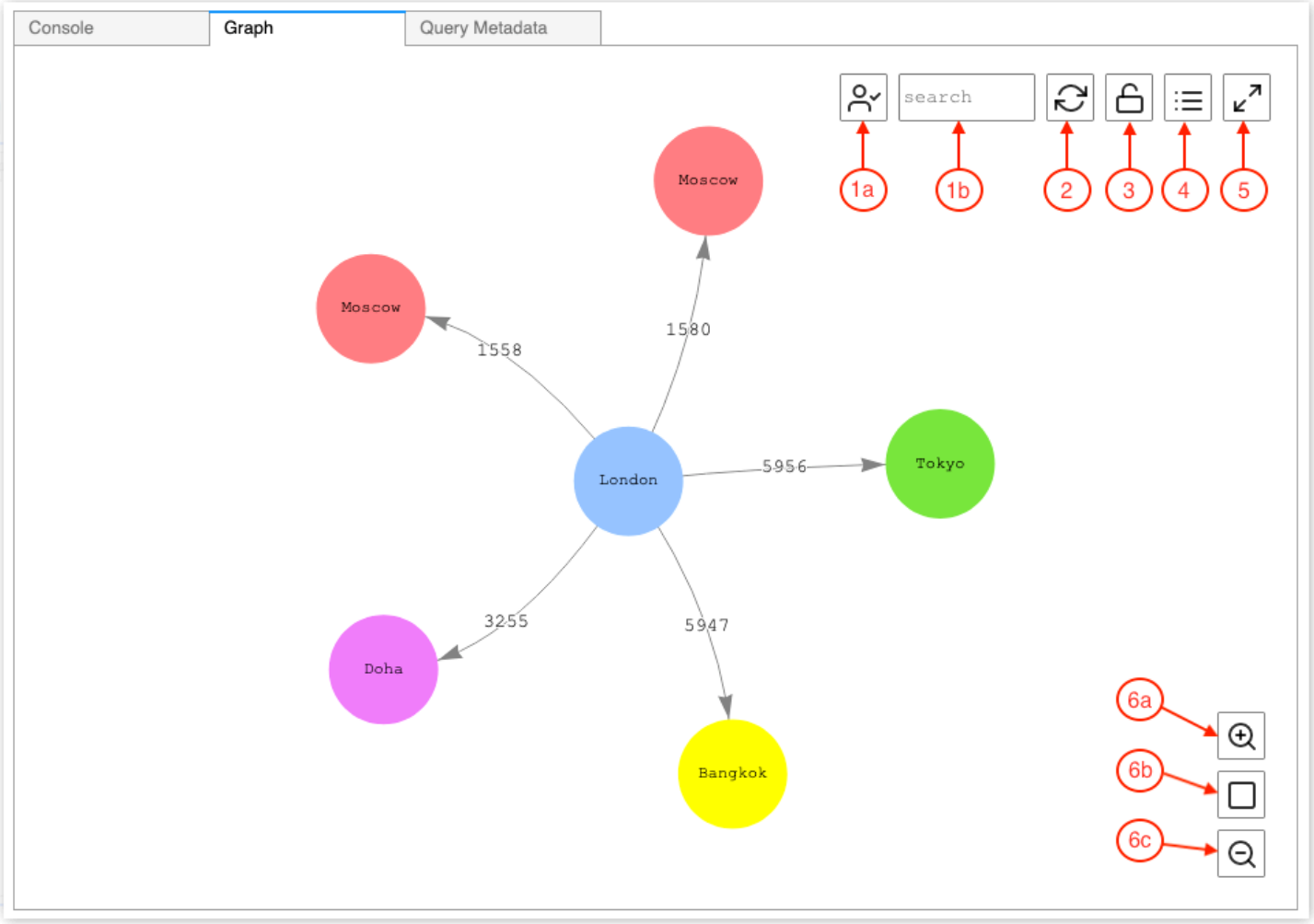
In vielen Fällen kann die Neptune-Workbench ein visuelles Diagramm der Abfrageergebnisse erstellen und diese auch in Tabellenform zurückgeben. Die Diagrammvisualisierung ist in den Abfrageergebnissen auf der Registerkarte Diagramm verfügbar, wenn eine Visualisierung möglich ist.

Zusätzlich zu den hier beschriebenen integrierten Visualisierungsfunktionen können Sie mit Neptune-Diagramm-Notebooks auch erweiterte Visualisierungstools verwenden.

> **Note**
>
> Um auf kürzlich hinzugefügte Funktionen und Problembehebungen in bereits verwendeten Notebooks zuzugreifen, beenden Sie zunächst Ihre Notebook-Instance und starten diese dann neu.

Erkunden Sie die Graph Tab-Oberfläche in Amazon Neptune

Dieses Diagramm zeigt Elemente der Benutzeroberfläche der Registerkarte für Diagramme.



1. Diagrammsuche

   a. UUID-Schalter: Schaltet die Einfügung von ID-Eigenschaftswerten in das Diagramm ein und aus. Die ID-Einfügung ist standardmäßig aktiviert. Wenn diese Option deaktiviert ist, führen Übereinstimmungen mit ID-Eigenschaften, einschließlich Kanteneigenschaften, die auf den Knoten verweisen IDs, nicht zur Hervorhebung von Elementen.

   b. Suchtextfeld: Hebt alle Eckpunkt- und Kanteneigenschaftswerte hervor, die die hier angegebene Textzeichenfolge enthalten.

2. Diagramm zurücksetzen – Führt die Diagrammphysiksimulation erneut aus und legt den Zoom so fest, dass das Diagramm in das Fenster passt.

3. Diagrammphysik umschalten – Schaltet die Ausführung der Diagrammphysiksimulation ein und aus. Die Physik ist standardmäßig aktiviert, sodass das Diagramm dynamisch geändert wird. Wenn diese Option deaktiviert ist, bleiben Eckpunkte in ihrer Position, wenn andere Eckpunkte verschoben werden.

4. Detailansicht – Wenn ein Knoten oder eine Kante ausgewählt ist, wird eine Liste der Eigenschaftsschlüssel und Werte des Elements angezeigt, wenn sie in den Abfrageergebnissen verfügbar sind.
5. Vollbildansicht – Erweitert das Fenster der Registerkarte für Diagramme so, dass es auf den Bildschirm passt. Durch erneutes Klicken wird die Registerkarte für Diagramme minimiert.
6. Zoom-Optionen
   a. Vergrößern
   b. Zoom zurücksetzen: Legt den Zoom so fest, dass alle Eckpunkte in das Fenster der Registerkarte für Diagramme passen.
   c. Verkleinern

Visualisieren von Gremlin-Abfrageergebnissen

Neptune Workbench erstellt eine Visualisierung der Abfrageergebnisse für jede Gremlin-Abfrage, die einen `path` zurückgibt. Um die Visualisierung anzuzeigen, wählen Sie nach der Ausführung der Abfrage die Registerkarte Diagramm rechts neben der Registerkarte Konsole unterhalb der Abfrage aus.

Mithilfe von Abfragevisualisierungshinweisen können Sie steuern, wie die Visualizer-Diagramme die Ausgabe abfragen. Diese Hinweise folgen der Zellen-Magics `%%gremlin`. Ihnen ist der Parametername `--path-pattern` (oder dessen Kurzform `-p`) vorangestellt:

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

Sie können auch das Flag `--group-by` (oder `-g`) verwenden, um eine Eigenschaft der Eckpunkte anzugeben, nach der sie gruppiert werden sollen. So können Sie Farben oder Symbole für verschiedene Gruppen von Eckpunkten angeben.

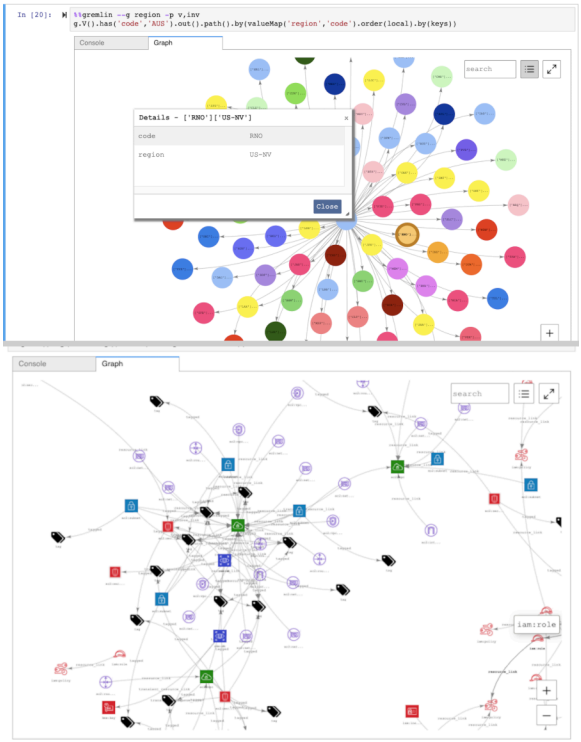
Die Namen der Hinweise reflektieren die Gremlin-Schritte, die allgemein für die Traversierung zwischen Eckpunkten verwendet werden, und sie verhalten sich entsprechend. Sie können mehrere Hinweise kombinieren, getrennt durch Komma, ohne Leerzeichen zwischen ihnen. Die verwendeten Hinweise sollten mit den entsprechenden Gremlin-Schritten in der Abfrage übereinstimmen, die visualisiert wird. Ein Beispiel:

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

Die verfügbaren Visualisierungshinweise sind wie folgt:

```
v
inv
outv
e
ine
oute
```

Beispiele für Diagrammvisualisierungen, die Gruppen verwenden:



Visualisieren von SPARQL-Abfrageergebnissen

Die Neptune-Workbench erstellt eine Visualisierung der Abfrageergebnisse für jede SPARQL-Abfrage in einem der folgenden Formate:

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

Um die Visualisierung anzuzeigen, wählen Sie nach der Ausführung der Abfrage die Registerkarte Diagramm rechts neben der Registerkarte Tabelle unterhalb der Abfrage aus.

Standardmäßig enthält eine SPARQL-Visualisierung nur Tripelmuster, bei denen `o?` entweder `uri` oder `bnode` (ein leerer Knoten) ist. Alle anderen `?o`-Bindungstypen, wie Literalzeichenfolgen oder Ganzzahlen, werden als Eigenschaften des `?s`-Knotens behandelt und können im Bereich Details auf der Registerkarte Diagramm angezeigt werden.

In vielen Fällen sollten Sie jedoch diese Literalwerte als Eckpunkte in die Visualisierung einfügen. Hierzu verwenden Sie den Abfragehinweis `--expand-all` nach der Zellen-Magics `%%sparql`:

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

Dies weist den Visualisierer an, alle Ergebnisse für `?s ?p ?o` unabhängig vom Bindungstyp in das Diagramm aufzunehmen.

Dieser Hinweis wird im gesamten `Air-Routes-SPARQL.ipynb`-Notebook verwendet. Sie können experimentieren, indem Sie die Abfragen mit und ohne Hinweis ausführen, um zu sehen, welche Unterschiede es in der Visualisierung gibt.

Zugriff auf Tutorial-Notebooks für die Visualisierungs in der Neptune-Workbench

Die beiden im Lieferumfang der Neptune-Workbench enthaltenen Tutorial-Notebooks für die Visualisierung enthalten zahlreiche Beispiele in Gremlin und SPARQL dafür, wie Sie Diagrammdaten effektiv abfragen und die Ergebnisse visualisieren.

Navigation zu den Visualisierungs-Notebooks

1. Klicken Sie im linken Navigationsbereich auf die Schaltfläche Notebook öffnen auf der rechten Seite.
2. Nach dem Öffnen der Neptune-Workbench wird bei Ausführung von Jupyter auf der obersten Ebene ein Neptune-Ordner angezeigt. Wählen Sie ihn aus, um den Ordner zu öffnen.
3. Auf der nächsten Ebene befindet sich ein Ordner mit dem Namen 02-Visualization. Öffnen Sie diesen Ordner. In diesem Ordner befinden sich mehrere Notebooks, die Sie durch verschiedene Methoden für die Abfrage Ihrer Diagrammdaten in Gremlin und SPARQL sowie die Visualisierung der Abfrageergebnisse führen:

   - Air-Routes-Gremlin
   - Air-Routes-SPARQL

- Blog zur Workbench-Visualisierung
- EPL-Gremlin
- EPL-SPARQL

Wählen Sie ein Notebook aus, um mit den enthaltenen Abfragen zu experimentieren.

Behebung von Problemen mit dem Amazon Neptune Neptune-Notebook

Bei der Arbeit mit Neptune-Notebooks in können gelegentlich Probleme auftreten JupyterLab, die mit dem Browserverhalten JupyterLab selbst oder der Amazon SageMaker AI-Integration zusammenhängen. In den folgenden Abschnitten werden die häufigsten Probleme und deren Lösung beschrieben.

Desynchronisierung des Widget-Status

Graph-Notebook-Widgets können manchmal zwischen der Benutzeroberfläche und dem Kernel-Backend desynchronisiert werden. JupyterLab In diesem Fall werden möglicherweise Fehler bei der Anzeige von Widgets oder eine Fehlermeldung wie die folgende angezeigt:

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

Dies tritt in der Regel nach langen Leerlaufzeiten, beim Wechsel zwischen Notizbuch-Tabs, nach dem Löschen des Browser-Cache, bei instabiler Netzwerkverbindung oder beim erneuten Öffnen eines Notizbuchs mit ungespeicherten Widget-Ausgaben auf.

Um dieses Problem zu beheben, wählen Sie im Kernel-Menü „Kernel neu starten“ und „Alle Ausgaben löschen“ und führen Sie dann Ihre Notebook-Zellen erneut aus.

JupyterLab 4.x Verzögerung beim ersten Laden des Bildschirms

Wenn Sie ein Notizbuch nach dem JupyterLab Start zum ersten Mal öffnen, wird möglicherweise ein permanenter Ladebildschirm mit der folgenden Meldung angezeigt: `Loading... The loading screen is taking a long time. Would you like to clear the workspace or keep waiting?`

Dies betrifft hauptsächlich Firefox und tritt nur beim ersten geöffneten Notebook in einer Sitzung auf. Sie können das Problem lösen, indem Sie „Arbeitsbereich löschen“ wählen, die Browserseite aktualisieren oder das Notizbuch einfach erneut öffnen.

Wenn Sie Firefox verwenden und wiederholt auf dieses Problem stoßen, versuchen Sie, die Cookies und Seitendaten für Ihre SageMaker AI-Domain in den Datenschutz- und Sicherheitseinstellungen von Firefox zu löschen, und aktualisieren Sie dann die JupyterLab Seite.

Probleme beim Zugriff auf Graph Explorer

Nach dem Neustart des Notebooks oder wenn Sie zwischengespeicherte Anmeldeinformationen verwenden, können Sie Probleme beim Zugriff auf Graph Explorer haben. Dies kann sich als Authentifizierungsfehler, Meldungen mit verweigertem Zugriff oder als Fehler beim Laden von Graph Explorer äußern.

Um dieses Problem zu umgehen, versuchen Sie, Ihren Browser-Cache und Ihre Cookies zu löschen. Sie können auch zuerst Jupyter Classic öffnen JupyterLab , um eine Aktualisierung der Anmeldeinformationen zu erzwingen, und dann Graph Explorer öffnen. Wenn das Problem weiterhin besteht, versuchen Sie, ein Inkognito-Fenster oder ein privates Browserfenster zu verwenden.

Allgemeine Tipps

Viele Probleme mit Notizbüchern hängen mit dem Browser zusammen. Beachten Sie Folgendes, um Probleme zu minimieren:

- Leeren Sie regelmäßig Ihren Browser-Cache, wenn Sie mit Neptune-Notizbüchern arbeiten.
- Verwenden Sie immer die letzte Notizbuchsitzung, anstatt veraltete Sitzungen erneut zu öffnen.
- Verwenden Sie ein Inkognito-Fenster oder ein privates Browserfenster, wenn Sie auf anhaltende Probleme stoßen.
- Halten Sie Ihre Erweiterungen JupyterLab und die Graph-Notebook-Erweiterungen auf dem neuesten Stand.

Amazon Neptune einrichten

Amazon Neptune ist ein vollständig verwalteter Graphdatenbank-Service, der es einfach macht, Anwendungen zu erstellen und auszuführen, die mit stark verbundenen Datensätzen arbeiten. Um Ihnen den Einstieg in Neptune zu erleichtern, behandelt dieser Abschnitt der Dokumentation eine Vielzahl von Themen, von der Auswahl der richtigen Instance-Typen und Speicherkonfigurationen bis hin zur sicheren Verbindung mit Ihrem Neptune-Cluster, dem Laden von Daten und der Leistungsüberwachung. Die wichtigsten Bereiche, die in diesem Abschnitt behandelt werden, sind:

- Instanztypen: Die Dokumentation beschreibt die verschiedenen Instance-Typen, die für Neptune verfügbar sind, zusammen mit ihren Spezifikationen und Überlegungen, damit Sie die für Ihren Workload am besten geeignete Option auswählen können.
- Speichertypen: Es behandelt auch die verschiedenen Speichertypen, die von Neptune unterstützt werden, und gibt Hinweise zur Auswahl der richtigen Speicherkonfiguration.
- Clustererstellung: Die Dokumentation enthält schrittweise Anleitungen zum Erstellen eines neuen Neptune-Clusters, einschließlich der Konfiguration der erforderlichen VPC-Einstellungen.
- Konnektivität und Sicherheit: Es behandelt, wie Sie eine sichere Verbindung zu Ihrem Neptune-Cluster herstellen, den VPC-Zugriff einrichten und die erforderlichen Sicherheitsmaßnahmen zum Schutz Ihrer Daten konfigurieren.
- Laden von Daten: Die Dokumentation bietet Anleitungen zum Laden von Daten in Ihren Neptune-Cluster, einschließlich Optionen für das Massenladen und Streamen von Daten mithilfe verschiedener Tools und APIs.
- Überwachung und Fehlerbehebung: Es enthält auch Informationen zur Überwachung Ihrer Neptune-Umgebung, zum Zugriff auf Protokolle und zur Behebung häufiger Probleme, die während des Betriebs auftreten können.

> **Note**
>
> Referenzarchitekturen für AWS Graphdatenbanken und Referenzbereitstellungsarchitekturen finden Sie unter Amazon Neptune Resources. Diese Ressourcen können Sie dabei unterstützen, Entscheidungen hinsichtlich Graphdatenmodellen und Abfragesprachen zu treffen und Ihren Entwicklungsprozess zu beschleunigen.

Instance-Typen für Amazon Neptune auswählen

Amazon Neptune stellt verschiedene Instance-Größen und -Familien mit verschiedenen Funktionen bereit, die für verschiedene Graph-Workloads geeignet sind. Dieser Abschnitt soll Ihnen helfen, den besten Instance-Typ für Ihre Anforderungen auszuwählen.

Die Preise für die einzelnen Instance-Typen in diesen Familien finden Sie auf der Seite für Neptune-Preise.

Übersicht über die Zuteilung von Instance-Ressourcen

Die verschiedenen in Neptune verwendeten Amazon-EC2-Instance-Typen und -Größen stellen jeweils eine definierte Menge an Rechenleistung (vCPUs) und Systemspeicher bereit. Der Primärspeicher für Neptune befindet sich außerhalb der DB-Instances in einem Cluster, damit Rechenleistung und Speicherkapazität unabhängig voneinander skaliert werden können.

Dieser Abschnitt beschreibt die Skalierung von Rechenressourcen und die Unterschiede zwischen den einzelnen Instance-Familien.

In allen Instance-Familien werden vCPU-Ressourcen zur Unterstützung von zwei (2) Abfrageausführungs-Threads pro vCPU zugewiesen. Diese Unterstützung ist von der Instance-Größe abhängig. Bei der Festlegung der richtigen Größe einer Neptune-DB-Instance müssen Sie die mögliche Gleichzeitigkeit Ihrer Anwendung und die durchschnittliche Latenz Ihrer Abfragen berücksichtigen. Sie können die Anzahl der benötigten vCPUs wie folgt schätzen, wobei die Latenz als durchschnittliche Abfragelatenz in Sekunden und die Gleichzeitigkeit als gewünschte Anzahl von Abfragen pro Sekunde gemessen wird:

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

> **Note**
>
> SPARQL-Abfragen, openCypher-Abfragen und Gremlin-Leseabfragen, die die DFE-Abfrage-Engine verwenden, können unter bestimmten Umständen mehr als einen Ausführungsthread pro Abfrage verwenden. Sie sollten bei der anfänglichen Dimensionierung Ihres DB-Clusters annehmen, dass jede Abfrage einen einzelnen Ausführungsthread pro Ausführung nutzt, und dies hochskalieren, wenn Sie einen Gegendruck in Richtung der Abfragewarteschlange beobachten. Dies kann mithilfe der `/sparql/status` APIs/

`gremlin/status`, `/oc/status` oder oder beobachtet werden, oder es kann auch anhand der `MainRequestsPendingRequestsQueue` CloudWatch Metrik beobachtet werden.

Der Systemspeicher der einzelnen Instances ist in zwei primäre Zuteilungen aufgeteilt: den Pufferpool-Cache und den Thread-Speicher für die Abfrageausführung.

Ungefähr zwei Drittel des verfügbaren Speichers in einer Instanz sind für den Pufferpool-Cache reserviert. Buffer-pool Cache wird verwendet, um die zuletzt verwendeten Komponenten des Graphen zwischenzuspeichern, sodass auf Abfragen, die wiederholt auf diese Komponenten zugreifen, schneller zugegriffen werden kann. Instances mit einer größeren Menge an Systemspeicher verfügen über größere Pufferpool-Caches, die einen größeren Teil des Graphen lokal speichern können. Ein Benutzer kann die entsprechende Menge an Pufferpool-Cache einstellen, indem er die in verfügbaren Metriken zu Treffern und Fehlschlägen im Puffercache überwacht. CloudWatch

Sie sollten die Größe Ihrer Instance erhöhen, wenn die Cache-Trefferrate für einen konsistenten Zeitraum unter 99,9 % sinkt. Dies deutet darauf hin, dass der Pufferpool nicht groß genug ist und die Engine häufiger Daten aus dem zugrunde liegenden Speichervolume abrufen muss, als es effizient ist.

Das verbleibende Drittel des Systemspeichers wird gleichmäßig auf die Threads zur Abfrageausführung verteilt, wobei ein Teil des Speichers für das Betriebssystem und ein kleiner dynamischer Pool für Threads reserviert werden, die wie notwendig verwendet werden können. Der für jeden Thread verfügbare Arbeitsspeicher nimmt von einer Instance-Größe zur nächsten leicht zu, bis ein `8xl`-Instance-Typ erreicht ist, bei dem der pro Thread zugewiesene Arbeitsspeicher ein Maximum erreicht.

Sie müssen mehr Thread-Speicher hinzufügen, wenn `OutOfMemoryException` (OOM) auftritt. OOM-Ausnahmen treten auf, wenn ein Thread mehr als den ihm zugeteilten maximalen Speicher benötigt. (Das bedeutet nicht, dass der Arbeitsspeicher für die Instance insgesamt nicht ausreicht.)

Instance-Typen t3 und t4g

Die Instance-Familien `t4g` und `t3` stellen kostengünstige Optionen für den Einstieg in Graphdatenbanken sowie in die anfängliche Entwicklung und Testen dar. Diese Instances kommen für das kostenlose Neptune-Angebot in Frage, mit dem Neukunden Neptune in den ersten 750 Instance-Stunden kostenlos nutzen können, wenn sie innerhalb eines eigenständigen AWS Kontos

oder zusammengefasst unter einer AWS Organisation mit konsolidierter Abrechnung (Zahlerkonto) genutzt werden.

Die Instances `t4g` und `t3` werden nur für mittelgroße Konfigurationen (`t3.medium` und `t4g.medium`) angeboten.

Sie sind nicht für die Verwendung in Produktionsumgebung vorgesehen.

Da diese Instances nur über sehr begrenzte Ressourcen verfügen, werden sie nicht zum Testen der Ausführungszeiten von Abfragen oder der allgemeinen Datenbankleistung empfohlen. Wenn Sie die Abfrageleistung bewerten möchten, sollten Sie ein Upgrade auf eine andere Instance-Familie ausführen.

Instance-Typen der r4-Familie

VERALTET — Die `r4` Familie wurde bei der Markteinführung von Neptune im Jahr 2018 angeboten, aber jetzt bieten neuere Instance-Typen viel bessere Ergebnisse. price/performance Ab Engine-Version 1.1.0.0 unterstützt Neptune die `r4`-Instance-Typen nicht mehr.

R5-Familie von Instance-Typen

Die `r5`-Familie enthält arbeitsspeicheroptimierte Instance-Typen, die für die meisten Graph-Anwendungsfälle gut geeignet sind. Die `r5`-Familie enthält Instance-Typen von `r5.large` bis `r5.24xlarge`. Ihre Rechenleistung kann linear entsprechend Ihren Anforderungen skaliert werden. Beispielsweise hat eine `r5.xlarge`-Instance (4 vCPUs und 32 GiB Arbeitsspeicher) doppelt so viele vCPUs und doppelt so viel Arbeitsspeicher wie eine `r5.large`-Instance (2 vCPUs und 16 GiB Arbeitsspeicher). Eine `r5.2xlarge`-Instance (8 vCPUs und 64 GiB Arbeitsspeicher) hat doppelt so viele vCPUs und doppelt so viel Arbeitsspeicher wie eine `r5.xlarge`-Instance. Die Abfrageleistung wird direkt entsprechend der Rechenleistung bis zum Instance-Typ `r5.12xlarge` skaliert.

Die `r5`-Instance-Familie verfügt über eine Intel-CPU-Architektur mit 2 Sockets. Der Instance-Typ `r5.12xlarge` und kleinere Typen verwenden einen einzelnen Socket und den Systemspeicher, der diesem Einzel-Socket-Prozessor zugewiesen ist. Die Typen `r5.24xlarge` und `r5.16xlarge` verwenden beide Sockets und den verfügbaren Arbeitsspeicher. Da zwischen zwei physischen Prozessoren in einer 2-Socket-Architektur ein gewisser Speicherverwaltungs-Overhead erforderlich ist, sind die Leistungssteigerungen beim Hochskalieren des Instance-Typs `r5.12xlarge` auf `r5.16xlarge` oder `r5.24xlarge` nicht so linear wie beim Hochskalieren kleinerer Instance-Typen.

Familie von Instance-Typen der **R5D-Familie**

Neptune besitzt ein Lookup-Cache-Feature, mit der die Leistung von Abfragen verbessert werden kann, die eine große Anzahl von Eigenschaftswerten und Literalen abrufen und zurückgeben müssen. Dieses Feature wird vor allem von Kunden verwendet, deren Abfragen zahlreiche Attribute zurückgeben müssen. Der Lookup-Cache steigert die Leistung dieser Abfragen, indem er diese Attributwerte lokal abruft, statt jeden einzelnen Wert immer wieder im indizierten Neptune-Speicher nachzuschlagen.

Der Lookup-Cache wird mithilfe eines NVMe-attached EBS-Volumes auf einem `r5d` Instance-Typ implementiert. Er wird über die Parametergruppe eines Clusters aktiviert. Wenn Daten aus dem indizierten Neptune-Speicher abgerufen werden, werden Eigenschaftswerte und RDF-Literale in diesem NVMe-Volume zwischengespeichert.

Wenn Sie das Lookup-Cache-Feature nicht benötigen, sollten Sie den Standard-Instance-Typ `r5` anstelle von `r5d` verwenden, um die höheren Kosten für `r5d` zu vermeiden.

Die Instance-Typen der `r5d`-Familie haben die gleiche Größe wie die Instance-Typen der `r5`-Familie, von `r5d.large` bis `r5d.24xlarge`.

R6G-Familie von Instance-Typen

AWS hat einen eigenen ARM-based Prozessor namens Graviton entwickelt, der bessere Ergebnisse liefert price/performance als die Äquivalente von Intel und AMD. Die `r6g`-Familie verwendet den Graviton2-Prozessor. In unseren Tests bietet der Graviton2-Prozessor eine um 10 bis 20% bessere Leistung für OLTP-style (eingeschränkte) Grafikabfragen. Größere OLAP-ish Abfragen sind mit den Graviton2-Prozessoren jedoch möglicherweise etwas weniger leistungsfähig als mit Intel-Prozessoren, was auf die etwas geringere Leistung beim Speicher-Paging zurückzuführen ist.

Sie sollten auch beachten, dass die `r6g` -Familie über eine Single-Socket-Architektur verfügt. Das bedeutet, dass die Leistung linear mit der Rechenkapazität von von `r6g.large` zu `r6g.16xlarge` (dem größten Typ in der Familie) skaliert wird.

Instance-Typen der R6i-Familie

Amazon R6i-Instances nutzen Intel Xeon Scalable-Prozessoren der 3. Generation (Codename Ice Lake) und sind ideal für speicherintensive Workloads geeignet. In der Regel bieten sie ein um bis zu 15 % besseres Preis-Leistungs-Verhältnis und eine um bis zu 20 % höhere Speicherbandbreite pro vCPU als vergleichbare R5-Instance-Typen.

x2g-Familie von Instance-Typen

In einigen Graph-Anwendungsfällen ist die Leistung besser, wenn Instances über größere Pufferpool-Caches verfügen. Die x2g-Familie wurde eingeführt, um diese Anwendungsfälle besser zu unterstützen. Die x2g-Familie hat ein größeres Arbeitsspeicher-vCPU-Verhältnis als die r5- oder r6g-Familie. Die x2g-Instances nutzen ebenfalls den Graviton2-Prozessor und besitzen viele Leistungsmerkmale der r6g-Instance-Typen sowie einen größeren Pufferpool-Cache.

Wenn Sie r6g Instance-Typen mit geringer CPU-Auslastung und hoher Ausfallrate beim Pufferpool-Cache verwenden, versuchen Sie es stattdessen mit der x2g Familie. r5 Auf diese Weise erhalten Sie den zusätzlichen Arbeitsspeicher, den Sie benötigen, ohne für mehr CPU-Kapazität bezahlen zu müssen.

x2iezn-Familie von Instance-Typen

Die x2iezn Familie bietet speicheroptimierte Instances, die auf skalierbaren Intel Xeon Prozessoren mit Hochfrequenzleistung betrieben werden. Diese Instances bieten ein hohes Verhältnis von Arbeitsspeicher zu vCPU (32 GiB pro vCPU) und eignen sich daher gut für speicherintensive Graph-Workloads, die von einer hohen Single-Thread-Leistung profitieren.

Zu den Hauptmerkmalen gehören eine All-Core-Turbofrequenz von bis zu 4,5 GHz und die Verfügbarkeit in Größen von 2xlarge bis 12xlarge.

x2iedn-Familie von Instance-Typen

Die x2iedn Familie bietet speicheroptimierte Instances mit lokalem NVMe-SSD-Speicher. Diese Instances kombinieren eine hohe Speicherkapazität (32 GiB pro vCPU) mit schnellem lokalen Speicher und eignen sich daher ideal für Graph-Workloads, die sowohl von großen In-Memory-Caches als auch von leistungsfähigem lokalem Festplatten-Caching profitieren.

Diese Instances basieren auf skalierbaren Intel Xeon Prozessoren der dritten Generation und sind in Größen von xlarge bis 32xlarge erhältlich. Sie sind für große Graphdatenbanken optimiert, die sowohl Arbeitsspeicher als auch Speicherleistung erfordern.

Instance-Typen der R8G-Familie

Die r8g Familie umfasst speicheroptimierte Instance-Typen, die auf Graviton4-Prozessoren basieren. AWS Diese Instances bieten erhebliche Leistungsverbesserungen gegenüber früheren Generationen

und eignen sich daher gut für speicherintensive Graph-Workloads. Die R8G-Instances bieten im Vergleich zu R7G-Instances eine um etwa 15 bis 20% bessere Leistung für Graphabfragen.

Die `r8g` Familie verwendet eine Dual-Socket-Plattform. Instance-Typen von `r8g.large` bis `r8g.24xlarge` werden auf einem einzigen Socket ausgeführt, was bedeutet, dass die Leistung linear mit der Rechenkapazität in diesem Bereich skaliert. Der `r8g.48xlarge` verwendet beide Sockets und ist der größte Instance-Typ in der Familie. Wie bei anderen Dual-Socket-Familien sind die Leistungssteigerungen bei der Skalierung von `r8g.24xlarge` bis aufgrund des Socket-übergreifenden Speicherverwaltungsaufwands `r8g.48xlarge` möglicherweise nicht perfekt linear.

Zu den wichtigsten Merkmalen der `r8g` Produktreihe gehören:

- Angetrieben von AWS Graviton4-Prozessoren ARM-based
- Höhere Speicherbandbreite pro vCPU im Vergleich zu früheren Generationen
- Hervorragendes price/performance Verhältnis sowohl für OLTP-style (eingeschränkte) Grafikabfragen als auch für analytische Workloads OLAP-style
- Verbesserte Speicherverwaltungsfunktionen, die komplexen Graphendurchläufen zugute kommen

Die `r8g` Produktreihe eignet sich ideal für Produktionsworkloads, die eine hohe Speicherkapazität und gleichbleibende Leistung erfordern. Sie sind besonders effektiv für Anwendungen mit hohen Anforderungen an die Parallelität von Abfragen.

Instance-Typen der **`R7G-Familie`**

Die `r7g` Familie verwendet den AWS Graviton3-Prozessor, der bessere price/performance Ergebnisse liefert als frühere Instances. Graviton2-based In Tests bietet der Graviton3-Prozessor im Vergleich zu R6G-Instances eine um 25 bis 30% bessere Leistung für OLTP-style Grafikabfragen.

Wie die `r6g` Produktfamilie verfügt auch die `r7g` Familie über eine Single-Socket-Architektur, was bedeutet, dass die Leistung linear mit der Rechenkapazität von A `r7g.large` nach An `r7g.16xlarge` (dem größten Typ der Familie) skaliert.

Zu den wichtigsten Merkmalen der `r7g` Produktreihe gehören:

- Angetrieben von AWS Graviton3-Prozessoren ARM-based
- Verbesserte Speicher-Paging-Leistung im Vergleich zu R6G, wovon sowohl OLTP- als auch OLAP-Workloads profitieren

- Verbesserte Cache-Effizienz im Pufferpool
- Geringere Latenz für speicherintensive Operationen

Die r7g Produktreihe eignet sich gut für Produktionsumgebungen mit unterschiedlichen Abfragemustern und ist besonders effektiv für Workloads, die von einer verbesserten Speicherbandbreite profitieren.

Instance-Typen der **R7i-Familie**

Die r7i Familie basiert auf skalierbaren Intel Xeon Prozessoren der 4. Generation (Codename Sapphire Rapids) und bietet deutliche Verbesserungen gegenüber R6i-Instances. Diese Instances bieten eine um etwa 15% bessere Rechenleistung price/performance und eine bis zu 20% höhere Speicherbandbreite pro vCPU als vergleichbare R6i-Instance-Typen.

Die r7i Instance-Familie verfügt über eine Intel-CPU-Architektur mit 2 Sockeln, die der Familie ähnelt. r5 Der Instance-Typ r7i.12xlarge und kleinere Typen verwenden einen einzelnen Socket und den Systemspeicher, der diesem Einzel-Socket-Prozessor zugewiesen ist. Die Typen r7i.24xlarge und r7i.16xlarge verwenden beide Sockets und den verfügbaren Arbeitsspeicher. Da zwischen zwei physischen Prozessoren in einer 2-Socket-Architektur ein gewisser Speicherverwaltungs-Overhead erforderlich ist, sind die Leistungssteigerungen beim Hochskalieren des Instance-Typs r7i.12xlarge auf r7i.16xlarge oder r7i.24xlarge nicht so linear wie beim Hochskalieren kleinerer Instance-Typen.

Zu den wichtigsten Merkmalen der r7i Familie gehören:

- Angetrieben von skalierbaren Intel Xeon Prozessoren der 4. Generation
- Die Leistung skaliert linear mit einer Rechenkapazität von bis zu r7i.12xlarge
- Verbesserte Speicherverwaltung zwischen physischen Prozessoren in der 2-Sockel-Architektur
- Verbesserte Leistung für speicherintensive Grafikoperationen

Für all diese Instance-Familien können Sie die Anzahl der benötigten vCPUs anhand derselben Formel schätzen, die bereits erwähnt wurde:

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

Dabei wird die Latenz als durchschnittliche Abfragelatenz in Sekunden und die Parallelität als die Zielanzahl von Abfragen pro Sekunde gemessen.

Serverloser Instanztyp

Mittels des Features Neptune Serverless kann die Instance-Größe anhand der Ressourcenanforderungen eines Workloads dynamisch skaliert werden. Anstatt zu berechnen, wie viele vCPUs für Ihre Anwendung benötigt werden, können Sie mit Neptune Serverless die unteren und oberen Rechenkapazitätsgrenzwerte (gemessen in Neptune-Kapazitätseinheiten) für die Instances in Ihrem DB-Cluster festlegen. Workloads mit schwankender Nutzung können hinsichtlich der Kosten optimiert werden, indem statt bereitgestellter Instances Serverless-Instances verwendet werden.

Sie können sowohl bereitgestellte als auch Serverless-Instances im selben DB-Cluster einrichten, um ein optimales Kosten-Leistungs-Verhältnis zu erzielen.

Speichertypen für Amazon Neptune auswählen

Neptune bietet zwei Speichertypen mit unterschiedlichen Preismodellen an:

- Standardspeicher — Standardspeicher bietet kostengünstigen Datenbankspeicher für Anwendungen mit mäßiger bis geringer I/O Auslastung.
- I/O—Optimierter Speicher — Mit I/O —Optimized Storage, verfügbar ab Engine-Version 1.3.0.0, zahlen Sie nur für den Speicher und die Instanzen, die Sie tatsächlich nutzen. Die Speicherkosten sind höher als bei Standardspeicher, und auch die Instanzkosten sind höher als bei Standardinstanzen. Sie zahlen nichts für I/O das, was Sie verwenden. Wenn Ihre I/O Nutzung hoch ist, kann bereitgestellter IOps-Speicher die Kosten erheblich senken.

  I/O—Optimierter Speicher ist darauf ausgelegt, die Anforderungen I/O intensiver Graph-Workloads zu vorhersehbaren Kosten zu erfüllen. Sie können nur einmal alle 30 Tage zwischen den Speichertypen I/O —Optimiert und Standard wechseln.

  Preisinformationen zu I/O —Optimized Storage finden Sie auf der Neptune-Preisseite. Im folgenden Abschnitt wird beschrieben, wie Sie I/O —Optimized Storage für einen Neptune-DB-Cluster einrichten.

Auswahl I/O —Optimierter Speicher für einen Neptune-DB-Cluster

Standardmäßig verwenden Neptune-DB-Cluster Standardspeicher. Sie können I/O —Optimized Storage auf einem DB-Cluster bei der Erstellung wie folgt aktivieren:

Hier ist ein Beispiel dafür, wie Sie I/O —Optimized Storage aktivieren können, wenn Sie einen Cluster mit dem erstellen: 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
```

Anschließend ist für jede Instanz, die Sie automatisch erstellen, I/O —Optimized storage aktiviert:

```
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
```

Sie können auch einen vorhandenen DB-Cluster ändern, um I/O —Optimized Storage darauf zu aktivieren. Gehen Sie dazu wie folgt vor:

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

Sie können einen Backup-Snapshot in einem DB-Cluster wiederherstellen, wenn I/O —Optimized storage aktiviert ist:

```
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
```

Sie können mit einem beliebigen `describe-` Aufruf feststellen, ob ein Cluster I/O —Optimized Storage verwendet. Wenn I/O —Optimized storage aktiviert ist, gibt der Aufruf ein Speichertyp-Feld zurück, das auf gesetzt ist. `iop1`

Einen Amazon Neptune Neptune-Cluster erstellen

Der einfachste Weg, einen neuen Amazon Neptune DB-Cluster zu erstellen, besteht darin, eine CloudFormation Vorlage zu verwenden, die alle erforderlichen Ressourcen für Sie erstellt, ohne dass Sie alles von Hand erledigen müssen. Die CloudFormation Vorlage übernimmt einen Großteil der Einrichtung für Sie, einschließlich der Erstellung einer Amazon Elastic Compute Cloud (Amazon EC2) -Instance:

Um einen neuen Neptune-DB-Cluster mit einem zu starten CloudFormation Vorlage

1. Erstellen Sie einen neuen IAM-Benutzer mit den Berechtigungen, die für die Arbeit mit dem Neptune-DB-Cluster benötigt werden, wie in IAM-Benutzerberechtigungen beschrieben.
2. Richten Sie zusätzliche Voraussetzungen ein, die für die Verwendung der CloudFormation Vorlage erforderlich sind, wie unter beschrieben. Voraussetzungen für die Einrichtung von Amazon Neptune mit AWS CloudFormation
3. Rufen Sie den CloudFormation Stack auf, wie unter beschriebenErstellen eines Amazon Neptune Neptune-Clusters mit AWS CloudFormation.

Sie können auch eine globale Neptune-Datenbank erstellen, die sich über mehrere erstreckt. Dies ermöglicht globale Lesevorgänge mit geringer Latenz und ermöglicht eine schnelle Wiederherstellung in den seltenen Fällen AWS-Regionen, in denen sich ein Ausfall auf ein ganzes System auswirkt. AWS-Region

Informationen zur manuellen Erstellung eines Amazon Neptune Neptune-Clusters mithilfe von finden Sie unter AWS-Managementkonsole. Starten eines Neptune-DB-Clusters mit dem AWS-Managementkonsole

Sie können auch eine CloudFormation Vorlage verwenden, um eine Lambda-Funktion zur Verwendung mit Neptune zu erstellen (siehe). Verwenden CloudFormation um eine Lambda-Funktion zur Verwendung in Neptune zu erstellen

Informationen zur Verwaltung von Clustern und Instances in Neptune finden Sie in Verwalten Ihrer Amazon-Neptune-Datenbank.

Voraussetzungen für die Einrichtung von Amazon Neptune mit AWS CloudFormation

Bevor Sie einen Amazon Neptune Neptune-Cluster mithilfe einer CloudFormation Vorlage erstellen, benötigen Sie Folgendes:

- Ein Amazon-EC2-Schlüsselpaar.
- Die für die Verwendung erforderlichen Berechtigungen. CloudFormation

Erstellen Sie ein Amazon EC2 EC2-Schlüsselpaar für den Start eines Neptune-Clusters mit CloudFormation

Um einen Neptune-DB-Cluster mithilfe einer CloudFormation Vorlage zu starten, muss in der Region, in der Sie den Stack erstellen, ein Amazon EC2-Schlüsselpaar (und die zugehörige PEM-Datei) verfügbar sein. CloudFormation

Wenn Sie das key pair erstellen müssen, finden Sie Anweisungen entweder unter Creating a Key Pair Using Amazon EC2 im Amazon EC2-Benutzerhandbuch oder Creating a Key Pair Using Amazon EC2 im Amazon EC2-Benutzerhandbuch.

Fügen Sie IAM-Richtlinien hinzu, um die für die Verwendung des erforderlichen Berechtigungen zu gewähren CloudFormation Vorlage

Zunächst müssen Sie einen IAM-Benutzer mit den für die Arbeit mit Neptune erforderlichen Berechtigungen einrichten, wie unter Erstellen eines IAM-Benutzers mit Berechtigungen für Neptunebeschrieben.

Anschließend müssen Sie die AWS verwaltete Richtlinie,`AWSCloudFormationReadOnlyAccess`, zu diesem Benutzer hinzufügen.

Schließlich müssen Sie die folgende, vom Kunden verwaltete Richtlinie erstellen und sie diesem Benutzer hinzufügen:

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**
>
> Die folgenden Berechtigungen sind nur erforderlich, um einen Stack zu löschen: `iam:DeleteRole`, `iam:RemoveRoleFromInstanceProfile`, `iam:DeleteRolePolicy`, `iam:DeleteInstanceProfile` und `ec2:DeleteVpcEndpoints`.
> Beachten Sie auch, dass `ec2:*Vpc` `ec2:DeleteVpc`-Berechtigungen erteilt.

Erstellen eines Amazon Neptune Neptune-Clusters mit AWS CloudFormation

Sie können eine CloudFormation Vorlage verwenden, um einen Neptune-DB-Cluster einzurichten.

1. Um den CloudFormation Stack auf der CloudFormation Konsole zu starten, wählen Sie eine der Schaltflächen Stack starten in der folgenden Tabelle.

Region	Anzeigen	In Designer anzeigen	Starten
USA Ost (Nord-Virginia)	Anzeigen	In Designer anzeigen	
USA Ost (Ohio)	Anzeigen	In Designer anzeigen	
USA West (Nordkalifornien)	Anzeigen	In Designer anzeigen	
USA West (Oregon)	Anzeigen	In Designer anzeigen	
Kanada (Zentral)	Anzeigen	In Designer anzeigen	
Kanada West (Calgary)	Anzeigen	In Designer anzeigen	
Südamerika (São Paulo)	Anzeigen	In Designer anzeigen	
Europa (Stockholm)	Anzeigen	In Designer anzeigen	
Europa (Irland)	Anzeigen	In Designer anzeigen	
Europa (London)	Anzeigen	In Designer anzeigen	
Europa (Paris)	Anzeigen	In Designer anzeigen	

Region	Anzeigen	In Designer anzeigen	Starten
Europa (Spain)	Anzeigen	In Designer anzeigen	
Europa (Frankfurt)	Anzeigen	In Designer anzeigen	
Europa (Zürich)	Anzeigen	In Designer anzeigen	
Middle East (Bahrain)	Anzeigen	In Designer anzeigen	
Naher Osten (VAE)	Anzeigen	In Designer anzeigen	
Israel (Tel Aviv)	Anzeigen	In Designer anzeigen	
Afrika (Kapstadt)	Anzeigen	In Designer anzeigen	
Asien-Pazifik (Hongkong)	Anzeigen	In Designer anzeigen	
Asien-Pazifik (Tokio)	Anzeigen	In Designer anzeigen	
Asien-Pazifik (Seoul)	Anzeigen	In Designer anzeigen	
Asien-Pazifik (Singapur)	Anzeigen	In Designer anzeigen	
Asien-Pazifik (Sydney)	Anzeigen	In Designer anzeigen	
Asien-Pazifik (Jakarta)	Anzeigen	In Designer anzeigen	
Asien-Pazifik (Melbourne)	Anzeigen	In Designer anzeigen	

Region	Anzeigen	In Designer anzeigen	Starten
Asien-Pazifik (Malaysia)	Anzeigen	In Designer anzeigen	
Asien-Pazifik (Mumbai)	Anzeigen	In Designer anzeigen	
Asien-Pazifik (Hyderabad)	Anzeigen	In Designer anzeigen	
China (Peking)	Anzeigen	In Designer anzeigen	
China (Ningxia)	Anzeigen	In Designer anzeigen	
AWS GovCloud (US-West)	Anzeigen	In Designer anzeigen	
AWS GovCloud (US-East)	Anzeigen	In Designer anzeigen	

2. Wählen Sie auf der Seite Vorlage auswählen die Option Weiter aus.
3. Wählen Sie auf der Seite „Details angeben“ ein key pair für EC2SSHKeyPairName.

   Dieses Schlüsselpaar ist erforderlich, um auf die EC2-Instance zuzugreifen. Stellen Sie sicher, dass Sie über die PEM-Datei für das Schlüsselpaar verfügen, das Sie auswählen.
4. Wählen Sie Weiter aus.
5. Wählen Sie auf der Seite Optionen Weiter aus.
6. Aktivieren Sie auf der Seite Überprüfen das erste Kontrollkästchen, um zu bestätigen, dass CloudFormation IAM-Ressourcen erstellt. Aktivieren Sie das zweite Kontrollkästchen, um `CAPABILITY_AUTO_EXPAND` für den neuen Stack zu bestätigen.

> **Note**
>
> `CAPABILITY_AUTO_EXPAND` bestätigt explizit, dass Makros ohne vorherige Überprüfung beim Erstellen des Stacks erweitert werden. Benutzer erstellen häufig einen Änderungssatz aus einer verarbeiteten Vorlage, sodass die von Makros durchgeführten

Änderungen überprüft werden können, ehe der Stack tatsächlich erstellt wird. Weitere Informationen finden Sie in der CloudFormation CreateStack-API.

Wählen Sie die Option Erstellen aus.

> **Note**
>
> Sie können Ihre CloudFormation Vorlage auch verwenden, um die Engine-Version Ihres DB-Clusters zu aktualisieren.

Die Amazon-VPC einrichten, in der sich Ihr Amazon-Neptune-DB-Cluster befindet

Ein Amazon-Neptune-DB-Cluster kann nur in einer Amazon Virtual Private Cloud (Amazon VPC) erstellt werden. Ihre Endpunkte sind innerhalb dieser VPC zugänglich, und wenn öffentliche Neptune-Endpunkte aktiviert sind, können sie auch außerhalb der VPC und über das Internet aufgerufen werden.

Es gibt verschiedene Möglichkeiten, die VPC einzurichten, je nachdem, wie Sie auf den DB-Cluster zugreifen möchten.

Dies sind einige Punkte, die Sie bei der Konfiguration der VPC beachten sollten, in der sich Ihr Neptune-DB-Cluster befindet:

- Ihre VPC muss mindestens zwei Subnetze besitzen. Diese Subnetze müssen sich in unterschiedlichen Availability Zones (AZs) befinden. Mit der Verteilung Ihrer Cluster-Instances über mindestens zwei AZs stellt Neptune sicher, dass im DB-Cluster auch im unwahrscheinlichen Fall eines Availability-Zone-Ausfalls stets Instances verfügbar sind. Das Cluster-Volume für Ihren Neptune-DB-Cluster umfasst immer drei AZs, um einen dauerhaften Speicher mit extrem geringer Wahrscheinlichkeit von Datenverlusten bereitzustellen.
- Die CIDR-Blöcke in den einzelnen Subnetzen müssen groß genug sein, um IP-Adressen bereitzustellen, die Neptune während Wartung, Failover und Skalierung benötigt.
- Die VPC muss über eine DB-Subnetzgruppe mit den Subnetzen verfügen, die Sie erstellt haben. Neptune wählt in der Subnetzgruppe ein Subnetz und in diesem Subnetz eine IP-Adresse aus, um

sie mit den einzelnen DB-Instances im DB-Cluster zu verknüpfen. Die DB-Instance befindet sich anschließend in derselben AZ wie das Subnetz.

- Die VPC sollte DNS-aktiviert sein (DNS-Hostnamen und DNS-Auflösung).
- Die VPC muss eine VPC-Sicherheitsgruppe besitzen, die den Zugriff auf Ihren DB-Cluster zulässt.
- Die Tenancy in einer Neptune-VPC sollte auf Standard festgelegt sein.

Hinzufügen von Subnetzen zur VPC, in der sich Ihr Neptune-DB-Cluster befindet

Ein Subnetz ist ein Bereich von IP-Adressen in Ihrer VPC. Sie können Ressourcen wie einen Neptune-DB-Cluster, oder eine EC2-Instance in einem Subnetz starten. Wenn Sie ein Subnetz erstellen, legen Sie den IPv4-CIDR-Block für das Subnetz fest. Dies ist ein Subnetz des VPC-CIDR-Blocks. Jedes Subnetz muss sich vollständig innerhalb einer einzigen Availability Zone (AZ) befinden und darf nicht mehrere Zonen umfassen. Durch den Start von Instances in unterschiedlichen Availability Zones können Sie Ihre Anwendungen vor Ausfällen in einer Zone schützen. Weitere Informationen finden Sie in der VPC-Subnetz-Dokumentation.

Ein Neptune-DB-Cluster erfordert mindestens zwei VPC-Subnetze.

So fügen Sie einer VPC Subnetze hinzu

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die Amazon VPC-Konsole unter https://console.aws.amazon.com/vpc/.
2. Wählen Sie im Navigationsbereich Subnetze aus.
3. Wählen Sie im VPC-Dashboard zunächst Subnetze und dann Subnetz erstellen aus.
4. Wählen Sie auf der Seite Subnetz erstellen die VPC aus, in der Sie das Subnetz erstellen möchten.
5. Führen Sie unter Subnetzeinstellungen Folgendes aus:

    a. Geben Sie in Subnetzname einen Namen für das neue Subnetz ein.

    b. Wählen Sie eine Availability Zone (AZ) für das Subnetz aus oder lassen Sie die Einstellung Keine Präferenz unverändert.

    c. Geben Sie in IPv4-CIDR-Block den IP-Adressblock des Subnetzes ein.

    d. Fügen Sie dem Subnetz Tags hinzu, wenn notwendig.

    e. Klicken Sie auf

6. Wenn Sie gleichzeitig ein weiteres Subnetz erstellen möchten, wählen Sie Neues Subnetz hinzufügen aus.
7. Wählen Sie Subnetz erstellen, um das oder die neuen Subnetze zu erstellen.

Konfiguration von VPC in Amazon Neptune

Erstellen Sie eine Subnetzgruppe.

So erstellen Sie eine Neptune-Subnetzgruppe

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die Amazon Neptune Neptune-Konsole unter. https://console.aws.amazon.com/neptune/home
2. Wählen Sie Subnetzgruppen und DB-Subnetzgruppe erstellen aus.
3. Geben Sie einen Namen und eine Beschreibung für die neue Subnetzgruppe ein (die Beschreibung ist erforderlich).
4. Wählen Sie in VPC die VPC aus, in der sich diese Subnetzgruppe befinden soll.
5. Wählen Sie in Availability Zone die AZ aus, in der sich diese Subnetzgruppe befinden soll.
6. Fügen Sie in Subnetz ein oder mehrere der Subnetze in dieser AZ zu dieser Subnetzgruppe hinzu.
7. Wählen Sie Erstellen aus, um die neue Subnetzgruppe zu erstellen.

Eine Sicherheitsgruppe über die Konsole erstellen

Sicherheitsgruppen ermöglichen den Zugriff auf Ihren Neptune-DB-Cluster in der VPC. Sie dienen als Firewall für den zugeordneten DB-Cluster und steuern den ein- und ausgehenden Datenverkehr auf Instance-Ebene. Eine DB-Instance wird standardmäßig mit einer Firewall und einer Standard-Sicherheitsgruppe erstellt, die den Zugriff auf den DB-Cluster verhindert. Um den Zugriff zu aktivieren, benötigen Sie eine VPC-Sicherheitsgruppe mit zusätzlichen Regeln.

Das folgende Verfahren zeigt, wie Sie eine benutzerdefinierte TCP-Regel zur Angabe des Port-Bereichs und der IP-Adressen hinzufügen, den/die die Amazon-EC2-Instance für den Zugriff auf Ihren Neptune-DB-Cluster verwenden soll. Sie können statt der IP-Adresse auch die VPC-Sicherheitsgruppe verwenden, die der EC2-Instance zugewiesen ist.

So erstellen Sie eine VPC-Sicherheitsgruppe für Neptune in der Konsole

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die Amazon VPC-Konsole unter https://console.aws.amazon.com/vpc/.
2. Wählen Sie in der oberen rechten Ecke der Konsole die AWS Region aus, in der Sie eine VPC-Sicherheitsgruppe für Neptune erstellen möchten. Die Liste der Amazon VPC-Ressourcen für diese Region sollte zeigen, dass Sie über mindestens eine VPC und mehrere Subnetze verfügen. Andernfalls verfügen Sie in dieser Region nicht über eine Standard-VPC.
3. Wählen Sie auf der Navigationsleiste unter Sicherheit die Option Sicherheitsgruppen aus.
4. Wählen Sie Sicherheitsgruppe erstellen aus. Geben im Fenster Sicherheitsgruppe erstellen den Namen der Sicherheitsgruppe, eine Beschreibung und die ID der VPC ein, in der sich Ihr Neptune-DB-Cluster befinden wird.
5. Fügen Sie der Sicherheitsgruppe einer Amazon-EC2-Instance, die Sie mit Ihrem Neptune-DB-Cluster verbinden möchten, eine Regel für eingehenden Datenverkehr hinzu:
   a. Wählen Sie im Bereich Regeln für eingehenden Datenverkehr die Option Regel hinzufügen aus.
   b. Lassen Sie in der Liste Typ die Option Benutzerdefiniertes TCP ausgewählt.
   c. Geben Sie im Textfeld Portbereich den Wert 8182ein. Dies ist der Standard-Portwert für Neptune.
   d. Geben Sie unter Quelle den IP-Adressbereich (CIDR-Wert) ein, von dem aus Sie auf Neptune zugreifen werden, oder wählen Sie den Namen einer vorhandenen Sicherheitsgruppe aus.
   e. Wenn Sie weitere IP-Adressen oder andere Portbereiche hinzufügen müssen, wählen Sie erneut Regel hinzufügen aus.
6. Im Bereich für Regeln für ausgehenden Datenverkehr können Sie auch eine oder mehrere Regeln für ausgehenden Datenverkehr hinzufügen, wenn notwendig.
7. Wenn Sie fertig sind, wählen Sie Sicherheitsgruppe erstellen aus.

Sie können diese neue VPC-Sicherheitsgruppe verwenden, wenn Sie einen neuen Neptune-DB-Cluster erstellen.

Wenn Sie eine Standard-VPC verwenden, wurde bereits eine Standard-Subnetzgruppe für Sie erstellt, die alle Subnetze der VPC umfasst. Wenn Sie in der Neptune-Konsole Datenbank erstellen auswählen, wird die Standard-VPC verwendet, wenn Sie keine andere VPC angeben.

Den DNS-Support in Ihrer VPC sicherstellen

Domain Name System (DNS) ist ein Standard, nach dem Namen, die im Internet verwendet werden, entsprechend der zugehörigen IP-Adressen aufgelöst werden. Ein DNS-Hostname benennt einen Computer eindeutig und besteht aus einem Hostnamen und einem Domainnamen. DNS-Server lösen DNS-Hostnamen zu den entsprechenden IP-Adressen auf.

Stellen Sie sicher, dass sowohl DNS-Hostnamen als auch die DNS-Auflösung in Ihrer VPC aktiviert sind. Die VPC-Netzwerkattribute `enableDnsHostnames` und `enableDnsSupport` müssen auf `true` gesetzt werden. Zum Anzeigen und Ändern dieser Attribute wechseln Sie über https://console.aws.amazon.com/vpc/ zur VPC-Konsole.

Weitere Informationen finden Sie unter Verwenden von DNS in Ihrer VPC.

> **Note**
>
> Stellen Sie bei Verwendung von Route 53 sicher, dass Ihre Konfiguration keine DNS-Netzwerkattribute in Ihrer VPC überschreibt.

Verbindung zu einem Amazon Neptune Neptune-Cluster herstellen

Nachdem Sie einen Neptune-Cluster erstellt haben, müssen Sie den Netzwerkzugriff einrichten, damit Ihre Anwendungen den Cluster-Endpunkt erreichen können. In den folgenden Abschnitten werden die Optionen für die Netzwerkkonnektivität beschrieben. Weitere Informationen zum Abfragen Ihrer Daten nach dem Herstellen der Verbindung finden Sie unterZugreifen auf Grafikdaten.

Optionen für die Netzwerkkonnektivität

Ein Amazon-Neptune-DB-Cluster kann nur in einer Amazon Virtual Private Cloud (Amazon VPC) erstellt werden. Ihre Endpunkte sind nur innerhalb dieser VPC zugänglich, es sei denn, Sie aktivieren und richten öffentliche Neptune-Endpunkte für den DB-Cluster ein.

Sie können den Netzwerkzugriff auf Ihren Neptune-DB-Cluster in seiner VPC auf verschiedene Arten einrichten:

- Herstellen einer Verbindung von einer Amazon-EC2-Instance in derselben VPC
- Herstellen einer Verbindung von einer Amazon-EC2-Instance in einer anderen VPC

- Herstellen einer Verbindung von einem privaten Netzwerk
- Verbindung von einem öffentlichen Endpunkt aus herstellen

Überprüfen Sie Ihre Verbindung

Nachdem Sie den Netzwerkzugriff eingerichtet haben, können Sie überprüfen, ob Ihre Verbindung funktioniert, indem Sie den Instanzstatus-Endpunkt aufrufen. Eine erfolgreiche Antwort bestätigt, dass Ihr Client den Neptun-Cluster erreichen kann.

AWS CLI

Führen Sie den folgenden Befehl aus:

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

Weitere Informationen finden Sie unter get-engine-status in der Befehlsreferenz. 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**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

Weitere Informationen zur Verwendung awscurl mit der IAM-Authentifizierung finden Sie unter. Verwenden von awscurl mit temporären Anmeldeinformationen, um eine sichere Verbindung zu einem DB-Cluster mit aktivierter IAM-Authentifizierung herzustellen

curl

Führen Sie den folgenden Befehl aus:

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

> **Note**
>
> Dieser Befehl funktioniert nur, wenn die IAM-Authentifizierung auf Ihrem Cluster nicht aktiviert ist. Wenn die IAM-Authentifizierung aktiviert ist, verwenden Sie stattdessen awscurl oder. AWS CLI

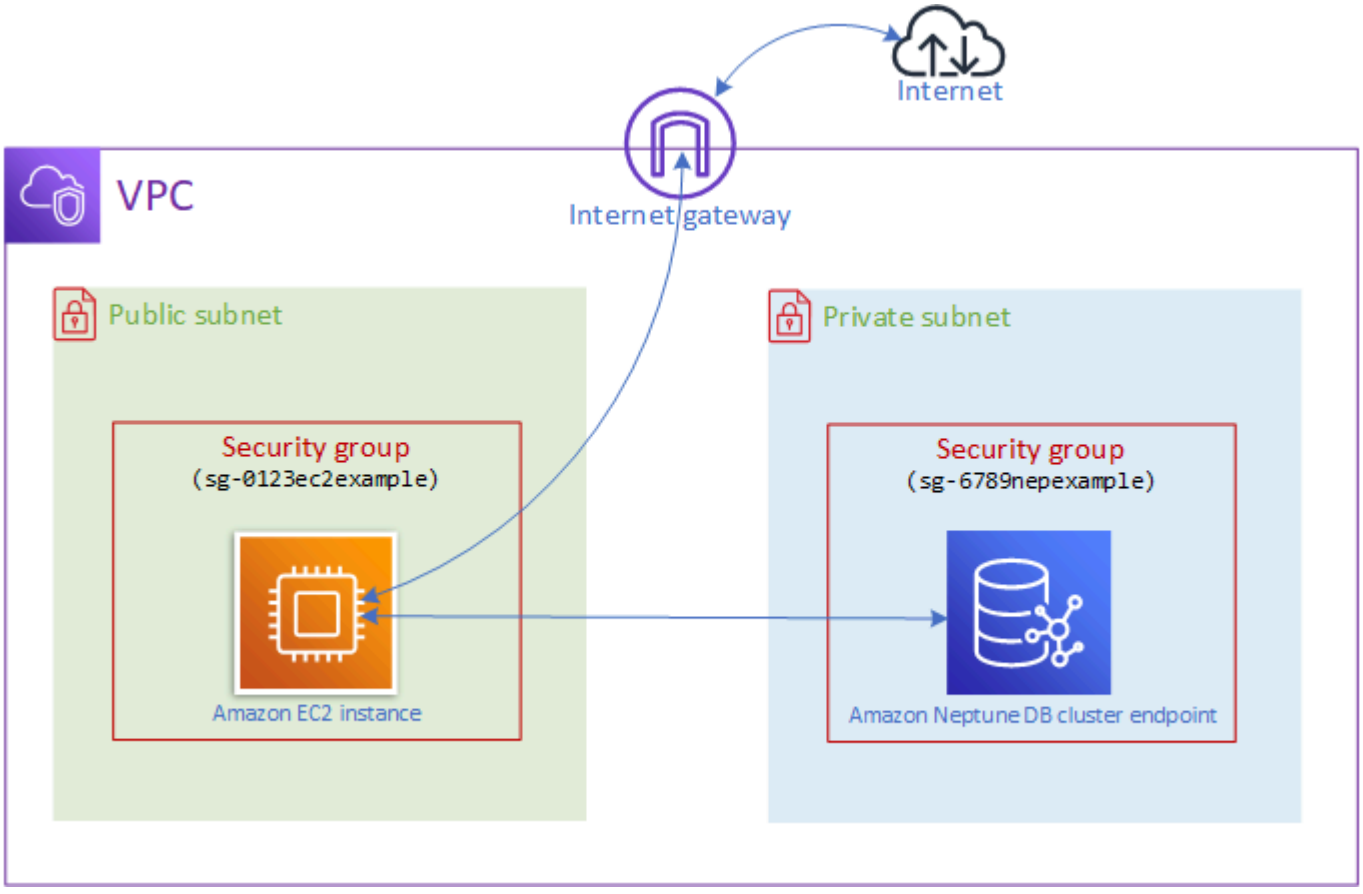
Ein fehlerfreier Cluster gibt eine JSON-Antwort zurück, die Folgendes beinhaltet: `"status": "healthy"` Beispiel:

```
{
    "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": { ... }
}
```

Wenn Sie den Endpunkt nicht erreichen können, stellen Sie sicher, dass Ihre Netzwerkkonfiguration Datenverkehr auf Port 8182 (oder Ihrem konfigurierten Port) zulässt und dass Ihre Sicherheitsgruppenregeln eingehende Verbindungen von Ihrem Client zulassen. Weitere Informationen finden Sie unter Sicherung des Zugangs zu Neptune. Hilfe beim Einrichten der AWS CLI SDKs oder `curl` finden Sie unter. Verwenden von Befehlszeilentools

Eine Amazon EC2 EC2-Instance mit dem Amazon Neptune Neptune-Cluster in derselben VPC verbinden

Eine der häufigsten Methoden für das Herstellen von Verbindungen mit einer Neptune-Datenbank ist die Verbindung von einer Amazon-EC2-Instance, die sich in derselben VPC wie Ihr Neptune-DB-Cluster befindet. Auf der EC2-Instance könnte beispielsweise ein Webserver ausgeführt werden, der eine Verbindung zum Internet herstellt. In diesem Fall hat nur die EC2-Instance Zugriff auf den Neptune-DB-Cluster, während das Internet lediglich Zugriff auf die EC2-Instance hat:



Um diese Konfiguration zu aktivieren, müssen die richtigen VPC-Sicherheitsgruppen und Subnetzgruppen eingerichtet worden sein. Der Webserver wird in einem öffentlichen Subnetz gehostet, damit er im öffentlichen Internet erreicht werden kann. Die Neptune-Cluster-Instance wird

aus Sicherheitsgründen in einem privaten Subnetz gehostet. Siehe Die Amazon-VPC einrichten, in der sich Ihr Amazon-Neptune-DB-Cluster befindet.

Damit die Amazon–EC2-Instance eine Verbindung mit Ihrem Neptune-Endpunkt herstellen kann, beispielsweise an Port 8182, müssen Sie eine Sicherheitsgruppe einrichten. Wenn Ihre Amazon–EC2-Instance eine Sicherheitsgruppe mit dem Namen `ec2-sg1` verwendet, müssen Sie eine andere Amazon–EC2-Sicherheitsgruppe erstellen (z. B. `db-sg1`), die Regeln für den eingehenden Datenverkehr für Port 8182 und `ec2-sg1` als Quelle enthält. Fügen Sie anschließend `db-sg1` zu Ihrem Neptune-Cluster hinzu, um die Verbindung zuzulassen.

Nach der Erstellung der Amazon-EC2-Instance können Sie sich mit SSH bei ihr anmelden und eine Verbindung zu Ihrem Neptune-DB-Cluster herstellen. Informationen zum Herstellen einer Verbindung mit einer EC2-Instance mithilfe von SSH finden Sie unter Connect to your Linux Instance im Amazon EC2 EC2-Benutzerhandbuch.

Wenn Sie eine Linux- oder macOS-Befehlszeile verwenden, um eine Verbindung zur EC2-Instance herzustellen, können Sie den SSH-Befehl aus dem SSHAccess-Element in den Outputs-Abschnitt des Stacks einfügen. CloudFormation Hierfür muss sich die PEM-Datei im aktuellen Verzeichnis befinden und die PEM-Dateiberechtigungen müssen auf 400 (`chmod 400 keypair.pem`) festgelegt werden.

So erstellen Sie eine VPC mit privaten und öffentlichen Subnetzen

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die Amazon VPC-Konsole unter https://console.aws.amazon.com/vpc/.
2. Wählen Sie in der oberen rechten Ecke von die Region aus AWS-Managementkonsole, in der Sie Ihre VPC erstellen möchten.
3. Wählen Sie im VPC-Dashboard die Option VPC-Assistent starten aus.
4. Füllen Sie den Bereich VPC-Einstellungen auf der Seite VPC erstellen aus:
    a. Wählen Sie unter Zu erstellende Ressourcen die Option VPC, Subnetze usw. aus.
    b. Lassen Sie das Standard-Namensschild unverändert oder geben Sie einen Namen Ihrer Wahl ein oder deaktivieren Sie das Auto-generateKontrollkästchen, um die Generierung von Namensschildern zu deaktivieren.
    c. Übernehmen Sie den IPv4-CIDR-Blockwert als `10.0.0.0/16`.
    d. Übernehmen Sie den IPv6-CIDR-Blockwert als Kein IPv6-CIDR-Block.
    e. Übernehmen Sie für Tenancy den Wert Standard.

f. Übernehmen Sie für die Anzahl der Availability Zones (AZs) den Wert 2.

g. Übernehmen Sie für NAT-Gateways ($) den Wert Keine, es sei denn, Sie benötigen ein oder mehrere NAT-Gateways.

h. Legen Sie VPC-Endpunkte auf Keine fest, es sei denn, Sie werden Amazon S3 verwenden.

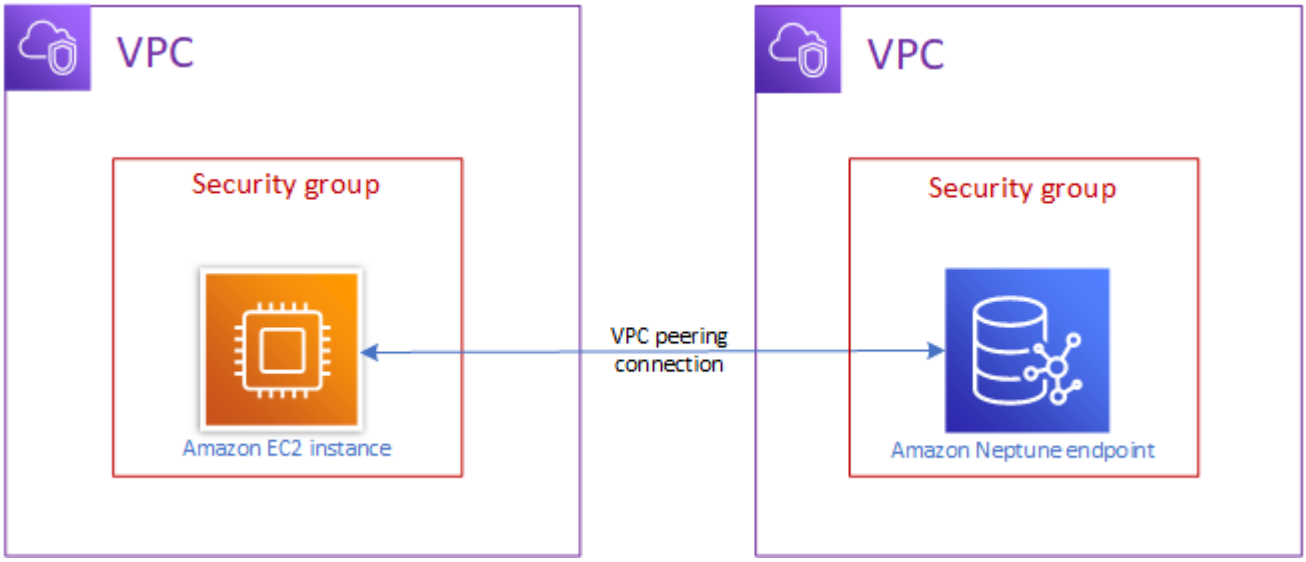
i. Sowohl DNS-Hostnamen aktivieren als auch DNS-Auflösung aktivieren sollten markiert sein.

5. Wählen Sie VPC erstellen aus.

Eine Amazon EC2 EC2-Instance mit einem Amazon Neptune Neptune-Cluster in einer anderen VPC verbinden

Ein Amazon Neptune Neptune-DB-Cluster kann nur in einer Amazon Virtual Private Cloud (Amazon VPC) erstellt werden, und seine Endpunkte sind innerhalb dieser VPC zugänglich, normalerweise von einer Amazon Elastic Compute Cloud (Amazon EC2) -Instance aus, die in dieser VPC ausgeführt wird. Alternativ kann über einen öffentlichen Endpunkt darauf zugegriffen werden. Weitere Informationen zu öffentlichen Endpunkten finden Sie unter Öffentliche Endpunkte von Neptune.

Wenn sich Ihr DB-Cluster in einer anderen VPC als die für den Zugriff verwendete EC2-Instance befindet, können Sie die Verbindung mittels VPC-Peering herstellen:



Eine VPC-Peering-Verbindung ist eine Netzwerkverbindung zwischen zwei VPCs, die den Datenverkehr zwischen ihnen privat leitet, sodass die Instances in den beiden VPCs miteinander kommunizieren können, als befänden sie sich im selben Netzwerk. Sie können eine VPC-Peering-Verbindung zwischen VPCs in Ihrem Konto, zwischen einer VPC in Ihrem AWS Konto und einer VPC in einem anderen Konto oder mit einer VPC in einer anderen AWS Region herstellen. AWS

AWS verwendet die bestehende Infrastruktur einer VPC, um eine VPC-Peering-Verbindung herzustellen. Es handelt sich weder um ein Gateway noch um eine AWS Site-to-Site VPN-Verbindung, und es ist nicht auf eine separate physische Hardware angewiesen. Es gibt keinen einzelnen Fehlerpunkt für die Kommunikation und keinen Bandbreitenengpass.

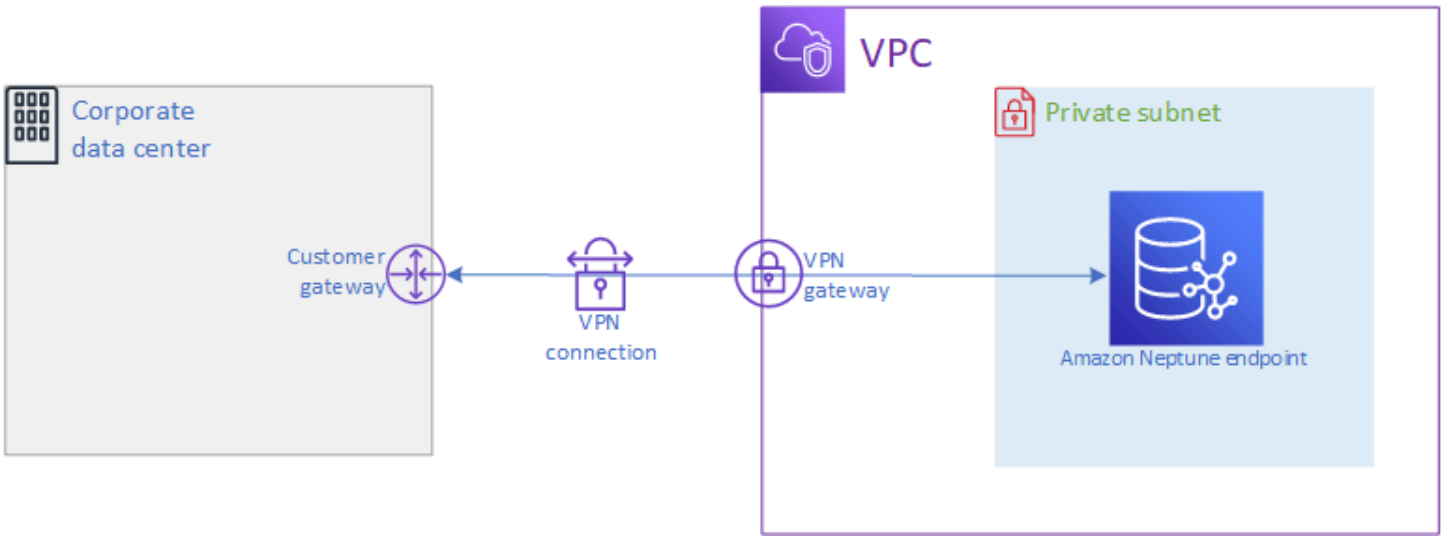
Weitere Informationen zur Verwendung von VPC-Peering finden Sie im Amazon-VPC-Peering-Leitfaden.

Verbindung zu einem Amazon Neptune Neptune-Cluster über ein privates Netzwerk herstellen

Sie können von einem privaten Netzwerk auf zwei verschiedene Arten auf einen Neptune-DB-Cluster zugreifen:

- Verwenden einer AWS Site-to-Site VPN-Verbindung.
- Verwendung einer AWS -Direct-Connect-Verbindung.

Die oben angegebenen Links enthalten Informationen zu diesen Verbindungsmethoden und zu ihrer Einrichtung. Die Konfiguration einer AWS Site-to-Site Verbindung könnte so aussehen:



Öffentliche Endpunkte von Neptune

-Übersicht

Amazon Neptune Neptune-Cluster werden normalerweise in Ihrer VPC bereitgestellt und können nur von dieser VPC aus aufgerufen werden. Dies erfordert die Konfiguration von Anwendungen

und Entwicklungsumgebungen innerhalb der VPC oder die Verwendung von Proxydiensten für die Verbindung mit der VPC, was die Einrichtungszeit und die Kosten erhöht.

Öffentliche Endpunkte vereinfachen diese Erfahrung, indem sie direkte Verbindungen zu Neptune über das Internet ermöglichen, was den Einstieg in Graphdatenbanken ohne spezielle Netzwerkkenntnisse erleichtert.

Wann sollten öffentliche Endpunkte verwendet werden

Erwägen Sie die Verwendung öffentlicher Endpunkte in den folgenden Szenarien:

- Sie möchten Neptune schnell in einer Entwicklungs- oder Testumgebung ohne komplexe Netzwerkkonfiguration testen
- Sie verfügen nicht über spezielle Netzwerkkenntnisse AWS
- Der Sicherheitsstatus Ihrer Anwendung erfordert keine private VPC
- Sie müssen von Ihrer lokalen Entwicklungsumgebung aus eine Verbindung zu Neptune herstellen

Sicherheitsüberlegungen

Beachten Sie bei der Verwendung öffentlicher Endpunkte die folgenden Sicherheitsaspekte:

- Für Cluster mit aktivierten öffentlichen Endpunkten ist eine IAM-Authentifizierung erforderlich.
- Der Zugriff auf die Datenbank wird von der verwendeten Sicherheitsgruppe gesteuert.
- Sie können einschränken, welche IP-Adressen eine Verbindung zu Ihrem Cluster herstellen können.
- Mithilfe von IAM-Richtlinien können Sie steuern, wer Cluster mit öffentlichem Zugriff erstellen oder ändern kann. Siehe: Die Erstellung von öffentlichem Zugriff einschränken

Öffentliche Endpunkte aktivieren

Standardmäßig werden neue Neptune-Datenbanken mit deaktivierten öffentlichen Endpunkten erstellt. Sie müssen den öffentlichen Zugriff explizit aktivieren, wenn Sie einen Cluster erstellen oder ändern.

Öffentliche Endpunkte werden ab der Neptune Engine-Release-Version 1.4.6.x unterstützt. Sie müssen bestehende Cluster auf mindestens diese Version aktualisieren, um diese Funktion nutzen zu können.

Die öffentliche Endpunkteinstellung ist auf der Neptune-Instanz und nicht auf dem Neptun-Cluster verfügbar. Daher kann ein Neptun-Cluster bei einigen Instanzen mit öffentlichen Endpunkten existieren und bei anderen ohne. Wir empfehlen jedoch nicht, eine solche Einstellung zu verwenden. Weitere Informationen dazu finden Sie unter: So funktionieren öffentliche Endgeräte

Voraussetzungen

IAM-Authentifizierungseinstellung auf dem Neptune-Cluster

Bevor Sie öffentliche Endpunkte auf einer Neptune-Instance aktivieren, stellen Sie sicher, dass Ihr Cluster die IAM-Authentifizierung unterstützt. Wenn nicht, aktivieren Sie es mit dem folgenden Befehl:

```
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
```

Netzwerkeinstellungen

1. Stellen Sie sicher, dass Ihre VPC über Subnetze verfügt, die öffentliches Routing ermöglichen (hat einen Eintrag für ein Internet-Gateway in der Routing-Tabelle der Subnetze). Wenn Sie beim Erstellen des Clusters keinen `db-subnet-group-name` Parameter angeben, wird die Standard-Subnetzgruppe für die Clustererstellung ausgewählt.
2. Stellen Sie sicher, dass die dem Cluster zugeordnete Sicherheitsgruppe eingehenden Datenverkehr für die zulässigen IP-Bereiche und die zulässigen Ports zulässt. Wenn Sie beispielsweise zulassen möchten, dass TCP-Verkehr von allen IPs eine Verbindung zur Neptune-Instanz herstellt, die auf Port 8182 ausgeführt wird, sollte die Regel für eingehenden Datenverkehr Folgendes enthalten:
   a. Typ: Alles TCP
   b. Protokoll: TCP
   c. Portbereich: 8182
   d. CIDR-Block: 0.0.0. 0/0

> **Note**
>
> Sie können den CIDR-Blockbereich jedoch auf 0,0.0 setzen. 0/0, wir empfehlen, diesen Wert auf einen bestimmten IP-Bereich Ihrer Client-Anwendung zu reduzieren, um eine bessere Sicherheitslage zu gewährleisten.

Eine neue Instanz mit öffentlichen Endpunkten erstellen

Sie können mithilfe der AWS Management Console, der AWS CLI oder des SDK eine neue Neptune-Instanz mit öffentlichen Endpunkten erstellen. AWS

Verwenden der 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
```

Änderung einer vorhandenen Instanz für den öffentlichen Zugriff

Um eine bestehende Neptune-Instanz zu ändern, um den öffentlichen Zugriff zu ermöglichen:

```
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**
>
> Der öffentliche Zugriff wird auf Instanzebene aktiviert, nicht auf Clusterebene. Um sicherzustellen, dass Ihr Cluster immer über öffentliche Endpunkte zugänglich ist, muss für alle Instances im Cluster der öffentliche Zugriff aktiviert sein.

Verwenden der öffentlichen Endpunkte

Um zu überprüfen, ob Ihre Datenbank erreichbar ist, überprüfen Sie den Status mithilfe der AWS `NeptuneData` CLI-API:

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

Wenn auf die Datenbank zugegriffen werden kann, lautet die Antwort wie folgt:

```
{
    "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"
    }
}
```

Beispiele für die Abfrage der Datenbank

AWS 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))
        }
    }
}
```

Wie funktionieren öffentliche Endpunkte

Wenn eine Neptun-Instanz öffentlich zugänglich ist:

- Sein DNS-Endpunkt wird in der VPC des DB-Clusters in die private IP-Adresse aufgelöst.
- Es wird in die öffentliche IP-Adresse von außerhalb der VPC des Clusters aufgelöst.
- Der Zugriff wird von der Sicherheitsgruppe gesteuert, die dem Cluster zugewiesen ist.
- Über das Internet kann nur auf öffentlich zugängliche Instanzen zugegriffen werden.

Verhalten der Leser-Endgeräte

- Wenn alle Reader-Instanzen öffentlich zugänglich sind, wird der Reader-Endpunkt immer über das öffentliche Internet aufgelöst.

- Wenn nur einige Reader-Instanzen öffentlich zugänglich sind, wird der Reader-Endpunkt nur dann öffentlich aufgelöst, wenn er eine öffentlich zugängliche Instanz für die Bearbeitung der Leseabfrage auswählt.

Verhalten der Cluster-Endpunkte

- Der DB-Cluster-Endpunkt wird immer zum Instance-Endpunkt des Writers aufgelöst.
- Wenn der öffentliche Endpunkt auf der Writer-Instance aktiviert ist, ist der Cluster-Endpunkt öffentlich zugänglich, andernfalls nicht.

Verhalten nach einem Cluster-Failover

- Ein Neptun-Cluster kann Instances in verschiedenen öffentlich zugänglichen Einstellungen haben.
- Wenn ein Cluster über einen öffentlichen Writer und einen nicht-öffentlichen Reader verfügt, wird nach einem Cluster-Failover der neue Writer (vorheriger Reader) nicht öffentlich und der neue Reader (vorheriger Writer) wird öffentlich.

Anforderungen an die Netzwerkkonfiguration

Damit öffentliche Endpunkte ordnungsgemäß funktionieren:

1. Die Neptune-Instances müssen sich in öffentlichen Subnetzen innerhalb Ihrer VPC befinden.
2. Die mit diesen Subnetzen verknüpften Routentabellen müssen eine Route zu einem Internet-Gateway für 0.0.0 enthalten. 0/0.
3. Die Sicherheitsgruppe muss den Zugriff von den öffentlichen IP-Adressen oder CIDR-Bereichen aus zulassen, für die Sie Zugriff gewähren möchten.

Die Erstellung von öffentlichem Zugriff einschränken

Sie können IAM-Richtlinien verwenden, um einzuschränken, wer Neptune-Cluster mit öffentlichem Zugriff erstellen oder ändern kann. Die folgende Beispielrichtlinie verweigert die Erstellung von Neptune-Instanzen mit öffentlichem Zugriff:

JSON

```
{
```

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

Weitere Informationen zum `rds:PublicAccessEnabled` IAM-Bedingungsschlüssel: Amazon RDS Service Authorization Reference

CloudFormation Support

Sie können CloudFormation Neptune-Cluster mit aktivierten öffentlichen Endpunkten starten, indem Sie den `PubliclyAccessible` Parameter in Ihrer Vorlage angeben. CloudFormation

Kompatibilität mit Neptune-Funktionen

Ein Cluster mit aktivierten öffentlichen Endpunkten unterstützt alle Neptune-Funktionen, die ein VPC-only Cluster unterstützt, einschließlich:

- Neptun-Werkbank
- Integration der Volltextsuche
- Neptunströme
- Benutzerdefinierte Endpunkte
- Neptune Serverlos
- Grafik-Explorer

Preisgestaltung

Öffentliche Endgeräte sind ohne zusätzliche Kosten verfügbar, die über die Standardpreise von Neptune hinausgehen. Eine Verbindung von Ihrer lokalen Umgebung zu Neptune über eine öffentliche IP kann jedoch zu erhöhten Datenübertragungskosten führen.

Sicherung des Zugriffs auf einen Amazon Neptune Neptune-Cluster

Es gibt mehrere Möglichkeiten, wie Sie Ihre Amazon–Neptune-Cluster schützen können.

Verwenden von IAM-Richtlinien zur Einschränkung des Zugriffs auf einen Neptune-DB-Cluster

Um zu kontrollieren, wer Neptune-Verwaltungsaktionen auf Neptune-DB-Clustern und DB-Instances ausführen kann, verwenden Sie AWS Identity and Access Management (IAM).

Wenn Sie ein IAM-Konto für den Zugriff auf die Neptune-Konsole verwenden, müssen Sie sich zunächst AWS-Managementkonsole mit Ihrem IAM-Konto anmelden, bevor Sie die Neptune-Konsole unter öffnen. https://console.aws.amazon.com/neptune/home

Wenn Sie AWS mithilfe von IAM-Anmeldeinformationen eine Verbindung herstellen, muss Ihr IAM-Konto über IAM-Richtlinien verfügen, die die für die Ausführung von Neptune-Verwaltungsoperationen erforderlichen Berechtigungen gewähren. Weitere Informationen finden Sie unter Verwenden verschiedener Arten von IAM-Richtlinien für die Steuerung des Zugriffs auf Neptune.

Verwenden von VPC-Sicherheitsgruppen zur Einschränkung des Zugriffs auf einen Neptune-DB-Cluster

Neptune-DB-Cluster müssen in einer Amazon Virtual Private Cloud (VPC) erstellt werden. Mit einer VPC-Sicherheitsgruppe können Sie steuern, welche Geräte und EC2-Instances Verbindungen zum Endpunkt und Port der DB-Instance für Neptune-DB-Cluster in einer VPC herstellen können. Weitere Informationen über VPCs finden Sie unter Eine Sicherheitsgruppe über die Konsole erstellen.

> **Note**
>
> Um eine Verbindung zu Ihrem Neptune-Cluster herzustellen, müssen Sie den Datenbank-Port des Clusters (Standard 8182) sowohl für die eingehenden als auch für die ausgehenden Regeln verfügbar machen, um eine ordnungsgemäße Konnektivität zu gewährleisten.

Verwenden der IAM-Authentifizierung zur Einschränkung des Zugriffs auf einen Neptune-DB-Cluster

Wenn Sie die AWS Identity and Access Management (IAM-) Authentifizierung in einem Neptune-DB-Cluster aktivieren, muss jeder, der auf den DB-Cluster zugreift, zuerst authentifiziert werden. Informationen zum Einrichten der IAM-Authentifizierung finden Sie unter Authentifizieren Sie Ihre Amazon Neptune Neptune-Datenbank mit AWS Identity and Access Management.

Informationen zur Verwendung temporärer Anmeldeinformationen für die Authentifizierung, einschließlich Beispiele für AWS CLI, und Amazon EC2 AWS Lambda, finden Sie unter. the section called “Temporäre Anmeldeinformationen”

Unter den folgenden Links finden Sie zusätzliche Informationen zur Verbindung mit Neptune mittels IAM-Authentifizierung mit den verschiedenen Abfragesprachen:

Verwenden von Gremlin mit IAM-Authentifizierung

- the section called “Von der Gremlin-Konsole aus eine Verbindung herstellen”
- the section called “Verbindung mit Gremlin Java herstellen”
- the section called “Gremlin-Python”
- the section called “Gremlin Javascript”
- the section called “Gremlin Go”
- the section called “Gremlin .NET”

Verwenden von OpenCypher mit IAM-Authentifizierung

- the section called “Beispiel für eine Java-IAM-Authentifizierung über Bolt”
- the section called “Beispiel für eine Python-IAM-Authentifizierung über Bolt”
- the section called “Node.js mit IAM-Authentifizierung”

- the section called “Beispiel für eine .NET-IAM-Authentifizierung über Bolt”
- the section called “Beispiel für Golang IAM-Authentifizierung (Bolt)”

Verwenden von SPARQL mit IAM-Authentifizierung

- the section called “Verbindung mit SPARQL Java herstellen”
- the section called “Herstellen einer Verbindung mit SPARQL und Node.js”
- the section called “Herstellen einer Verbindung mit Python”

Zugreifen auf Grafikdaten in Amazon Neptune

Sie können mit einem Amazon Neptune Neptune-DB-Cluster interagieren, nachdem Sie die Netzwerkkonnektivität hergestellt haben. Wenn Sie noch keinen Zugriff auf Ihren Cluster eingerichtet haben, finden Sie weitere Informationen unter. Verbinden mit einem Cluster In den folgenden Abschnitten werden die Tools und Abfragesprachen beschrieben, mit denen Sie Daten laden, Abfragen ausführen und Ihr Diagramm verwalten können.

Tools für den Zugriff auf Neptune

Neptune unterstützt mehrere Tools zum Senden von Abfragen und zum Verwalten Ihrer Grafikdaten:

- AWS CLI— Verwenden Sie die `aws neptunedata` Befehle, um Gremlin- und OpenCypher-Abfragen auszuführen, den Engine-Status zu überprüfen, Massenladungen zu verwalten und vieles mehr. Weitere Informationen finden Sie in der Befehlsreferenz. neptunedata AWS CLI
- AWS SDKs — Verwenden Sie die Neptune Data API über die AWS SDKs, um Abfragen programmgesteuert auszuführen. SDKs sind für Gremlin und OpenCypher verfügbar.
- **`curl`**und **`awscurl`** — Wird verwendet`curl`, um HTTP-Anfragen direkt an die Neptune-Endpunkte zu senden. Wenn die IAM-Authentifizierung aktiviert ist, verwenden Sie awscurl oder `curl` 7.75.0+ mit der Option, Anfragen zu signieren. `--aws-sigv4` Weitere Informationen finden Sie unter Verwenden von `awscurl` mit temporären Anmeldeinformationen, um eine sichere Verbindung zu einem DB-Cluster mit aktivierter IAM-Authentifizierung herzustellen.
- Neptun-Notizbücher — Verwenden Sie Neptun-Notizbücher, um interaktive Abfragen in einer Jupyter-Umgebung mit integrierten Visualisierungen auszuführen. Weitere Informationen finden Sie unter Neptune mit Grafiknotizbüchern verwenden.
- Treiber — Connect mit sprachspezifischen Treibern für jede Abfragesprache her. Verwenden Sie für Gremlin TinkerPop-compliant Treiber, die im Apache-Projekt verfügbar sind. TinkerPop

Verwenden Sie für OpenCypher die Bolt-Protokolltreiber, wie unter beschrieben. Verwenden des Bolt-Protokolls Verwenden Sie für SPARQL kompatible Tools wie RDF4J für Java, wie unter beschrieben. Java

Verwenden einer Abfragesprache für den Zugriff auf Diagrammdaten in Ihrem Neptune-DB-Cluster

Wenn Sie eine Verbindung hergestellt haben, können Sie die Abfragesprachen Gremlin und openCypher verwenden, um ein Eigenschaftsdiagramm zu erstellen und abzufragen, oder die Abfragesprache SPARQL, um ein Diagramm mit RDF-Daten zu erstellen und abzufragen.

Von Neptune unterstützte Diagrammabfragesprachen

- Gremlin ist eine Diagrammtraversierungssprache für Eigenschaftsdiagramme. Eine Abfrage in Gremlin ist eine Transversale, die aus verschiedenen Schritten besteht. Jeder Schritt folgt einem Edge zu einem Knoten. Weitere Informationen finden Sie in der Gremlin-Dokumentation bei Apache. TinkerPop

  Die Neptune-Implementierung von Gremlin weist einige Unterschiede zu anderen Implementierungen auf, insbesondere wenn Sie Gremlin-Groovy (Gremlin-Abfragen, die als serialisierter Text gesendet werden) verwenden. Weitere Informationen finden Sie unter Einhaltung der Gremlin-Standards in Amazon Neptune.

  Um zu beginnen, sehen Sie sich Verwenden von Gremlin an.

- openCypher ist eine deklarative Abfragesprache für Eigenschaftsdiagramme. Ursprünglich von Neo4j entwickelt, wurde sie 2015 als Open-Source-Software veröffentlicht und ist unter einer Apache 2-Open-Source-Lizenz für das openCypher-Projekt verfügbar. Die Syntax ist in der Cypher Query Language Reference, Version 9 dokumentiert.

  Um zu beginnen, sehen Sie sich Verwenden von openCypher an.

- SPARQL ist eine deklarative Abfragesprache für RDF-Daten, die auf dem Graph Pattern Matching basiert, der vom World Wide Web Consortium (W3C) standardisiert und in SPARQL 1.1 Overview und der SPARQL 1.1 Query Language Specification beschrieben ist.

  Um zu beginnen, sehen Sie sich Verwenden von SPARQL an.

> **Note**
>
> Sie können mit Gremlin oder openCypher auf Eigenschaftsdiagrammdaten in Neptune zugreifen, nicht mit SPARQL. Sie können nur mit SPARQL auf RDF-Daten zugreifen, nicht mit Gremlin oder openCypher.

Verwenden von Befehlszeilentools für den Zugriff auf Amazon Neptune

Sie können die AWS CLI AWS SDKs und HTTP-Tools wie `curl` und verwenden, `awscurl` um Anfragen an Ihren Neptune-DB-Cluster zu senden. In den folgenden Abschnitten wird gezeigt, wie Sie die einzelnen Tools einrichten und grundlegende Gremlin- und OpenCypher-Abfragen ausführen.

Verwendung der AWS CLI

Mit den `aws neptunedata` Befehlen können Sie Gremlin- und OpenCypher-Abfragen ausführen, den Engine-Status überprüfen, Massenladungen verwalten und vieles mehr. Die vollständige Befehlsreferenz finden neptunedataSie in der Befehlsreferenz. AWS CLI

Die folgenden Beispiele zeigen, wie eine einfache Abfrage ausgeführt wird:

Gremlin

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

Weitere Informationen finden Sie unter execute-gremlin-query in der Befehlsreferenz. 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"
```

Weitere Informationen finden Sie unter execute-open-cypher-query in der Befehlsreferenz. AWS CLI

Verwendung der AWS SDK

Sie können die Neptune Data API über die AWS SDKs verwenden, um Abfragen programmgesteuert auszuführen. Die folgenden Python-Beispiele zeigen, wie eine einfache Abfrage ausgeführt wird:

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))
```

AWS SDK-Beispiele in anderen Sprachen finden Sie unterAWS 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))
```

AWS SDK-Beispiele in anderen Sprachen finden Sie unterAWS SDK.

Verwenden von curl und awscurl

Das curl-Befehlszeilentool sendet HTTP-Anfragen direkt an die Neptune-Endpunkte. Wenn die IAM-Authentifizierung aktiviert ist, verwenden Sie awscurl oder 7.75.0+ mit der Option, Anfragen zu signieren. `curl --aws-sigv4` Weitere Informationen finden Sie unter Verwenden von `awscurl` mit temporären Anmeldeinformationen, um eine sichere Verbindung zu einem DB-Cluster mit aktivierter IAM-Authentifizierung herzustellen.

Curl für HTTPS einrichten

Um eine Verbindung über HTTPS herzustellen (wie es Neptune in den meisten Regionen erfordert), `curl` ist Zugriff auf entsprechende Zertifikate erforderlich. Informationen zum Abrufen von Zertifikaten und deren Formatierung in einen Zertifizierungsstellenspeicher (CA) finden Sie in der `curl` Dokumentation unter SSL-Zertifikatsverifizierung.

Sie können den Speicherort dieses CA-Zertifikatsspeichers mithilfe der `CURL_CA_BUNDLE` Umgebungsvariablen angeben. Unter Windows sucht `curl` automatisch in einer Datei mit dem Namen `curl-ca-bundle.crt` danach. Es sucht zunächst in demselben Verzeichnis wie `curl.exe` und anschließend an anderer Stelle im Pfad. Weitere Informationen finden Sie unter SSL Certificate Verification.

Solange `curl` die entsprechenden Zertifikate finden kann, werden HTTPS-Verbindungen wie HTTP-Verbindungen behandelt, ohne dass zusätzliche Parameter angegeben werden müssen. Die Beispiele in dieser Dokumentation basieren auf diesem Szenario.

Weitere Informationen über das Tool finden Sie in der curl-Manpage und im Buch Everything curl.

Abfragebeispiele

Die folgenden Beispiele zeigen, wie eine einfache Abfrage mit `curl` und ausgeführt wird: `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 `curl` funktioniert nur, wenn die IAM-Authentifizierung auf Ihrem Cluster nicht aktiviert ist.

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 `curl` funktioniert nur, wenn die IAM-Authentifizierung auf Ihrem Cluster nicht aktiviert ist.

Weitere Gremlin-HTTP-Beispiele finden Sie unter. HTTPS REST Weitere OpenCypher-HTTP-Beispiele finden Sie unter. HTTPS-Endpunkt

Verwenden von Gremlin für den Zugriff auf Grafikdaten in Amazon Neptune

Sie können die Gremlin-Konsole verwenden, um mit TinkerPop Graphen und Abfragen in einer REPL-Umgebung (Read-Eval-Print Loop) zu experimentieren.

Das folgende Tutorial führt Sie durch die Verwendung der Gremlin-Konsole zum Hinzufügen von Scheitelpunkten, Kanten, Eigenschaften und mehr zu einem Neptun-Diagramm und zeigt einige Unterschiede in der Gremlin-Implementierung auf. Neptune-specific

> **Note**
>
> In diesem Beispiel wird davon ausgegangen, dass Sie Folgendes durchgeführt haben:
>
> - Sie haben über SSH eine Verbindung mit einer Amazon–EC2-Instance hergestellt.
> - Sie haben einen Neptune-Cluster wie in Neptun-Cluster erstellen beschrieben erstellt.
> - Sie haben die Gremlin-Konsole wie in Installieren der Gremlin-Konsole beschrieben installiert.

Verwenden der Gremlin-Konsole

1. Wechseln Sie in den Ordner, in dem die Dateien der Gremlin-Konsole extrahiert werden.

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

2. Geben Sie den folgenden Befehl ein, um die Gremlin-Konsole auszuführen.

   ```
   bin/gremlin.sh
   ```

   Die Ausgabe sollte folgendermaßen aussehen:

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

Sie sehen nun die `gremlin>`-Eingabeaufforderung. Die nächsten Schritte geben Sie über diese Eingabeaufforderung ein.

3. Geben Sie an der `gremlin>`-Eingabeaufforderung Folgendes ein, um eine Verbindung zur Neptune-DB-Instance herzustellen.

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

4. Geben Sie an der `gremlin>`-Eingabeaufforderung Folgendes ein, um in den Remote-Modus zu wechseln. Dadurch werden alle Gremlin-Abfragen an die Remote-Verbindung gesendet.

```
:remote console
```

5. Fügen Sie einen Eckpunkt mit Bezeichnung und Eigenschaft hinzu.

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

Dem Vertex wird eine `string`-ID mit einer GUID zugewiesen. Alle Eckpunkt-IDs sind Zeichenfolgen in Neptune.

6. Fügen Sie einen Eckpunkt mit einer benutzerdefinierter ID hinzu.

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

Die `id`-Eigenschaft wird nicht in Anführungszeichen gesetzt. Es handelt sich um ein Schlüsselwort für die ID des Vertex. Die Vertex-ID ist eine Zeichenfolge mit der Zahl 1.

Normale Eigenschaftsnamen müssen in Anführungszeichen eingeschlossen werden.

7. Ändern Sie die Eigenschaft oder fügen Sie eine Eigenschaft hinzu, wenn keine vorhanden ist.

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

Hier ändern Sie die `name`-Eigenschaft für den Vertex aus dem vorherigen Schritt. Dadurch werden alle vorhandenen Werte aus der `name`-Eigenschaft entfernt.

Wenn Sie `single` nicht angegeben haben, wird stattdessen der Wert der `name`-Eigenschaft angehängt, sofern dieser Vorgang noch nicht ausgeführt wurde.

8. Hinzufügen einer Eigenschaft bzw. Anfügen der Eigenschaft, wenn die Eigenschaft bereits über einen Wert verfügt

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

Neptune verwendet Set-Kardinalität als Standardaktion.

Dieser Befehl fügt die Eigenschaft `age` mit dem Wert 29 hinzu, ersetzt jedoch keine vorhandenen Werte.

Wenn die `age`-Eigenschaft bereits einen Wert enthält, fügt dieser Befehl der Eigenschaft 29 an. Beispiel: Wenn die `age`-Eigenschaft 27 lautet, ist der neue Wert `[ 27, 29 ]`.

9. Fügen Sie mehrere Eckpunkte hinzu.

```
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)
```

Sie können mehrere Anweisungen gleichzeitig an Neptune senden.

Anweisungen können durch Zeilenumbrüche (`'\n'`), Leerzeichen (`' '`), Semikolon (`'; '`) oder nichts (`g.addV('person').iterate()g.V()` ist z. B. gültig) getrennt werden.

> **Note**
>
> Die Gremlin-Konsole sendet bei jedem Zeilenumbruch (`'\n'`) einen separaten Befehl, sodass in diesem Fall jeweils eine separate Transaktion stattfindet. Dieses Beispiel enthält alle Befehle in getrennten Zeilen, um die Lesbarkeit zu verbessern. Entfernen Sie die Zeilenumbruch-Zeichen (`'\n'`), um sie als einen einzelnen Befehl über die Gremlin-Konsole zu senden.

Alle Anweisungen mit Ausnahme der letzten müssen mit einem Beendigungsschritt wie `.next()` oder `.iterate()` enden. Andernfalls werden sie nicht ausgeführt. Die Gremlin-

Konsole erfordert keinen Beendigungsschritt. Verwenden Sie `.iterate` immer dann, wenn die Ergebnisse nicht serialisiert werden müssen.

Alle Anweisungen, die zusammen gesendet werden, sind in einer einzigen Transaktion enthalten und sind entweder alle erfolgreich oder schlagen alle fehl.

10. Fügen Sie Eckpunkte hinzu.

```
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)
```

Nachfolgend sind zwei verschiedene Möglichkeiten zum Hinzufügen einer Edge aufgeführt.

11. Fügen Sie den Rest des Graphs „Modern“ hinzu.

```
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. Löschen Sie einen Eckpunkt.

```
g.V().has('name', 'justin').drop()
```

Entfernt den Vertex mit der `name`-Eigenschaft gleich `justin`.

> **Important**
>
> Wenn Sie hier aufhören, haben Sie das vollständige Apache Modern-Diagramm. TinkerPop Die Beispiele im Abschnitt Traversal der TinkerPop Dokumentation verwenden den Modern-Graphen.

13. Führen Sie einen Durchlauf aus.

```
g.V().hasLabel('person')
```

Gibt alle `person`-Eckpunkte zurück.

14. Führen Sie einen Durchlauf mit den Werten (valueMap()) aus.

```
g.V().has('name', 'marko').out('knows').valueMap()
```

Gibt Schlüssel-Wert-Paare für alle Vertices zurück, die `marko` "kennt".

15. Geben Sie mehrere Bezeichnungen an.

```
g.addV("Label1::Label2::Label3")
```

Neptune unterstützt mehrere Bezeichnungen für einen Eckpunkt. Sie können mehrere Bezeichnungen angeben, indem Sie diese durch `::` trennen.

In diesem Beispiel wird ein Vertex mit drei verschiedenen Bezeichnungen hinzugefügt.

Der `hasLabel`-Schritt entspricht dem Knoten mit allen drei Bezeichnungen: `hasLabel("Label1")`, `hasLabel("Label2")` und `hasLabel("Label3")`.

Das `::`-Trennzeichen ist dieser Verwendung vorbehalten.

Sie können im `hasLabel`-Schritt nicht mehrere Bezeichnungen angeben. Beispiel: Für `hasLabel("Label1::Label2")` gibt es keine Übereinstimmung.

16. Spezifizieren Sie. Time/date

```
g.V().property(single, 'lastUpdate', datetime('2018-01-01T00:00:00'))
```

Neptune unterstützt kein Java-Datum. Verwenden Sie stattdessen die `datetime()` Funktion. `datetime()`akzeptiert eine ISO8061-compliant `datetime` Zeichenfolge.

Unterstützt werden die folgenden Formate: `YYYY-MM-DD`, `YYYY-MM-DDTHH:mm`, `YYYY-MM-DDTHH:mm:SS` und `YYYY-MM-DDTHH:mm:SSZ`.

17. Löschen Sie Eckpunkte, Eigenschaften oder Edges.

```
g.V().hasLabel('person').properties('age').drop().iterate()
g.V('1').drop().iterate()
g.V().outE().hasLabel('created').drop()
```

Im Folgenden finden Sie einige "drop"-Beispiele.

> **Note**
>
> Der `.next()`-Schritt funktioniert nicht mit `.drop()`. Verwenden Sie stattdessen `.iterate()`.

18. Wenn Sie fertig sind, geben Sie den folgenden Befehl ein, um die Gremlin-Konsole zu beenden.

```
:exit
```

> **Note**
>
> Verwenden Sie ein Semikolon (`;`) oder ein Zeilenumbruchzeichen (`\n`), um die Anweisungen voneinander abzutrennen.
> Jede Traversierung, die der letzten Traversierung vorausgeht, muss zum Ausführen mit `iterate()` enden. Es werden nur die Daten der letzten Traversierung zurückgegeben.

Verwenden von OpenCypher für den Zugriff auf Grafikdaten in Amazon Neptune

Informationen zu den ersten Schritten mit OpenCypher finden Sie unter openCypher oder verwenden Sie die OpenCypher-Notizbücher im Neptune Graph-Notebook-Repository. GitHub

Verwenden von SPARQL für den Zugriff auf Grafikdaten in Amazon Neptune

SPARQL ist eine Abfragesprache für das Resource Description Framework (RDF), ein für das Web entwickelts Datenformat für Diagramme. Amazon Neptune ist mit SPARQL 1.1 kompatibel. Das bedeutet, dass Sie eine Verbindung zu einer Neptune-DB-Instance herstellen und das Diagramm mittels der in der Spezifikation SPARQL 1.1 Query Language beschriebenen Abfragesprache abfragen können.

Eine Abfrage in SPARQL besteht aus einer `SELECT`-Klausel zur Angabe der Variablen, die zurückgegeben werden sollen, und einer `WHERE`-Klausel, um anzugeben, welche Daten im Diagramm abgeglichen werden sollen. Wenn Sie noch keine Erfahrungen mit SPARQL-Abfragen haben, lesen Sie den Abschnitt Writing Simple Queries in SPARQL 1.1 Query Language.

Der HTTP-Endpunkt für SPARQL-Abfragen an eine Neptune-DB-Instance ist `https://your-neptune-endpoint:port/sparql`.

So stellen Sie eine Verbindung mit SPARQL her

1. Sie können den SPARQL-Endpunkt für Ihren Neptune-Cluster aus dem SparqlEndpointElement im Abschnitt Outputs des Stacks abrufen. CloudFormation
2. Geben Sie Folgendes ein, um ein SPARQL-**`UPDATE`** über HTTP `POST` und den curl-Befehl zu senden.

   ```
   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
   ```

   Im vorherigen Beispiel wird das folgende Triple in das standardmäßige SPARQL-Diagramm eingefügt: `<https://test.com/s> <https://test.com/p> <https://test.com/o>`
3. Geben Sie Folgendes ein, um ein SPARQL-**`QUERY`** über HTTP `POST` und den curl-Befehl zu senden.

   ```
   curl -X POST --data-binary 'query=select ?s ?p ?o where {?s ?p ?o} limit 10'
    https://your-neptune-endpoint:port/sparql
   ```

   Das vorherige Beispiel gibt bis zu 10 der Triples (subject-predicate-object) im Graph zurück, indem Sie die `?s ?p ?o`-Abfrage mit einem Grenzwert von 10 ausführen. Um etwas anderes abzufragen, ersetzen Sie diese durch eine andere SPARQL-Abfrage.

   > **Note**
   >
   > Der Standard-MIME-Typ einer Antwort ist `application/sparql-results+json` für `SELECT`- und `ASK`-Abfragen.
   > Der Standard-MIME-Typ einer Antwort ist `application/n-quads` für `CONSTRUCT`- und `DESCRIBE`-Abfragen.
   > Eine Liste der verfügbaren MIME-Typen finden Sie unter SPARQL HTTP-API.

Daten in einen Amazon Neptune Neptune-Cluster laden

Amazone Neptune bietet einen Prozess, um Daten aus externen Dateien direkt in eine Neptune-DB-Instance zu laden. Sie können diese Vorgehensweise verwenden, anstatt eine große Anzahl von `INSERT`-Anweisungen, `addV`- und `addE`-Schritten oder andere API-Aufrufe auszuführen.

Im Folgenden finden Sie Links zu weiteren Lade-Informationen.

- Methoden für das Laden von Daten – Laden von Daten
- Vom Bulk-Loader unterstützte Datenformate – the section called “Datenformate”
- Beispiel für das Laden – the section called “Beispiel für das Laden”

Überwachung eines Amazon Neptune Neptune-Clusters

Amazon Neptune unterstützt die folgenden Überwachungsmethoden.

- Amazon CloudWatch — Amazon Neptune sendet automatisch Messwerte an Alarme CloudWatch und unterstützt CloudWatch diese auch. Weitere Informationen finden Sie unter the section called “Benutzen CloudWatch”.
- AWS CloudTrail— Amazon Neptune unterstützt die API-Protokollierung mithilfe von. CloudTrail Weitere Informationen finden Sie unter the section called “Neptune-API-Aufrufe protokollieren mit AWS CloudTrail”.
- Tagging – Sie können Tags verwenden, um Ihren Neptune-Ressourcen Metadaten hinzuzufügen und die Nutzung mit Tags nachzuverfolgen. Weitere Informationen finden Sie unter the section called “Neptun-Ressourcen taggen”.
- Audit-Protokolldateien – Sie können Datenbankprotokolldateien über die Neptune-Konsole anzeigen, herunterladen oder beobachten. Weitere Informationen finden Sie unter the section called “Prüfprotokolle mit Neptune”.
- Instance-Status – Überprüfen Sie den Zustand der Graph-Datenbank–Engine einer Neptune-Instance, ermitteln Sie die installierte Version der Engine und rufen Sie über die Instance-Status-API weitere Engine-Statusinformationen ab.

Fehlerbehebung bei einem Amazon Neptune-Cluster

Die folgenden Links können zum Beheben von Problemen mit Amazon Neptune nützlich sein.

- Bewährte Methoden – Lösungen für häufige Probleme und Vorschläge zur Verbesserung der Leistung finden Sie unter Best Practices.
- Service-Fehler – Eine Liste von Fehlern für Verwaltungs-APIs und Graphdatenbank-Verbindungen finden Sie in Neptune-Fehler.
- Service Limits – Informationen zu Neptune-Einschränkungen finden Sie in Neptune-Einschränkungen .
- Engine-Versionen – Informationen zu Graph-Engine-Versionen und Versionshinweise finden Sie in Engine-Versionen.
- Support-Foren – Um an Diskussionen über Neptune teilzunehmen, wechseln Sie zum Amazon-Neptune-Forum.
- Preise – Informationen zu den Kosten der Verwendung von Amazon Neptune finden Sie in Amazon-Neptune-Preise.
- AWS Support — Hilfe und Anleitungen von Experten finden Sie unter AWS Support.

Verwendung von Amazon Neptune Serverless

Amazon Neptune Serverless ist eine On-Demand-Konfiguration zur automatischen Skalierung, die so konzipiert ist, dass Ihr DB-Cluster nach Bedarf skaliert wird, um auch sehr stark gestiegenen Verarbeitungsanforderungen gerecht zu werden, und dann wieder herunterskaliert wird, wenn der Bedarf sinkt. Es hilft, die Prozesse der Überwachung der Workload und der Anpassung der Kapazität für Ihre Neptune-Datenbank zu automatisieren. Die Kapazität wird automatisch basierend auf dem Anwendungsbedarf angepasst und Ihnen werden nur Ressourcen in Rechnung gestellt, die Ihre DB-Cluster verbrauchen.

Anwendungsfälle für Neptune Serverless

Neptune Serverless unterstützt viele Arten von Workloads. Es eignet sich für anspruchsvolle, sehr variable Workloads und kann sehr hilfreich sein, wenn Ihre Datenbanknutzung in der Regel für kurze Zeiträume sehr hoch ist, gefolgt von langen Zeiträumen mit nur leichter oder gar keiner Aktivität. Neptune Serverless ist besonders nützlich für die folgenden Anwendungsfälle:

- Variable Workloads – Workloads mit plötzlichen und unvorhersehbaren CPU-Aktivitätszuwächsen. Mit Neptune Serverless wird Ihre Graphdatenbank automatisch auf die Kapazität skaliert, mit der die Workload erfüllt werden kann, und wieder herunterskaliert, wenn die Aktivitätsspitze vorbei ist. Sie müssen nicht mehr Spitzen- oder durchschnittliche Kapazitäten bereitstellen. Sie können eine obere Kapazitätsgrenze angeben, um Spitzen-Workloads zu bewältigen. Diese Kapazität wird nur genutzt, wenn sie benötigt wird.

  Die Granularität der Skalierung von Neptune Serverless hilft Ihnen, die Kapazität genau an die Anforderungen Ihrer Workload anzupassen. Neptune Serverless kann Kapazität je nach Bedarf in fein abgestuften Schritten hinzufügen oder entfernen. Es kann nur eine halbe Neptune Capacity Unit (NCU) hinzufügen, wenn nur etwas mehr Kapazität benötigt wird.
- Multi-tenant Anwendungen — Durch die Nutzung von Neptune Serverless können Sie für jede der Anwendungen, die Sie ausführen müssen, einen separaten DB-Cluster erstellen, ohne diese Mandantencluster einzeln verwalten zu müssen. Jeder der Tenant-Cluster kann je nach mehreren Faktoren unterschiedliche Auslastungs- und Leerlaufzeiten haben, aber Neptune Serverless kann sie ohne Ihr Eingreifen effizient skalieren.
- Neue Anwendungen – Wenn Sie eine neue Anwendung bereitstellen, sind Sie sich oft nicht sicher, wie viel Datenbankkapazität sie benötigt. Mit Neptune Serverless können Sie einen DB-Cluster

einrichten, der automatisch skaliert werden kann, um den Kapazitätsanforderungen der neuen Anwendung gerecht zu werden, während sie sich entwickeln.

- Planung der Kapazität – Angenommen, Sie passen normalerweise Ihre Datenbankkapazität an oder überprüfen die optimale Datenbankkapazität für Ihre Workload, indem Sie die DB-Instance-Klassen aller DB-Instances in einem Cluster ändern. Mit Neptune Serverless können Sie diesen Verwaltungsaufwand vermeiden. Stattdessen können Sie bestehende DB-Instances von bereitgestellt zu Serverless oder von Serverless zu bereitgestellt ändern, ohne einen neuen DB-Cluster oder eine neue Instance erstellen zu müssen.
- Entwicklung und Testen – Neptune Serverless eignet sich auch perfekt für Entwicklungs- und Testumgebungen. Mit Neptune Serverless können Sie DB-Instances mit einer ausreichend hohen maximalen Kapazität erstellen, um Ihre anspruchsvollsten Anwendungen zu testen, und einer niedrigen Mindestkapazität für alle anderen Zeiten, in denen das System zwischen den Tests möglicherweise inaktiv ist.

Neptune Serverless skaliert nur die Rechenkapazität. Ihr Speichervolumen bleibt gleich und wird durch die Serverless-Skalierung nicht beeinträchtigt.

> **Note**
>
> Sie können das Auto Scaling von Neptune auch mit Neptune Serverless verwenden, um verschiedene Arten von Workload-Variationen zu bewältigen.

Einschränkungen von Amazon Neptune Serverless

- Neptune Serverless ist nur in den folgenden Regionen verfügbar:
  - USA Ost (Nord-Virginia): `us-east-1`
  - USA Ost (Ohio): `us-east-2`
  - USA West (Nordkalifornien): `us-west-1`
  - USA West (Oregon): `us-west-2`
  - Kanada (Zentral): `ca-central-1`
  - Europa (Stockholm): `eu-north-1`
  - Europa (Spanien): `eu-south-2`
  - Europa (Irland): `eu-west-1`

- Europa (London): `eu-west-2`
- Europa (Paris): `eu-west-3`
- Europa (Frankfurt): `eu-central-1`
- Europa (Zürich): `eu-central-2`
- Asien-Pazifik (Tokio): `ap-northeast-1`
- Asien-Pazifik (Seoul): `ap-northeast-2`
- Asien-Pazifik (Singapur): `ap-southeast-1`
- Asien-Pazifik (Sydney): `ap-southeast-2`
- Asien-Pazifik (Jakarta): `ap-southeast-3`
- Asien-Pazifik (Hongkong): `ap-east-1`
- Asien-Pazifik (Mumbai): `ap-south-1`
- Asien-Pazifik (Hyderabad): `ap-south-2`
- Südamerika (São Paulo): `sa-east-1`

- In frühen Engine-Versionen nicht verfügbar – Neptune Serverless ist nur in Engine-Versionen 1.2.0.1 oder höher verfügbar.
- Nicht kompatibel mit dem Neptune-Lookup-Cache – der Lookup-Cache funktioniert nicht mit Serverless-DB-Instances.
- Der maximale Arbeitsspeicher in einer Serverless-Instance beträgt 256 GB – die Einstellung `MaxCapacity` auf 128 NCUs (die höchste unterstützte Einstellung) ermöglicht es einer Neptune Serverless-Instance, auf 256 GB Arbeitsspeicher zu skalieren, was dem eines `R6g.8XL` bereitgestellten Instance-Typs entspricht.

Kapazitätsskalierung in einem Neptune-Serverless-DB-Cluster

Die Einrichtung eines Neptune Serverless DB-Clusters ähnelt der Einrichtung eines normal bereitgestellten Clusters, mit zusätzlicher Konfiguration für minimale und maximale Einheiten für die Skalierung und bei der der Instance-Typ auf `db.serverless` gesetzt ist. Die Skalierungskonfiguration wird in Neptune Capacity Units (NCUs) definiert, von denen jede aus 2 GiB (Gibibyte) Arbeitsspeicher (RAM) zusammen mit der zugehörigen virtuellen Prozessorkapazität (vCPU) und dem Netzwerk besteht. Sie wird als Teil eines `ServerlessV2ScalingConfiguration`-Objekts festgelegt und in JSON wie folgt dargestellt:

```
"ServerlessV2ScalingConfiguration": {
```

```
  "MinCapacity": (minimum NCUs, a floating-point number such as   1.0),
  "MaxCapacity": (maximum NCUs, a floating-point number such as 128.0)
}
```

Zu jedem Zeitpunkt hat jede Neptune-Writer- oder Reader-Instance eine Kapazität, die anhand einer Gleitkommazahl gemessen wird, die die Anzahl der NCUs darstellt, die derzeit von dieser Instance verwendet werden. Sie können die CloudWatch ServerlessDatabaseCapacityMetrik auf Instance-Ebene verwenden, um herauszufinden, wie viele NCUs eine bestimmte DB-Instance derzeit verwendet, und anhand der Metrik NCUUtilization herausfinden, wie viel Prozent der maximalen Kapazität die Instance verwendet. Beide Metriken sind auch auf DB-Cluster-Ebene verfügbar, um die durchschnittliche Ressourcenauslastung für den gesamten DB-Cluster anzuzeigen.

Wenn Sie einen Neptune Serverless DB-Cluster erstellen, legen Sie sowohl die minimale als auch die maximale Anzahl von Neptune Capacity Units (NCUs) für alle Serverless-Instances fest.

Der NCU-Mindestwert, den Sie angeben, legt die kleinste Größe fest, auf die eine Serverless-Instance in Ihrem DB-Cluster schrumpfen kann. Ebenso legt der maximale NCU-Wert die größte Größe fest, auf die eine Serverless-Instance wachsen kann. Der höchste maximale NCU-Wert, den Sie festlegen können, ist 128,0 NCUs, und der niedrigste Minimalwert ist 1,0 NCUs.

Neptune verfolgt kontinuierlich die Auslastung jeder Neptune Serverless-Instance, indem es die Auslastung von Ressourcen wie CPU, Arbeitsspeicher und Netzwerk überwacht. Die Last wird durch die Datenbankoperationen Ihrer Anwendung, durch die Hintergrundverarbeitung für den Server und durch andere Verwaltungsaufgaben generiert.

Wenn die Auslastung einer Serverless-Instance die Grenze der aktuellen Kapazität erreicht oder wenn Neptune andere Leistungsprobleme feststellt, wird die Instance automatisch skaliert. Wenn die Auslastung der Instance abnimmt, wird die Kapazität auf die konfigurierten Mindestkapazitätseinheiten herunterskaliert, wobei die CPU-Kapazität vor dem Arbeitsspeicher freigegeben wird. Diese Architektur ermöglicht die kontrollierte Freigabe von Ressourcen nach unten und bewältigt effektiv Nachfrageschwankungen.

Sie können dafür sorgen, dass eine Reader-Instance zusammen mit der Writer-Instance skaliert oder unabhängig skaliert wird, indem Sie ihre Promotion-Stufe festlegen. Reader-Instances der Upgrade-Stufen 0 und 1 werden gleichzeitig mit dem Writer skalieren, sodass sie die richtige Kapazität dimensioniert sind, um die Workload des Writers im Falle eines Failovers schnell zu übernehmen. Die Leser der Aufstiegsstufen 2 bis 15 skalieren unabhängig von der Autoren-Instance und voneinander.

Wenn Sie Ihren Neptune-DB-Cluster als Cluster erstellt haben, um eine Multi-AZ hohe Verfügbarkeit zu gewährleisten, skaliert Neptune Serverless die Instances in allen AZs entsprechend Ihrer

Datenbanklast nach oben und unten. Sie können die Promotion-Stufe einer Reader-Instance in einer sekundären AZ auf 0 oder 1 setzen, sodass sie zusammen mit der Kapazität der Writer-Instance in der primären AZ nach oben und unten skaliert wird, sodass sie jederzeit bereit ist, die aktuelle Workload zu übernehmen.

> **Note**
>
> Der Speicher für einen Neptune-DB-Cluster besteht aus sechs Kopien all Ihrer Daten, die auf drei AZs verteilt sind, unabhängig davon, ob Sie den Cluster als Multi-AZ Cluster erstellt haben oder nicht. Die Speicherreplikation wird vom Speichersubsystem abgewickelt und wird von Neptune Serverless nicht beeinflusst.

Auswahl eines maximalen Kapazitätswerts für einen Neptune Serverless DB-Cluster

Der kleinste Wert, den Sie für die Mindestkapazität festlegen können, ist `1.0` NCUs.

Achten Sie darauf, dass der Mindestwert nicht unter dem liegt, was Ihre Anwendung für einen effizienten Betrieb benötigt. Ein zu niedriger Wert kann bei bestimmten speicherintensiven Workloads zu einer höheren Timeout-Rate führen.

Wenn Sie den Mindestwert so niedrig wie möglich festlegen, können Sie Geld sparen, da Ihr Cluster bei geringer Nachfrage nur minimale Ressourcen verbraucht. Wenn Ihre Workload jedoch stark schwankt, von sehr niedrig bis sehr hoch, sollten Sie das Minimum höher ansetzen, da durch ein höheres Minimum Ihre Neptune Serverless-Instances schneller skaliert werden können.

Der Grund dafür ist, dass Neptune Skalierungsschritte auf der Grundlage der aktuellen Kapazität wählt. Wenn die aktuelle Kapazität niedrig ist, wird Neptune zunächst langsam hochskalieren. Wenn das Minimum höher ist, beginnt Neptune mit einem größeren Skalierungsschritt und kann daher schneller hochskalieren, um einem starken plötzlichen Anstieg der Workload gerecht zu werden.

Auswahl eines maximalen Kapazitätswerts für einen Neptune Serverless DB-Cluster

Der größte Wert, den Sie für die maximale Kapazität festlegen können, ist `128.0` NCUs, und der kleinste Wert, den Sie für die maximale Kapazität festlegen können, ist `2.5` NCUs. Der von Ihnen festgelegte maximale Kapazitätswert muss mindestens so groß sein wie der von Ihnen festgelegte Mindestkapazitätswert.

Als allgemeine Regel gilt, dass der Maximalwert hoch genug sein sollte, um die Spitzenlast zu bewältigen, der Ihre Anwendung voraussichtlich ausgesetzt sein wird. Ein zu niedriger Wert kann bei bestimmten speicherintensiven Workloads zu einer höheren Timeout-Rate führen.

Wenn Sie den Maximalwert so hoch wie möglich festlegen, hat dies den Vorteil, dass Ihre Anwendung wahrscheinlich auch die unerwartetsten Workloads bewältigen kann. Der Nachteil besteht darin, dass Sie nicht mehr in der Lage sind, die Ressourcenkosten vorherzusagen und zu kontrollieren. Ein unerwarteter Anstieg des Bedarfs kann am Ende viel mehr kosten, als Ihr Budget erwartet hat.

Der Vorteil eines sorgfältig ausgewählten Maximalwerts besteht darin, dass Sie damit Spitzennachfragen decken und gleichzeitig die Rechenkosten von Neptune begrenzen können.

> **Note**
>
> Durch Ändern des Kapazitätsbereichs eines Neptune Serverless DB-Clusters können sich die Standardwerte einiger Konfigurationsparameter ändern. Neptune kann einige dieser neuen Standardeinstellungen sofort anwenden, aber einige der dynamischen Parameteränderungen werden erst nach einem Neustart wirksam. Status `pending-reboot` gibt an, dass ein Neustart erforderlich ist, um einige Parameteränderungen anzuwenden.

Verwenden Sie Ihre bestehende Konfiguration, um die Anforderungen an Serverless abzuschätzen

Wenn Sie in der Regel die DB-Instance-Klasse Ihrer bereitgestellten DB-Instances ändern, um eine außergewöhnlich hohe oder niedrige Workload zu bewältigen, können Sie diese Erfahrung nutzen, um eine grobe Schätzung des entsprechenden Neptune Serverless-Kapazitätsbereichs vorzunehmen.

Schätzen Sie die beste Einstellung für die Mindestkapazität

Sie können das, was Sie über Ihren bestehenden Neptune-DB-Cluster wissen, anwenden, um abzuschätzen, welche Einstellung für die Serverless-Mindestkapazität am besten geeignet ist.

Wenn Ihre bereitgestellte Workload beispielsweise Speicheranforderungen hat, die für kleine DB-Instance-Klassen wie `T3` oder `T4g` zu hoch sind, wählen Sie eine minimale NCU-Einstellung, die vergleichbaren Speicher wie eine DB-Instance-Klasse `R5` oder `R6g` bietet.

Oder nehmen Sie an, dass Sie die DB-Instance-Klasse `db.r6g.xlarge` verwenden, wenn Ihr Cluster eine geringe Workload hat. Diese DB-Instance-Klasse hat 32 GiB Arbeitsspeicher, sodass Sie eine minimale NCU-Einstellung von 16 angeben können, um Serverless-Instances zu erstellen, die auf ungefähr dieselbe Kapazität herunterskalieren können (jede NCU entspricht etwa 2 GiB Arbeitsspeicher). Wenn Ihre `db.r6g.xlarge`-Instance manchmal nicht ausgelastet ist, können Sie einen niedrigeren Wert angeben.

Wenn Ihre Anwendung dann am effizientesten arbeitet, wenn Ihre DB-Instances eine bestimmte Datenmenge im Speicher oder im Puffer-Cache speichern können, sollten Sie eine minimale NCU-Einstellung angeben, die groß genug ist, um genügend Speicher dafür zu haben. Andernfalls könnten Daten aus dem Puffer-Cache entfernt werden, wenn die Serverless-Instances herunterskaliert werden, und müssen im Laufe der Zeit wieder in den Puffer-Cache gelesen werden, wenn Instances wieder hochskaliert werden. Wenn die Menge I/O , mit der Daten wieder in den Puffercache zurückgebracht werden müssen, beträchtlich ist, könnte es sich lohnen, einen höheren NCU-Mindestwert zu wählen.

Wenn Sie feststellen, dass Ihre Serverless-Instances die meiste Zeit mit einer bestimmten Kapazität ausgeführt werden, ist es sinnvoll, die Mindestkapazität etwas niedriger festzulegen. Neptune Serverless kann am effektivsten abschätzen, wie viel und wie schnell hochskaliert werden muss, wenn die aktuelle Kapazität nicht deutlich niedriger als die erforderliche Kapazität ist.

In einer gemischten Konfiguration mit einem bereitgestellten Writer und Neptune Serverless-Readern skalieren die Reader nicht mit dem Writer. Da sie unabhängig skalieren, kann das Festlegen einer geringen Mindestkapazität für sie zu einer übermäßigen Replikationsverzögerung führen. Sie verfügen möglicherweise nicht über ausreichend Kapazität, um mit den Änderungen Schritt zu halten, die der Autor bei einer sehr schreibintensiven Workload vornimmt. Legen Sie in diesem Fall eine Mindestkapazität fest, die mit der Schreibkapazität vergleichbar ist. Wenn Sie die Replikatverzögerung bei Readern beobachten, die sich in den Hochstufungsstufen 2 bis 15 befinden, sollten Sie die Mindestkapazitätseinstellung für Ihren Cluster ggf. erhöhen.

Schätzen Sie die beste Einstellung für die Höchstkapazität

Sie können das, was Sie über Ihren bestehenden Neptune-DB-Cluster wissen, anwenden, um abzuschätzen, welche Einstellung für die Serverless-Höchstkapazität am besten geeignet ist.

Angenommen, Sie verwenden die DB-Instance-Klasse `db.r6g.4xlarge`, wenn Ihr Cluster eine hohe Workload hat. Diese DB-Instance-Klasse hat 128 GiB Arbeitsspeicher, sodass Sie eine maximale NCU-Einstellung von 64 angeben können, um äquivalente Neptune Serverless-Instances

einzurichten (jede NCU entspricht etwa 2 GiB Arbeitsspeicher). Sie können einen höheren Wert angeben, damit die DB-Instance weiter skaliert wird, falls Ihre `db.r6g.4xlarge`-Instance die Workload nicht immer bewältigen kann.

Wenn unerwartete Workload-Spitzen selten sind, kann es sinnvoll sein, Ihre maximale Kapazität hoch genug festzulegen, um die Anwendungsleistung auch während dieser Spitzen aufrechtzuerhalten. Auf der anderen Seite möchten Sie möglicherweise eine niedrigere maximale Kapazität festlegen, um den Durchsatz bei ungewöhnlichen Spitzen zu reduzieren, wobei Neptune jedoch Ihre erwarteten Workloads problemlos bewältigen kann, und um die Kosten zu begrenzen.

Zusätzliche Konfiguration für Neptune Serverless DB-Cluster und -Instances

Neben der Festlegung der Mindest- und Höchstkapazität für Ihren Neptune Serverless DB-Cluster gibt es noch einige weitere Konfigurationsoptionen, die Sie in Betracht ziehen sollten.

Kombination von Serverless- und bereitgestellten Instances in einem DB-Cluster

Ein DB-Cluster muss nicht nur Serverless sein – Sie können eine Kombination aus Serverless- und bereitgestellten Instances (eine gemischte Konfiguration) erstellen.

Nehmen Sie zum Beispiel an, dass Sie mehr Schreibkapazität benötigen, als in einer Serverless-Instance verfügbar ist. In diesem Fall können Sie den Cluster mit einem sehr großen bereitgestellten Writer einrichten und trotzdem Serverless-Instances für die Reader-Instances verwenden.

Oder nehmen Sie an, dass die Schreib-Workload auf Ihrem Cluster variiert, die Lese-Workload jedoch stabil ist. In diesem Fall können Sie Ihren Cluster mit einem Serverless-Writer und einem oder mehreren bereitgestellten Readern einrichten.

Siehe Verwendung von Amazon Neptune Serverless für Informationen darüber, wie Sie einen DB-Cluster mit gemischter Konfiguration erstellen.

Festlegung der Promotion-Stufen für Neptune Serverless-Instances

Bei Clustern mit mehreren Serverless-Instances oder mit einer Mischung aus bereitgestellten und Serverless-Instances achten Sie auf die Einstellung der Hochstufungsstufe für jede einzelne

Serverless-Instance. Diese Einstellung steuert mehr Verhaltensweisen für Serverless-Instances als für bereitgestellte DB-Instances.

In der AWS-Managementkonsole geben Sie diese Einstellung mithilfe der Failover-Priorität unter Zusätzliche Konfiguration auf den Seiten Datenbank erstellen, Instanz ändern und Leser hinzufügen an. Sie sehen diese Eigenschaft für vorhandene Instances in der optionalen Spalte Priority tier (Prioritätsstufe) auf der Seite Databases (Datenbanken). Diese Eigenschaft können Sie auch der Detailseite für einen DB-Cluster oder eine Instance entnehmen.

Bei bereitgestellten Instances bestimmt die Auswahl der Stufe 0 bis 15 nur die Reihenfolge, in der Neptune entscheidet, welche Reader-Instance während eines Failover-Vorgangs zum Writer hochgestuft werden soll. Bei Neptune Serverless Reader-Instances bestimmt die Tier-Nummer auch, ob die Instance skaliert wird, um der Kapazität der Writer-Instance zu entsprechen, oder ob sie unabhängig von ihr nur auf der Grundlage ihrer eigenen Workload skaliert wird.

Neptune Serverless Reader-Instances in Tier 0 oder 1 verfügen über eine Mindestkapazität, die mindestens so hoch ist wie die Writer-Instance, sodass sie im Falle eines Failovers bereit sind, den Writer zu übernehmen. Wenn der Writer eine bereitgestellte Instance ist, schätzt Neptune die entsprechende Serverless-Kapazität und verwendet diese Schätzung als Mindestkapazität für die Serverless-Reader-Instance.

Neptune Serverless-Reader-Instances der Stufen 2–15 haben nicht die gleiche Einschränkung ihrer minimalen Kapazität und werden unabhängig vom Writer skaliert. Wenn sie inaktiv sind, können sie auf den minimalen NCU herunterskaliert werden, der im Kapazitätsbereich des Clusters angegeben ist. Dies kann jedoch zu Problemen führen, wenn der Lese-Workload schnell ansteigt.

Halten Sie die Lesekapazität an die Schreibkapazität im Einklang

Eine wichtige Sache, die Sie beachten sollten, ist, dass Sie sicherstellen möchten, dass Ihre Reader-Instances mit Ihrer Writer-Instance Schritt halten können, um übermäßige Verzögerungen bei der Replikation zu vermeiden. Dies ist insbesondere in zwei Situationen ein Problem, in denen Serverless-Reader-Instances nicht automatisch synchron mit der Writer-Instance skaliert werden:

- Wenn Ihr Writer bereitgestellt ist und Ihre Reader Serverless sind.
- Wenn Ihr Writer Serverless ist und Ihre Serverless-Reader den Aktionsstufen 2–15 zugeordnet sind.

Stellen Sie in beiden Fällen die Serverless-Mindestkapazität so ein, dass sie der erwarteten Schreibkapazität entspricht, um sicherzustellen, dass es bei Lesegeräten nicht zu Timeouts kommt,

die möglicherweise zu Neustarts führen. Stellen Sie im Fall einer bereitgestellten Writer-Instance die Mindestkapazität so ein, dass sie der bereitgestellten Instance entspricht. Bei einem Serverless-Writer ist die optimale Einstellung möglicherweise schwieriger vorherzusagen.

Da der Instance-Kapazitätsbereich auf Clusterebene festgelegt wird, werden alle Serverless-Instances durch dieselben Mindest- und Höchstkapazitätseinstellungen gesteuert. Reader-Instances der Stufen 0 und 1 werden synchron mit der Writer-Instance skaliert, aber Instances in den Promotion-Stufen 2-15 skalieren unabhängig voneinander und von der Writer-Instance, abhängig von ihrem Workload. Wenn Sie die Mindestkapazität zu niedrig festlegen, können inaktive Instances der Stufen 2 bis 15 zu niedrig herunterskaliert werden, um schnell genug wieder hochskaliert zu werden, um einen plötzlichen Anstieg der Schreibaktivität zu bewältigen.

Vermeiden Sie es, den Timeout-Wert zu hoch festzulegen

Es können unerwartete Kosten entstehen, wenn Sie den Wert für den Abfrage-Timeout bei Serverless-Instances zu hoch festlegen.

Ohne eine angemessene Timeout-Einstellung können Sie versehentlich eine Abfrage ausgeben, die einen leistungsstarken, teuren Instance-Typ erfordert und die sehr lange läuft, wodurch Kosten entstehen, mit denen Sie nie gerechnet haben. Sie können diese Situation vermeiden, indem Sie einen Timeout-Wert für Abfragen verwenden, der die meisten Ihrer Abfragen abdeckt und nur unerwartet lange laufende Abfragen in den Timeout versetzt.

Dies gilt sowohl für allgemeine Timeoutwerte für Abfragen, die mithilfe von Parametern festgelegt wurden, als auch für Timeoutwerte pro Abfrage, die mithilfe von Abfragehinweisen festgelegt wurden.

Optimieren Sie Ihre Neptune Serverless-Konfiguration

Wenn Ihr Neptune Serverless DB-Cluster nicht auf die Workload abgestimmt ist, die er ausführt, stellen Sie möglicherweise fest, dass er nicht optimal läuft. Sie können die Einstellung für die minimale und and/or maximale Kapazität so anpassen, dass sie skaliert werden kann, ohne dass Speicherprobleme auftreten.

- Erhöhen Sie die Einstellung für die Mindestkapazität des Clusters. Dadurch kann die Situation korrigiert werden, dass eine Instance im Leerlauf auf eine Kapazität zurückskaliert wird, die weniger Speicher hat, als Ihre Anwendung und die aktivierten Funktionen benötigen.
- Erhöhen Sie die Einstellung für die maximale Kapazität des Clusters. Auf diese Weise kann die Situation korrigiert werden, in der eine ausgelastete Datenbank nicht auf eine Kapazität

mit genügend Speicher für die Workload und alle aktivierten speicherintensiven Funktionen hochskaliert werden kann.

- Ändern Sie die Workload der betreffenden Instance. Beispielsweise können Sie dem Cluster Reader-DB-Instances hinzufügen, um die Last von schreibgeschützten Abfragen auf weitere DB-Instances zu verteilen.
- Passen Sie die Abfragen Ihrer Anwendung so an, dass sie weniger Ressourcen verbrauchen.
- Versuchen Sie, eine bereitgestellte Instance zu verwenden, die größer ist als die maximale Anzahl an NCUs, die in Neptune Serverless verfügbar sind, um herauszufinden, ob sie besser für die Speicher- und CPU-Anforderungen des Workloads geeignet ist.

Verwendung von Amazon Neptune Serverless

Sie können einen neuen Neptune-DB-Cluster als Serverless-Cluster erstellen, oder in einigen Fällen können Sie einen vorhandenen DB-Cluster auf Serverless umstellen. Sie können auch DB-Instances in einem Serverless-DB-Cluster in und aus Serverless-Instances konvertieren. Sie können Neptune Serverless nur in einem der Länder verwenden, in AWS-Regionen denen es unterstützt wird, mit einigen anderen Einschränkungen (siehe). Einschränkungen von Amazon Neptune Serverless

Sie können den Neptune- CloudFormation -Stack auch verwenden, um einen Neptune Serverless DB-Cluster zu erstellen.

Einen neuen DB-Cluster erstellen, der Serverless verwendet

Um einen Neptune DB-Cluster zu erstellen, der Serverless verwendet, können Sie die AWS-Managementkonsole auf die gleiche Weise verwenden, wie Sie es bei der Erstellung eines bereitgestellten Clusters tun. Der Unterschied besteht darin, dass Sie unter DB-Instance-Größe die DB-Instance-Klasse auf Serverless setzen müssen. Wenn Sie das tun, müssen Sie dann den Serverless-Kapazitätsbereich für den Cluster festlegen.

Sie können einen serverlosen DB-Cluster auch AWS CLI mit folgenden Befehlen erstellen (ersetzen Sie unter Windows '\' durch '^'):

```
aws neptune create-db-cluster \
  --region (an AWS-Region 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"
```

Sie könnten den `serverless-v2-scaling-configuration`-Parameter auch wie folgt angeben:

```
  --serverless-v2-scaling-configuration '{"MinCapacity":1.0, "MaxCapacity":128.0}'
```

Sie können dann den `describe-db-clusters`-Befehl für das `ServerlessV2ScalingConfiguration`-Attribut ausführen, der die von Ihnen angegebenen Einstellungen für den Kapazitätsbereich zurückgeben sollte:

```
"ServerlessV2ScalingConfiguration": {
    "MinCapacity": (the specified minimum number of NCUs),
    "MaxCapacity": (the specified maximum number of NCUs)
}
```

Einen vorhandenen DB-Cluster oder eine bestehende Instance in Serverless konvertieren

Wenn Sie einen Neptune-DB-Cluster haben, der die Engine-Version 1.2.0.1 oder höher verwendet, können Sie ihn in einen Serverless-Cluster umwandeln. Dieser Prozess verursacht einige Ausfallzeiten.

Der erste Schritt besteht darin, dem vorhandenen Cluster einen Kapazitätsbereich hinzuzufügen. Sie können dies mit dem AWS-Managementkonsole oder mit einem AWS CLI Befehl wie diesem tun (unter Windows ersetzen Sie '\' durch '^'):

```
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)
```

Der nächste Schritt besteht darin, eine neue Serverless-DB-Instance zu erstellen, um die bestehende primäre Instance (den Writer) im Cluster zu ersetzen. Auch hier können Sie diesen und alle nachfolgenden Schritte entweder mit dem AWS-Managementkonsole oder dem ausführen AWS CLI. Geben Sie in beiden Fällen die DB-Instance-Klasse als Serverless an. Der AWS CLI Befehl würde so aussehen (ersetzen Sie unter Windows '\' durch '^'):

```
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
```

Wenn die neue Writer-Instance verfügbar ist, führen Sie einen Failover durch, um sie zur Writer-Instance für den Cluster zu machen:

```
aws neptune failover-db-cluster \
  --db-cluster-identifier (ID of the DB cluster) \
  --target-db-instance-identifier (instance ID of the new serverless instance)
```

Löschen Sie als Nächstes die alte Writer-Instance:

```
aws neptune delete-db-instance \
  --db-instance-identifier (instance ID of the old writer instance) \
  --skip-final-snapshot
```

Gehen Sie abschließend genauso vor, um eine neue Serverlesss-Instance zu erstellen, die an die Stelle jeder vorhandenen bereitgestellten Reader-Instance tritt, die Sie in eine Serverless-Instance umwandeln möchten, und löschen Sie die vorhandenen bereitgestellten Instances (für Reader-Instances ist kein Failover erforderlich).

Ändern des Kapazitätsbereichs eines vorhandenen Serverless-DB-Clusters

Sie können den Kapazitätsbereich eines Neptune Serverless DB-Clusters AWS CLI wie folgt ändern (unter Windows, ersetzen Sie ‚\' durch ‚^'):

```
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
```

Durch Ändern des Kapazitätsbereichs können sich die Standardwerte einiger Konfigurationsparameter ändern. Neptune kann einige dieser neuen Standardeinstellungen sofort anwenden, aber einige der dynamischen Parameteränderungen werden erst nach einem Neustart wirksam. Status `pending-reboot` gibt an, dass ein Neustart erforderlich ist, um einige Parameteränderungen anzuwenden.

Änderung einer Serverless-DB-Instance in eine bereitgestellte

Um eine Neptune Serverless-Instance in eine bereitgestellte zu konvertieren, müssen Sie lediglich ihre Instance-Klasse in eine der bereitgestellten Instance-Klassen ändern. Siehe Ändern einer Neptune-DB-Instance (und sofortige Anwendung).

Konfiguration von Gremlin-Clients für Serverless

Wenn Sie WebSocket Gremlin-Clients mit Neptune Serverless verwenden, müssen Sie das Heartbeat-Intervall des Clients entsprechend konfigurieren, um stabile Verbindungen während Skalierungsereignissen aufrechtzuerhalten. Detaillierte Konfigurationsanweisungen für Java-JavaScript/Node.js, Go- und Python-Clients finden Sie unterHeartbeat-Konfiguration für Neptune Serverless.

Überwachung serverloser Kapazitäten mit Amazon CloudWatch

Sie können CloudWatch es verwenden, um die Kapazität und Auslastung der serverlosen Neptune-Instances in Ihrem DB-Cluster zu überwachen. Es gibt zwei CloudWatch Metriken, mit denen Sie die aktuelle serverlose Kapazität sowohl auf Cluster- als auch auf Instance-Ebene verfolgen können:

- **`ServerlessDatabaseCapacity`** – Als Metrik auf Instance-Ebene wird die aktuelle Instance-Kapazität in NCUs gemeldet von `ServerlessDatabaseCapacity`. Als Metrik auf Clusterebene gibt sie den Durchschnitt der `ServerlessDatabaseCapacity`-Werte aller DB-Instances im Cluster an.
- **`NCUUtilization`** – Diese Metrik gibt an, wie viel Prozent der möglichen Kapazität genutzt wird. Sie wird berechnet als der aktuelle `ServerlessDatabaseCapacity` (entweder auf Instance-Ebene oder auf Cluster-Ebene) geteilt durch die maximale Kapazitätseinstellung für den DB-Cluster.

  Wenn sich diese Metrik auf Clusterebene 100 % nähert, was bedeutet, dass der Cluster so hoch wie möglich skaliert wurde, sollten Sie eine Erhöhung der Einstellung für die maximale Kapazität in Betracht ziehen.

  Wenn sie sich für eine Reader-Instance 100 % nähert, während die Writer-Instance nicht annähernd die maximale Kapazität erreicht, sollten Sie erwägen, weitere Reader-Instances hinzuzufügen, um den Lese-Workload zu verteilen.

Beachten Sie, dass die Metriken `CPUUtilization` und `FreeableMemory` für Serverless-Instances eine etwas andere Bedeutung haben als für bereitgestellte Instances. Ist in einem Serverless-Kontext ist `CPUUtilization` ein Prozentsatz, der als der aktuell CPU-Verbrauch dividiert durch die CPU-Verbrauch dividiert durch die Menge der CPU berechnet wird, die bei maximaler Kapazität verfügbar wäre. `FreeableMemory` meldet in ähnlicher Weise die Menge an freiem Speicher, die verfügbar wäre, wenn eine Instance die maximale Kapazität erreicht hätte.

Das folgende Beispiel zeigt, wie Sie AWS CLI unter Linux die minimalen, maximalen und durchschnittlichen Kapazitätswerte für eine bestimmte DB-Instance abrufen können, gemessen alle 10 Minuten über eine Stunde. Der Linux-Befehl `date` gibt die Start- und Endzeiten relativ zum aktuellen Datum und zur aktuellen Uhrzeit an. Die `sort_by`-Funktion im `--query`-Parameter sortiert die Ergebnisse chronologisch basierend auf dem Feld `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
```

Amazon Neptune mit einer globalen Datenbank verwenden

Eine globale Amazon Neptune Neptune-Datenbank erstreckt sich über mehrere AWS-Regionen, ermöglicht globale Lesevorgänge mit geringer Latenz und ermöglicht eine schnelle Wiederherstellung in den seltenen Fällen, in denen ein Ausfall eine gesamte Datenbank betrifft. AWS-Region

Eine globale Neptune-Datenbank hat einen primären DB-Cluster in einer Region und bis zu fünf sekundäre DB-Cluster in verschiedenen Regionen.

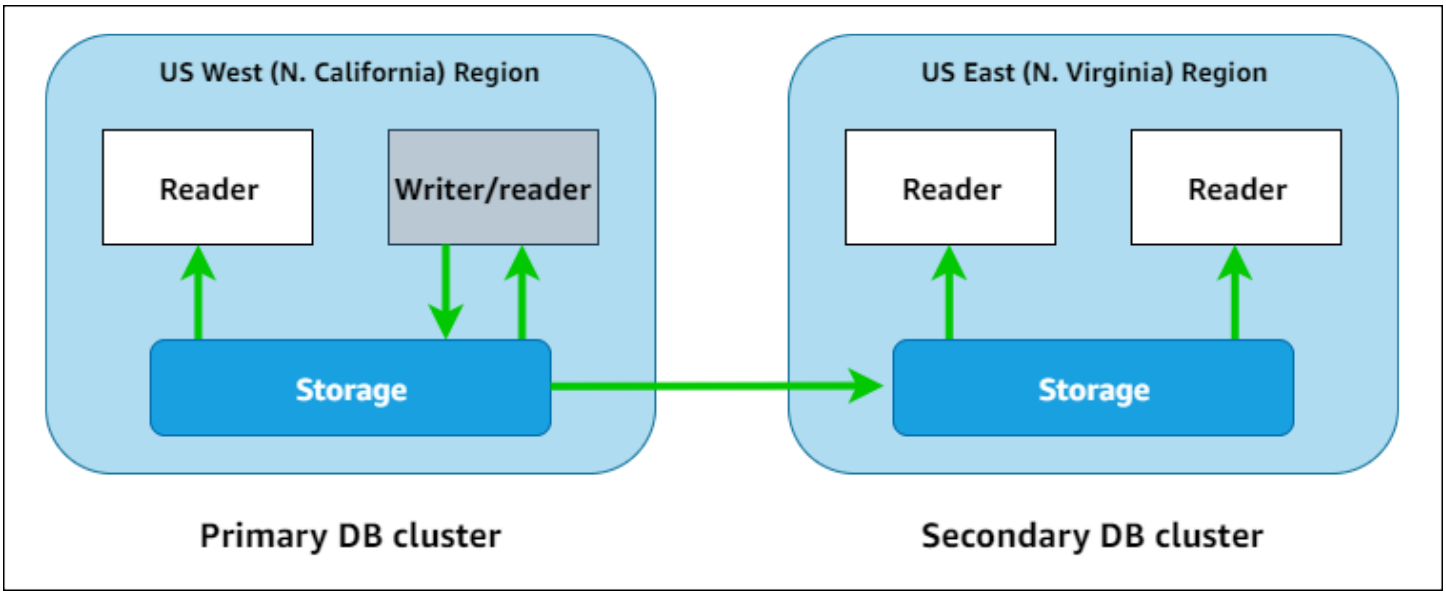
Schreibvorgänge können nur in der primären Region erfolgen. Sekundäre Regionen unterstützen nur Lesevorgänge. Jede sekundäre Region kann bis zu 16 Leser-Instances haben.

Globale Datenbanken in Amazon Neptune

Durch die Verwendung einer globalen Neptune-Datenbank können Sie Ihre global verteilten Anwendungen mit einer einzigen Datenbank ausführen, die mehrere AWS-Regionen umfasst.

Eine globale Neptune-Datenbank besteht aus einem DB-Cluster in einer primären AWS-Region , wo die Daten geschrieben werden, und bis zu fünf schreibgeschützten DB-Clustern in sekundären AWS-Regionen. Wenn Sie einen Schreibvorgang auf dem primären DB-Cluster durchführen, repliziert Neptune die geschriebenen Daten über eine dedizierte Infrastruktur auf alle sekundären DB-Cluster mit einer Latenz von in der Regel weniger als einer Sekunde.

Das folgende Diagramm zeigt ein Beispiel für eine globale Datenbank, die sich über zwei Bereiche erstreckt: AWS-Regionen



Sie können jeden sekundären Cluster zur Verarbeitung von Nur-Lese-Workloads unabhängig skalieren, indem Sie eine oder mehrere Lesereplikat-Instances hinzufügen.

Für Schreibvorgänge müssen Sie eine Verbindung mit dem DB-Cluster-Endpunkt des primären DB-Clusters herstellen. Nur der primäre Cluster kann Schreibvorgänge ausführen. Anschließend erfolgt die Replikation, wie im obigen Diagramm dargestellt, vom Cluster-Speicher-Volume und nicht von der Datenbank-Engine aus.

Globale Neptune-Datenbanken sind für Anwendungen mit weltweiter Präsenz konzipiert. Die schreibgeschützten sekundären DB-Cluster unterstützen Lesevorgänge näher an den Anwendungsbenutzern.

Eine globale Neptune-Datenbank unterstützt zwei verschiedene Failover-Ansätze:

- Verwenden Sie zur Wiederherstellung nach einem Ausfall in der primären Region den manuellen ungeplanten detach-and-promote Prozess, bei dem Sie einen der sekundären Cluster trennen, ihn in einen eigenständigen Cluster umwandeln und ihn dann zum neuen primären Cluster heraufstufen.
- Verwenden Sie für geplante Betriebsabläufe wie etwa Wartungsmaßnahmen das verwaltete geplante Failover, bei dem Sie den primären Cluster ohne Datenverlust in eine seiner sekundären Regionen verschieben.

Vorteile der Verwendung globaler Datenbanken in Amazon Neptune

Durch die Verwendung einer globalen Datenbank profitieren Sie von folgenden Vorteilen:

- Globale Lesevorgänge mit lokaler Latenz – Wenn Sie Niederlassungen auf der ganzen Welt haben, ermöglicht eine globale Datenbank den Zugriff auf Daten in ihrer eigenen Region mit lokaler Latenz.
- Skalierbare sekundäre Neptune-DB-Cluster – Sie können Ihre sekundären Cluster skalieren, indem Sie Lesereplikat-DB-Cluster hinzufügen. Da sekundäre Cluster schreibgeschützt sind, können sie jeweils bis zu 16 Lesereplikate anstelle des üblichen Limits von 15 unterstützen.
- Schnelle Replikation zu sekundären DB-Clustern – Die Replikation von primären zu sekundären DB-Clustern erfolgt schnell, mit einer Latenz von in der Regel weniger als einer Sekunde und mit geringen Leistungseinbußen auf dem primären DB-Cluster. Da die Replikation auf Speicherebene

durchgeführt wird, stehen DB-Instance-Ressourcen vollständig für Lese- und Schreib-Workloads von Anwendungen zur Verfügung.

- Wiederherstellung nach regionsweiten Ausfällen – Die sekundären DB-Cluster ermöglichen Ihnen, den primären Cluster schneller zu einer neuen Region zu verschieben, mit geringeren RTO- und RPO-Werten als bei herkömmlichen Replikationslösungen.

Einschränkungen bei globalen Datenbanken in Amazon Neptune

Für globale -Datenbanken gelten aktuell die folgenden Einschränkungen:

- Globale Neptune-Datenbanken sind nur in den folgenden AWS-Regionen verfügbar:
  - USA Ost (Nord-Virginia): `us-east-1`
  - USA Ost (Ohio): `us-east-2`
  - USA West (Nordkalifornien): `us-west-1`
  - USA West (Oregon): `us-west-2`
  - Kanada West (Calgary): `ca-west-1`
  - Europa (Spanien): `eu-south-2`
  - Europa (Irland): `eu-west-1`
  - Europa (London): `eu-west-2`
  - Europa (Frankfurt): `eu-central-1`
  - Asien-Pazifik (Tokio): `ap-northeast-1`
  - Asien-Pazifik (Osaka): `ap-northeast-3`
  - Asien-Pazifik (Singapur): `ap-southeast-1`
  - Asien-Pazifik (Jakarta): `ap-southeast-3`
  - Asien-Pazifik (Melbourne): `ap-southeast-4`
  - Asien-Pazifik (Malaysia): `ap-southeast-5`
  - Israel (Tel Aviv): `il-central-1`
- Neptune Global Databases unterstützt keine `db.t3.medium` `db.t4g.medium` Instanztypen.
- Globale Neptune-Datenbanken unterstützen kein Auto Scaling für sekundäre DB-Cluster.
- Sie können keine benutzerdefinierte Parametergruppe auf den globalen Datenbank-Cluster anwenden, während Sie ein Hauptversions-Upgrade der jeweiligen globalen Datenbank durchführen. Erstellen Sie stattdessen Ihre benutzerdefinierten Parametergruppen in jeder Region

des globalen Clusters und wenden Sie sie nach dem Upgrade manuell auf die regionalen Cluster an.

- Sie können die DB-Cluster in einer globalen Datenbank nicht einzeln anhalten oder starten.
- Read-Replica-Instances in einem sekundären DB-Cluster können unter bestimmten Umständen neu gestartet werden, einschließlich geplanter Upgrades während Ihres Wartungsfensters. Wenn die Writer-DB-Instance des primären Clusters neu gestartet oder ein Failover durchgeführt wird, werden alle Instances in sekundären Regionen ebenfalls neu gestartet. Der sekundäre Cluster ist erst dann wieder verfügbar, wenn alle seine Instances wieder mit der Writer-Instance des primären DB-Clusters synchronisiert sind.

Einrichtung einer globalen Amazon Neptune Neptune-Datenbank

Sie können eine globale Neptune-Datenbank auf eine der folgenden Arten erstellen:

- Erstellen Sie eine globale Datenbank mit neuen DB-Clustern und Instances.
- Erstellen Sie eine globale Datenbank mit einem vorhandenen Neptune-DB-Cluster als primärem Cluster.

Themen

- Konfigurationsanforderungen für eine globale Datenbank in Amazon Neptune
- Verwenden von AWS CLI , um eine globale Datenbank in Amazon Neptune zu erstellen
- Umwandlung eines vorhandenen DB-Clusters in eine globale Datenbank
- Hinzufügen sekundärer globaler Datenbankregionen zu einer primären Region in Amazon Neptune
- Herstellen einer Verbindung mit einer globalen Neptune-Datenbank

Konfigurationsanforderungen für eine globale Datenbank in Amazon Neptune

Eine globale Neptun-Datenbank umfasst mindestens zwei. AWS-Regionen Die primäre AWS-Region enthält einen Neptune-DB-Cluster mit einer Writer-Instance. Eine bis fünf sekundäre AWS-Regionen enthalten jeweils einen schreibgeschützten Neptune-DB-Cluster, der vollständig aus Lesereplikat-Instances besteht. Es ist mindestens eine AWS-Region Sekundärseite erforderlich.

Die Neptune-DB-Cluster, aus denen eine globale Datenbank besteht, haben die folgenden spezifischen Anforderungen:

- DB-Instance-Klassenanforderungen – Eine globale Datenbank erfordert Die DB-Instance-Klasse `r5` oder `r6g`, die für speicherintensive Workloads optimiert sind, wie z. B. einen `db.r5.large`-Instance-Typ.
- AWS-Region Anforderungen — Eine globale Datenbank benötigt einen primären Neptune-DB-Cluster in einem AWS-Region und mindestens einen sekundären Neptune-DB-Cluster in einer anderen Region. Sie können bis zu fünf sekundäre schreibgeschützte Neptune-DB-Cluster erstellen und jeder davon muss sich in einer anderen Region befinden. Mit anderen Worten: Es können sich keine zwei Neptune-DB-Cluster in derselben AWS-Region befinden.
- Anforderungen an die Engine-Version – Die Neptune-Engine-Version, die von allen DB-Clustern in der globalen Datenbank verwendet wird, sollte identisch sein und muss größer oder gleich `1.2.0.0` sein. Wenn Sie bei der Erstellung einer neuen globalen Datenbank, eines Clusters oder einer Instance die Engine-Version nicht angeben, wird die neueste Engine-Version verwendet.

> **Important**
>
> Obwohl die DB-Cluster-Parametergruppen unabhängig für jeden DB-Cluster in einer globalen Datenbank konfiguriert werden können, empfiehlt es sich, die Einstellungen in allen Clustern konsistent zu halten, um unerwartete Verhaltensänderungen zu vermeiden, wenn ein sekundärer Cluster zum primären Cluster heraufgestuft werden muss. Verwenden Sie beispielsweise die gleichen Einstellungen für Objektindizes, Streams usw. in allen DB-Clustern.

Verwenden von AWS CLI , um eine globale Datenbank in Amazon Neptune zu erstellen

> **Note**
>
> Die Beispiele in diesem Abschnitt folgen der UNIX-Konvention mit einem umgekehrten Schrägstrich (\) als Zeilenerweiterungszeichen. Ersetzen Sie unter Windows den umgekehrten Schrägstrich durch ein Caret-Zeichen (^).

Um eine globale Datenbank mit dem zu erstellen AWS CLI

1. Erstellen Sie zunächst mit dem `create-global-cluster` AWS CLI Befehl (der die CreateGlobalCluster API umschließt) eine leere globale Datenbank. Geben Sie den Namen der AWS-Region ,an, die Sie als primäre Region verwenden möchten, legen Sie Neptune als Datenbank-Engine fest und geben Sie optional die Engine-Version an, die Sie verwenden möchten (dies muss Version 1.2.0.0 oder höher sein):

```
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. Es kann einige Minuten dauern, bis die globale Datenbank verfügbar ist. Bevor Sie mit dem nächsten Schritt fortfahren, überprüfen Sie mit dem `describe-global-clusters`-CLI-Befehl (der die DescribeGlobalClusters-API umschließt), ob die globale Datenbank verfügbar ist:

```
aws neptune describe-global-clusters \
  --region (primary region) \
  --global-cluster-identifier (global database ID)
```

3. Sobald die globale Neptune-Datenbank verfügbar ist, können Sie einen neuen Neptune-DB-Cluster als primären Cluster erstellen:

```
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. Verwenden Sie den `describe-db-clusters` AWS CLI Befehl, um zu bestätigen, dass der neue DB-Cluster bereit ist, damit Sie seine primäre DB-Instance hinzufügen können:

```
aws neptune describe-db-clusters \
  --region (primary region) \
  --db-cluster-identifier (primary DB cluster ID)
```

Wenn die Antwort den Status `"Status": "available"` anzeigt, fahren Sie mit dem nächsten Schritt fort.

5. Erstellen Sie mit dem `create-db-instance` AWS CLI Befehl die primäre DB-Instance für den primären Cluster. Sie müssen einen der speicheroptimierten Instance-Typen `r5` oder `r6g` verwenden, z. B. `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**
>
> Wenn Sie planen, dem neuen primären DB-Cluster mithilfe des Neptune-Bulk-Loaders Daten hinzuzufügen, tun Sie dies, bevor Sie sekundäre Regionen hinzufügen. Dies ist schneller und kostengünstiger als das Durchführen eines Massenladevorgangs, nachdem die globale Datenbank vollständig eingerichtet ist.

Fügen Sie nun der neuen globalen Datenbank eine oder mehrere sekundäre Regionen hinzu, wie unter Hinzufügen einer sekundären Region mit dem AWS CLI beschrieben.

Umwandlung eines vorhandenen DB-Clusters in eine globale Datenbank

Um einen vorhandenen DB-Cluster in eine globale Datenbank umzuwandeln, verwenden Sie den `create-global-cluster` AWS CLI Befehl, um eine neue globale Datenbank in derselben AWS-Region Position zu erstellen, in der sich der vorhandene DB-Cluster befindet, und setzen Sie ihren `--source-db-cluster-identifier` Parameter auf den Amazon-Ressourcennamen (ARN) des vorhandenen Clusters, der sich dort befindet:

```
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)
```

Fügen Sie nun der neuen globalen Datenbank eine oder mehrere sekundäre Regionen hinzu, wie unter Hinzufügen einer sekundären Region mit dem AWS CLI beschrieben.

Verwenden eines aus einem Snapshot wiederhergestellten DB-Clusters als primärer Cluster

Sie können einen aus einem Snapshot wiederhergestellten DB-Cluster in eine globale Neptune-Datenbank umwandeln. Machen Sie nach Abschluss der Wiederherstellung den dadurch erstellten DB-Cluster wie oben beschrieben zum primären Cluster einer neuen globalen Datenbank.

Hinzufügen sekundärer globaler Datenbankregionen zu einer primären Region in Amazon Neptune

Eine globale Neptune-Datenbank benötigt mindestens einen sekundären Neptune-DB-Cluster in einem anderen AWS-Region als dem primären DB-Cluster. Sie können bis zu fünf sekundäre DB-Cluster an den primären DB-Cluster anfügen.

Jeder sekundäre DB-Cluster, den Sie hinzufügen, reduziert die maximale Anzahl von Lesereplikat-Instances, die Sie auf dem primären Cluster haben können, um einen. Wenn es beispielsweise 4 sekundäre Cluster gibt, beträgt die maximale Anzahl von Lesereplikat-Instances, die Sie auf dem primären Cluster haben können, `15 - 4 = 11`. Wenn Sie zum Beispiel 14 Reader-Instances im primären DB-Cluster und einen sekundären Cluster haben, können Sie keinen weiteren sekundären Cluster hinzufügen.

Verwenden von AWS CLI , um einer globalen Datenbank in Neptune eine sekundäre Region hinzuzufügen

Um eine Sekundärdatei AWS-Region zu einer globalen Neptune-Datenbank hinzuzufügen, verwenden Sie den AWS CLI

1. Verwenden Sie den `create-db-cluster` AWS CLI Befehl, um ein neues DB-Cluster in einer anderen Region als der Ihres primären Clusters zu erstellen, und legen Sie seinen `--global-cluster-identifier` Parameter fest, um die ID der globalen Datenbank anzugeben:

   ```
   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. Verwenden Sie den `describe-db-clusters` AWS CLI Befehl, um zu bestätigen, dass der neue DB-Cluster bereit ist, damit Sie seine primäre DB-Instance hinzufügen können:

```
aws neptune describe-db-clusters \
  --region (primary region) \
  --db-cluster-identifier (primary DB cluster ID)
```

   Wenn die Antwort den Status `"Status": "available"` anzeigt, fahren Sie mit dem nächsten Schritt fort.

3. Erstellen Sie die primäre DB-Instance für den primären Cluster mithilfe des `create-db-instance` AWS CLI Befehls und verwenden Sie dabei einen Instance-Typ in der `r6g` Instance-Klasse `r5` oder:

```
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**
>
> Wenn Sie nicht beabsichtigen, eine große Anzahl von Leseanfragen in der sekundären Region zu bearbeiten, und Ihnen hauptsächlich wichtig ist, Ihre Daten dort zuverlässig zu sichern, können Sie einen sekundären DB-Cluster ohne DB-Instances erstellen. Dies spart Geld, da Sie dann nur für den Speicher des sekundären Clusters zahlen, den Neptune mit dem Speicher im primären DB-Cluster synchronisiert.

Herstellen einer Verbindung mit einer globalen Neptune-Datenbank

Wie Sie eine Verbindung mit einer globalen Neptune-Datenbank herstellen, hängt davon ab, ob Sie in die Datenbank schreiben oder aus der Datenbank lesen müssen:

- Bei Nur-Lese-Anfragen oder Abfragen stellen Sie eine Verbindung zum Reader-Endpunkt des Aurora-Clusters in Ihrer AWS-Region her.
- Um Mutationsabfragen auszuführen, stellen Sie eine Verbindung zum Cluster-Endpunkt für den primären DB-Cluster her, der sich möglicherweise in einer anderen AWS-Region als Ihrer Anwendung befindet.

Verwalten einer globalen Amazon-Neptune-Datenbank

Mit Ausnahme des verwalteten geplanten Failover-Prozesses führen Sie die meisten Verwaltungsvorgänge auf den einzelnen Clustern durch, aus denen eine globale Neptune-Datenbank besteht. Der verwaltete geplante Failover-Prozess steht nur globalen Neptune-Datenbanken zur Verfügung, nicht einzelnen Neptune-DB-Clustern. Weitere Informationen hierzu finden Sie unter Ausführen von verwalteten geplanten Failovers für globale Neptune-Datenbanken.

Informationen zum Wiederherstellen einer globalen Neptune-Datenbank nach einem ungeplanten Ausfall in ihrer primären Region finden Sie unter Detach-and-promote eine globale Neptun-Datenbank für den Fall eines ungeplanten Ausfalls.

Obwohl die DB-Cluster-Parametergruppen unabhängig für jeden DB-Cluster in einer globalen Datenbank konfiguriert werden können, empfiehlt es sich, die Einstellungen in allen Clustern konsistent zu halten, um unerwartete Verhaltensänderungen zu vermeiden, wenn ein sekundärer Cluster zum primären Cluster heraufgestuft werden muss. Verwenden Sie beispielsweise die gleichen Einstellungen für Objektindizes, Streams usw. in allen DB-Clustern.

Entfernen eines DB-Clusters aus einer globalen Neptune-Datenbank

Es gibt verschiedene Gründe, aus denen Sie einen DB-Cluster aus einer globalen Datenbank entfernen möchten. Beispiel:

- Wenn der primäre Cluster heruntergestuft oder isoliert wird, können Sie ihn aus der globalen Datenbank entfernen und er wird zu einem eigenständigen bereitgestellten Cluster, der zum Erstellen einer neuen globalen Datenbank verwendet werden kann (siehe Detach-and-promote eine globale Neptun-Datenbank für den Fall eines ungeplanten Ausfalls).
- Sie müssen zunächst alle zugehörigen Cluster entfernen, wobei der primäre Cluster zuletzt entfernt (abgetrennt) wird (siehe Löschen einer globalen Neptune-Datenbank).

Sie können den remove-from-global-clusterCLI-Befehl (der die RemoveFromGlobalCluster API umschließt) verwenden, um einen Neptune-DB-Cluster von einer globalen Datenbank zu trennen:

```
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)
```

Der getrennte DB-Cluster wird dann zu einem eigenständigen DB-Cluster.

Löschen einer globalen Neptune-Datenbank

Sie können eine globale Datenbank und ihre zugehörigen Cluster nicht in einem einzigen Schritt löschen. Stattdessen müssen Sie ihre Komponenten einzeln löschen:

1. Trennen Sie alle sekundären DB-Cluster von der globalen Datenbank, wie unter Entfernen eines Clusters beschrieben. Sie können sie jetzt einzeln löschen, wenn Sie dies möchten.
2. Trennen Sie den primären DB-Cluster von der globalen Datenbank.
3. Löschen Sie alle Lesereplikat-DB-Instances aus dem primären Cluster.
4. Löschen Sie die primäre (Writer-)DB-Instance aus dem primären Cluster. Wenn Sie dies auf der Konsole tun, wird auch der DB-Cluster gelöscht.
5. Löschen Sie die globale Datenbank selbst. Verwenden Sie dazu den AWS CLIdelete-global-clusterCLI-Befehl (der die DeleteGlobalCluster API umschließt) wie folgt:

   ```
   aws neptune delete-global-cluster \
     --region (region of the DB cluster to delete) \
     --global-cluster-identifier (global database ID)
   ```

Ändern einer globalen Neptune-Datenbank

Obwohl die DB-Cluster-Parametergruppen unabhängig für jeden Neptune-DB-Cluster in einer globalen Datenbank konfiguriert werden können, empfiehlt es sich, die Einstellungen in allen Clustern konsistent zu halten, um unerwartete Verhaltensänderungen zu vermeiden, wenn ein sekundärer Cluster zum primären Cluster heraufgestuft werden muss.

Sie können die Einstellungen der globalen Datenbank selbst mit dem modify-global-clusterCLI-Befehl (der die ModifyGlobalCluster API umschließt) ändern. Sie könnten beispielsweise die globale Datenbank-ID ändern und gleichzeitig den Löschschutz wie folgt deaktivieren:

```
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
```

Durchführung der Notfallwiederherstellung Amazon Neptune

Eine globale Neptune-Datenbank bietet umfassendere Failover-Funktionen als ein eigenständiger Neptune-DB-Cluster. Durch die Verwendung einer globalen Datenbank können Sie relativ schnell für einen Notfall planen und danach eine Wiederherstellung durchführen. Die Notfallwiederherstellung wird in der Regel anhand einer Bewertung des Recovery Time Objective (RTO) und des Recovery Point Objective (RPO) bewertet:

- Recovery Time Objective (RTO) – So schnell kehrt ein System nach einem Notfall in einen arbeitsfähigen Zustand zurück. Mit anderen Worten: RTO misst die Ausfallzeit. Für eine globale Neptune-Datenbank kann sich die RTO in einer Größenordnung von Minuten bewegen.
- Recovery-Point-Objective (RPO) – Die Zeitspanne, während der Daten verloren gehen. Für eine globale Neptune-Datenbank wird der RPO-Wert in der Regel in Sekunden gemessen (siehe Ausführen von verwalteten geplanten Failovers für globale Neptune-Datenbanken).

Für eine globale Neptune-Datenbank gibt es zwei verschiedene Failover-Konzepte:

- Detach-and-promote (manuelle ungeplante Wiederherstellung) — Um die Daten nach einem ungeplanten Ausfall wiederherzustellen oder Notfallwiederherstellungstests (DR-Tests) durchzuführen, führen Sie einen regionsübergreifenden Vorgang detach-and-promote auf einem der sekundären DB-Cluster in der globalen Datenbank durch. Die RTO für diesen manuellen Prozess ist davon abhängig, wie schnell Sie die unter aufgeführten Aufgaben ausführen könne Trennen und Hochstufen. Der RPO-Wert wird in der Regel in Sekunden gemessen, dies hängt jedoch von der Verzögerung der Speicherreplikation im gesamten Netzwerk zum Zeitpunkt des Ausfalls ab.
- Verwaltetes geplantes Failover – Dieser Ansatz ist für die betriebliche Wartung und andere geplante Betriebsabläufe vorgesehen, z. B. für die Verlagerung des primären DB-Clusters der globalen Datenbank in eine der sekundären Regionen. Da dieser Prozess die sekundären DB-Cluster mit dem primären synchronisiert, bevor andere Änderungen vorgenommen werden, ist der

RPO-Wert effektiv 0 (d. h. kein Datenverlust). Siehe Ausführen von verwalteten geplanten Failovers für globale Neptune-Datenbanken.

Detach-and-promote eine globale Neptun-Datenbank für den Fall eines ungeplanten Ausfalls

In der sehr seltenen Situation, in der Ihre globale Neptune-Datenbank einen unerwarteten Ausfall in ihrer Primärdatenbank erlebt AWS-Region, sind Ihr primärer Neptune-DB-Cluster und sein Writer-Knoten nicht mehr verfügbar und die Replikation zwischen dem primären Cluster und den sekundären Clustern wird eingestellt. Um sowohl die daraus resultierenden Ausfallzeiten (RTO) als auch den Datenverlust (RPO) zu minimieren, sollten Sie schnell eine regionsübergreifende Rekonstruktion der globalen Datenbank durchführen. detach-and-promote

> **Tip**
>
> Sie sollten diesen Prozess verstehen, bevor Sie ihn anwenden, und einen Plan haben, um beim ersten Anzeichen eines regionsweiten Problems schnell vorgehen zu können.
>
> - Verwenden Sie Amazon CloudWatch regelmäßig, um die Verzögerungszeiten für die sekundären Cluster zu verfolgen, sodass Sie die sekundäre Region mit der geringsten Verzögerungszeit identifizieren können, falls Sie ein Failover benötigen.
> - Testen Sie Ihren Plan, um zu überprüfen, ob Ihre Verfahren vollständig und korrekt sind.
> - Verwenden Sie eine simulierte Umgebung, um sicherzustellen, dass Ihre Mitarbeiter geschult und bereit sind, ein DR-Failover schnell durchzuführen, falls dies jemals erforderlich sein sollte.

Führen Sie nach einem ungeplanten Ausfall in der primären Region ein Failover auf einen sekundären Cluster wie folgt durch:

1. Beenden Sie die Ausgabe von Mutationsabfragen und anderen Schreibvorgängen auf dem primären DB-Cluster.
2. Identifizieren Sie einen DB-Cluster in einem sekundären AWS-Region Bereich, der als neuer primärer DB-Cluster der globalen Datenbank verwendet werden soll. Wenn Sie zwei oder mehr sekundäre AWS-Regionen in Ihrer globalen -Datenbank haben, wählen Sie den sekundären Cluster mit der geringsten Verzögerungszeit.

3. Trennen Sie den sekundären DB-Cluster, den Sie ausgewählt haben, von der globalen Neptune-Datenbank.

   Durch das Entfernen eines sekundären DB-Clusters aus einer globalen Neptune-Datenbank wird sofort die Replikation von Daten vom primären zum sekundären DB-Cluster gestoppt und zu einem eigenständigen DB-Cluster mit vollem read/write Funktionsumfang heraufgestuft. Alle anderen sekundären Cluster in der globalen Datenbank sind weiterhin verfügbar und können Leseaufrufe aus Ihrer Anwendung annehmen.

   Bevor Sie die globale Neptune-Datenbank neu erstellen, müssen Sie auch die anderen sekundären Cluster trennen, um Dateninkonsistenzen zwischen den Clustern zu vermeiden (siehe Entfernen eines Clusters).

4. Konfigurieren Sie Ihre Anwendung neu, um alle Schreibvorgänge mit ihrem neuen Endpunkt an den eigenständigen Neptune-DB-Cluster zu senden, den Sie als neuen primären Cluster ausgewählt haben. Wenn Sie die Standard-Namen beim Erstellen der globalen Neptune-Datenbank akzeptiert haben, können Sie den Endpunkt ändern, indem Sie den `-ro` aus der Endpunkt-Zeichenfolge des Clusters in Ihrer Anwendung entfernen.

   Beispielsweise wird der Endpunkt `my-global.cluster-`**`ro`**`-aaaaaabbbbbb.us-west-1.neptune.amazonaws.com` des sekundären Clusters zu `my-global.cluster-aaaaaabbbbbb.us-west-1.neptune.amazonaws.com`, wenn dieser Cluster von der globalen Datenbank getrennt wird.

   Dieser Neptune-DB-Cluster wird zum primären Cluster einer neuen globalen Neptune-Datenbank, wenn Sie im nächsten Schritt beginnen, Regionen hinzuzufügen.

5. Fügen Sie dem AWS-Region DB-Cluster einen hinzu. Wenn Sie dies tun, beginnt der Replikationsprozess vom primären zum sekundären Cluster. Siehe Hinzufügen sekundärer globaler Datenbankregionen zu einer primären Region in Amazon Neptune.

6. Fügen Sie nach AWS-Regionen Bedarf weitere hinzu, um die Topologie wiederherzustellen, die zur Unterstützung Ihrer Anwendung erforderlich ist.

Stellen Sie sicher, dass Schreibvorgänge der Anwendung vor, während und nach diesen Änderungen an den korrekten Neptune-DB-Cluster gesendet werden. Dadurch werden Dateninkonsistenzen zwischen den DB-Clustern in der globalen Neptune-Datenbank vermieden (so genannte „Split-brain-Probleme“).

Ausführen von verwalteten geplanten Failovers für globale Neptune-Datenbanken

Mit Managed Planned Failover können Sie den primären Cluster Ihrer globalen Neptune-Datenbank jederzeit in einen anderen AWS-Region verschieben. Manche Organisationen werden ihre primären Cluster-Standorte regelmäßig verlagern wollen.

> **Note**
>
> Ein verwaltetes geplantes Failover ist für die Verwendung in einer fehlerfreien globalen Neptune-Datenbank konzipiert. Um nach einem ungeplanten Ausfall eine Wiederherstellung durchzuführen oder Notfallwiederherstellung (DR-)Tests durchzuführen, befolgen Sie stattdessen den Trennen-und-Hochstufen-Prozess.

Während eines verwalteten geplanten Failovers erfolgt ein Failover Ihres primären Clusters zu einer von Ihnen gewählten sekundären Region, während die vorhandene Replikationstopologie Ihrer globalen Datenbank beibehalten wird. Bevor der verwaltete geplante Failover-Prozess beginnt, synchronisiert die globale Datenbank alle sekundären Cluster mit ihrem primären Cluster. Nachdem sichergestellt wurde, dass alle Cluster synchronisiert wurden, beginnt das verwaltete geplante Failover. Der DB-Cluster in der primären Region wird schreibgeschützt und der gewählte sekundäre Cluster stuft eine seiner schreibgeschützten Instances auf den Status eines vollständigen Writers hoch, so dass der Cluster die Rolle des primären Clusters übernehmen kann. Da alle sekundären Cluster zu Beginn des Prozesses mit dem primären Cluster synchronisiert wurden, setzt der neue primäre Cluster den Betrieb für die globale Datenbank fort, ohne dass Daten verloren gehen.
Die Datenbank ist nur für kurze Zeit nicht verfügbar, während der primäre und der ausgewählte sekundäre Cluster ihre neuen Rollen übernehmen.

Um die Anwendungsverfügbarkeit zu optimieren, können Sie den Failover-Vorgang außerhalb der Spitzenzeiten durchführen, zu einer Zeit, in der nur minimale Schreibvorgänge in den primären DB-Cluster stattfinden. Führen Sie außerdem die folgenden Schritte aus, bevor Sie den Failover-Vorgang starten:

- Schalten Sie Anwendungen wann immer möglich offline, um die Anzahl der Schreibvorgänge auf den primären Cluster zu reduzieren.
- Überprüfen Sie die Verzögerungszeiten für alle sekundären Neptune-DB-Cluster in der globalen Datenbank und wählen Sie den sekundären Cluster mit der geringsten

Gesamtverzögerungszeit als primären Cluster aus. Verwenden Sie Amazon CloudWatch , um die `NeptuneGlobalDBProgressLag` Metrik für alle sekundären Geräte anzuzeigen. Diese Metrik gibt an, wie weit (in Millisekunden) ein sekundärer Cluster gegenüber dem primären DB-Cluster verzögert ist. Der Wert verhält sich direkt proportional zu der Zeit, die Neptune zum Abschließen des Failovers benötigt. Anders ausgedrückt: Je größer der Verzögerungswert ist, umso länger dauert der Failover-Ausfall. Sie sollten daher die Region mit der geringsten Verzögerung wählen. Weitere Informationen finden Sie unter Neptun-Metriken CloudWatch.

Während eines verwalteten geplanten Failovers wird der ausgewählte sekundäre DB-Cluster zu seiner neuen Rolle als primärer Cluster hochgestuft, übernimmt aber nicht die vollständige Konfiguration des primären DB-Clusters. Wenn die Konfiguration nicht übereinstimmt, kann dies Leistungsprobleme, Workload-Inkompatibilitäten und anderes anomales Verhalten zur Folge haben. Um solche Probleme zu vermeiden, sollten Sie vor dem Failover die folgenden Arten von Konfigurationsunterschieden zwischen globalen Datenbank-Clustern beheben:

- Konfigurieren Sie die Parameter in dem neuen primären Cluster so, dass sie mit denen des aktuellen primären Clusters übereinstimmen.
- Konfigurieren von Überwachungs-Tools, -Optionen und -Alarmen – Konfigurieren Sie den DB-Cluster, der der neue primäre Cluster sein wird, mit den gleichen Protokollierungsfunktionen, Alarmen usw., die auch der aktuelle primäre Cluster hat.
- Integrationen mit anderen AWS Diensten konfigurieren — Wenn Ihre globale Neptune-Datenbank in AWS Dienste wie AWS Identity and Access Management (IAM), Amazon S3 oder AWS Lambda integriert ist, stellen Sie sicher, dass diese für die Integration mit dem neuen primären DB-Cluster nach Bedarf konfiguriert sind.

Wenn der Failover-Prozess abgeschlossen und der hochgestufte DB-Cluster bereit ist, Schreibvorgänge für die globale Datenbank abzuwickeln, stellen Sie sicher, dass Sie Ihre Anwendungen so ändern, dass sie den neuen Endpunkt für den neuen primären Server verwenden.

Verwenden von, um ein verwaltetes geplantes AWS CLI Failover einzuleiten

Verwenden Sie den failover-global-clusterCLI-Befehl (der die FailoverGlobalCluster API umschließt), um ein Failover Ihrer globalen Neptune-Datenbank durchzuführen:

```
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)
```

Überwachen Sie Ihre globale Amazon Neptune Neptune-Datenbank mithilfe von Metriken CloudWatch

Neptune unterstützt die folgenden CloudWatch Metriken, mit denen Sie eine globale Neptune-Datenbank überwachen können:

- **`GlobalDbDataTransferBytes`**— Die Anzahl der Byte an Redo-Log-Daten, die AWS-Region in einer globalen Neptune-Datenbank von der Primär- AWS-Region zur Sekundärdatenbank übertragen wurden.
- **`GlobalDbReplicatedWriteIO`**— Die Anzahl der I/O Schreibvorgänge, die von der Primärdatenbank in der globalen Datenbank auf das Cluster-Volume AWS-Region in einer Sekundärdatenbank repliziert wurden. AWS-Region

  Die Abrechnungsberechnungen für jeden DB-Cluster in einer globalen Neptune-Datenbank verwenden die `VolumeWriteIOPS`-Metrik, um die im Cluster ausgeführten Schreibvorgänge zu berücksichtigen. Verwenden Sie für den primären DB-Cluster bei den Abrechnungsberechnungen `NeptuneGlobalDbReplicatedWriteIO` zur Berücksichtigung der regionsübergreifenden Replikation zu sekundären DB-Clustern.
- **`GlobalDbProgressLag`** – Die Anzahl von Millisekunden, die ein sekundärer Cluster sowohl bei Benutzertransaktionen als auch bei Systemtransaktionen hinter dem primären Cluster zurückliegt.

Metrik	Dimension	Veröffentlicht in	Einheiten
`GlobalDbDataTransferBytes`	SourceRegion, DBCluster Bezeichner	Sekundär	Bytes
`GlobalDbReplicatedWriteIO`	SourceRegion, DBCluster Bezeichner	Sekundär	Anzahl
`GlobalDbProgressLag`	DBClusterBezeichner, Secondary Region : im primären	Primär, Sekundär	Millisekunden

Metrik	Dimension	Veröffentlicht in	Einheiten
	DBCluster Bezeichner, SourceRegion : im sekundären Bezeichner		

Übersicht über Amazon-Neptune-Features

> **Note**
>
> Dieser Abschnitt behandelt nicht die Verwendung der Abfragesprachen, mit denen Sie auf die Daten in einem Neptune-Diagramm zugreifen können.
> Informationen zum Herstellen von Verbindungen mit einem ausgeführten Neptune-DB-Cluster mit Gremlin finden Sie unter Zugriff auf ein Neptune-Diagramm mit Gremlin.
> Informationen zum Herstellen von Verbindungen mit einem ausgeführten Neptune-DB-Cluster mit openCypher finden Sie unter Zugriff auf das Neptun-Diagramm mit openCypher.
> Informationen zum Herstellen von Verbindungen mit einem ausgeführten Neptune-DB-Cluster mit SPARQL finden Sie unter Zugriff auf das Neptune-Diagramm mit SPARQL.

Dieser Abschnitt bietet eine Übersicht über spezifische Neptune-Features, darunter:

- Einhaltung der Standards für Abfragesprachen durch Neptune.
- Diagrammdatenmodell von Neptune.
- Beschreibung der Neptune-Transaktionssemantik.
- Einführung in Neptune-Cluster und -Instances.
- Speicher, Zuverlässigkeit und Verfügbarkeit von Neptune.
- Beschreibung der Neptune-Endpunkte.
- Verwenden des Neptune-Labormodus zur Aktivierung experimenteller Funktionen.
- Beschreibung der DFE-Engine von Neptune.
- JDBC-Konnektivität von Neptune.
- Liste der Neptune-Engine-Releases und Aktualisierung Ihrer Engine.

Hinweise zur Einhaltung von Amazon-Neptune-Standards

Amazon Neptune hält bei der Implementierung der Abfragesprachen Gremlin und SPARQL die geltenden Standards meistens ein.

Dieser Abschnitt beschreibt die Standards sowie die Bereiche, in denen Neptune-Standards diese erweitern oder von diesen abweichen.

Themen

- Einhaltung der Gremlin-Standards in Amazon Neptune
- Einhaltung von SPARQL-Standards in Amazon Neptune
- Einhaltung der OpenCypher-Spezifikationen in Amazon Neptune

Einhaltung der Gremlin-Standards in Amazon Neptune

Die folgenden Abschnitte bieten einen Überblick über die Neptune-Implementierung von Gremlin und wie sie sich von der Apache-Implementierung unterscheidet. TinkerPop

Neptune implementiert einige Gremlin-Schritte nativ in seiner Engine und verwendet die Apache TinkerPop Gremlin-Implementierung, um andere zu verarbeiten (siehe). Native Unterstützung für Gremlin-Schritte in Amazon Neptune

> **Note**
>
> Einige Beispiele für diese Implementierungsunterschiede zwischen der Gremlin-Konsole und Amazon Neptune finden Sie im Abschnitt the section called “Verwenden von Gremlin” im Schnellstart.

Themen

- Geltende Standards für Gremlin
- Variablen und Parameter in Skripts
- TinkerPop Aufzählungen
- Java-Code
- Eigenschaften von Elementen
- Skriptausführung

- Sitzungen
- Transaktionen
- Eckpunkt- und Kanten-IDs
- User-supplied IDs
- Eckpunkt-Eigenschaft-IDs
- Kardinalität von Eckpunkteigenschaften
- Aktualisieren einer Eckpunkteigenschaft
- Labels
- Escape-Zeichen
- Groovy-Einschränkungen
- Serialisierung
- Lambda-Schritte
- Nicht unterstützte Gremlin-Methoden
- Nicht unterstützte Gremlin-Schritte
- Features von Gremlin-Diagrammen in Neptune

Geltende Standards für Gremlin

- Die Gremlin-Sprache wird eher durch die TinkerPop Apache-Dokumentation und die TinkerPop Apache-Implementierung von Gremlin als durch eine formale Spezifikation definiert.
- Für numerische Formate folgt Gremlin dem IEEE 754-Standard (IEEE 754-2019 - IEEE Standard for Arithmetic). Floating-Point Weitere Informationen finden Sie auch auf der Wikipedia-Seite IEEE 754).

Variablen und Parameter in Skripts

Wenn es sich um vorgebundene Variablen handelt, befindet sich das Traversal-Objekt Pre-bound in Neptune, und das `graph` Objekt `g` wird nicht unterstützt.

Obwohl Neptune weder Gremlin-Variablen noch Parametrisierung in Skripts unterstützt, finden Sie im Internet häufig Beispielskripts für Gremlin Server, die Variablendeklarationen enthalten, zum Beispiel:

```
String query = "x = 1; g.V(x)";
```

```
List<Result> results = client.submit(query).all().get();
```

Es gibt auch viele Beispiele, die beim Absenden von Abfragen Parametrisierung (oder Bindungen) verwenden, zum Beispiel:

```
Map<String,Object> params = new HashMap<>();
params.put("x",1);
String query = "g.V(x)";
List<Result> results = client.submit(query).all().get();
```

Die Parameterbeispiele sind in der Regel mit Warnungen vor einer Beeinträchtigung der Leistung verbunden, wenn keine Parametrisierung erfolgt, wenn dies möglich ist. Es gibt eine Menge solcher Beispiele, denen Sie vielleicht begegnen werden TinkerPop , und sie klingen alle ziemlich überzeugend, was die Notwendigkeit der Parametrisierung angeht.

Sowohl die Funktion zur Deklaration von Variablen als auch die Parametrisierungsfunktion (zusammen mit den Warnungen) gelten jedoch nur für den Gremlin-Server, TinkerPop wenn dieser den verwendet. `GremlinGroovyScriptEngine` Sie gelten nicht, wenn Gremlin Server die `gremlin-language`-ANTLR-Grammatik von Gremlin verwendet, um Abfragen zu analysieren. Die ANTLR-Grammatik unterstützt weder Variablendeklarationen noch Parametrisierung. Wenn Sie ANTLR verwenden, müssen Sie sich daher keine Sorgen über eine fehlende Parametrisierung machen. Da es sich bei der ANTLR-Grammatik um eine neuere Komponente handelt TinkerPop, spiegeln ältere Inhalte, denen Sie im Internet möglicherweise begegnen, diesen Unterschied im Allgemeinen nicht wider.

Neptune verwendet in der Abfrageverarbeitungs-Engine die ANTLR-Grammatik und nicht `GremlinGroovyScriptEngine`. Daher werden Variablen, Parametrisierung oder die Eigenschaft `bindings` nicht unterstützt. Daher sind die Probleme im Zusammenhang mit einer fehlenden Parametrisierung nicht auf Neptune anwendbar. Mit Neptune können Abfragen einfach so gesendet werden, wie sie sind, ohne sie zu parametrisieren. Daher kann das vorherige Beispiel wie folgt vereinfacht werden, ohne dass die Leistung beeinträchtigt wird:

```
String query = "g.V(1)";
List<Result> results = client.submit(query).all().get();
```

TinkerPop Aufzählungen

Neptune unterstützt keine vollständig qualifizierten Klassennamen für Aufzählungswerte. In Ihrer Groovy-Anforderung müssen Sie beispielsweise `single` anstelle von

`org.apache.tinkerpop.gremlin.structure.VertexProperty.Cardinality.single` verwenden.

Der Aufzählungstyp wird durch den Parametertyp bestimmt.

Die folgende Tabelle zeigt die zulässigen Aufzählungswerte und den zugehörigen TinkerPop vollqualifizierten Namen.

Zulässige Werte	Klasse
`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

Java-Code

Neptune unterstützt keine Aufrufe von Methoden, die durch willkürliche Java- oder Java-Bibliotheks-Aufrufe definiert werden und nicht durch unterstützte Gremlin-APIs. Beispiel: `java.lang.*`, `Date()` und `g.V().tryNext().orElseGet()` sind nicht zulässig.

Eigenschaften von Elementen

Neptune unterstützt das in TinkerPop 3.7.0 eingeführte `materializeProperties` Flag zur Rückgabe von Eigenschaften von Elementen nicht. Infolgedessen gibt Neptune immer noch nur Scheitelpunkte oder Kanten als Referenzen mit nur ihrem UND zurück. `id` `label`

Skriptausführung

Alle Abfragen müssen mit `g`, dem Traversal-Objekt, beginnen.

Bei der Übermittlung von Zeichenfolgenabfragen können mehrere Traversierungen ausgegeben werden, durch Semikolon (`;`) oder ein Zeichen für eine neue Zeile (`\n`) getrennt. Alle Anweisungen außer der letzten Anweisung müssen mit einem `.iterate()`-Schritt beendet werden, um ausgeführt zu werden. Es werden nur die Daten der letzten Traversierung zurückgegeben. Beachten Sie, dass dies nicht für ByteCode GLV-Abfragen gilt.

Sitzungen

Sitzungen in Neptune sind auf 10 Minuten Dauer beschränkt. Weitere Informationen finden Sie unter Skriptbasierte Gremlin-Sitzungen und in der TinkerPopSitzungsreferenz.

Transaktionen

Neptune öffnet zu Beginn jeder Gremlin-Traversierung eine neue Transaktion und schließt sie nach dem erfolgreichen Abschluss der Traversierung. Wenn ein Fehler auftritt, wird ein Rollback für die Transaktion durchgeführt.

In einer einzelnen Transaktion sind mehrere Anweisungen, die durch Semikolon (`;`) oder Zeilenumbruchzeichen (\n) getrennt sind, enthalten. Jede Anweisung, die nicht die letzte Anweisung ist, muss mit einem auszuführenden `next()`-Schritt enden. Es werden nur die Daten der letzten Traversierung zurückgegeben.

Eine manuelle Transaktionslogik mit `tx.commit()` und `tx.rollback()` wird nicht unterstützt.

> **Important**
>
> Dies gilt nur für Methoden, bei denen Sie die Gremlin-Abfrage als Textzeichenfolge senden (siehe Gremline-Transaktionen).

Eckpunkt- und Kanten-IDs

Neptune-Gremlin-Eckpunkt– und -Kanten-IDs müssen den Typ `String` haben. Diese ID-Zeichenfolgen unterstützen Unicode-Zeichen und dürfen eine Größe von 55 MB nicht überschreiten.

User-supplied IDs werden unterstützt, sind aber bei normaler Verwendung optional. Wenn Sie beim Hinzufügen eines Eckpunkts oder einer Kante keine ID angeben, generiert Neptune eine UUID und konvertiert sie in eine Zeichenfolge mit dem folgenden Format: `"48af8178-50ce-971a-fc41-8c9a954cea62"` Diese UUIDs entsprechen nicht dem RFC-Standard. Wenn Sie Standard-UUIDs benötigen, sollten Sie diese extern generieren und angeben, wenn Sie Eckpunkte oder Kanten hinzufügen.

> **Note**
>
> Der Neptune-Befehl `Load` erfordert, dass Sie im Feld ~id IDs im Neptune-CSV-Format angeben.

User-supplied IDs

User-supplied Ausweise sind in Neptune Gremlin mit den folgenden Bestimmungen erlaubt.

- Bereitgestellte IDs sind optional.
- Es werden nur Eckpunkte und Kanten unterstützt.
- Es wird nur der `String`-Typ unterstützt.

Zum Erstellen eines neuen Vertex mit benutzerdefinierter ID verwenden Sie den `property`-Schritt mit dem `id`-Schlüsselwort `g.addV().property(id, 'customid')`.

> **Note**
>
> Schließen Sie das `id`-Schlüsselwort nicht in Anführungszeichen ein. Dies bezieht sich auf `T.id`.

Alle Eckpunkt-IDs und alle Kanten-IDs müssen eindeutig sein. Neptune lässt jedoch zu, dass ein Eckpunkt und eine Kante dieselbe ID haben.

Wenn Sie versuchen, einen neuen Vertex mit `g.addV()` zu erstellen und es ist bereits ein Vertex mit dieser ID vorhanden, schlägt die Operation fehl. Dabei gilt die folgende Ausnahme: Wenn Sie eine neue Bezeichnung für den Vertex angeben, ist die Operation erfolgreich, fügt die neue Bezeichnung und alle zusätzlichen angegebenen Eigenschaften aber zum vorhandenen Vertex hinzu. Es wird nichts überschrieben. Es wird kein neuer Vertex erstellt. Die Vertex-ID ändert sich nicht und bleibt eindeutig.

Die folgende Gremlin-Konsolenbefehl können beispielsweise erfolgreich ausgeführt werden:

```
gremlin> g.addV('label1').property(id, 'customid')
gremlin> g.addV('label2').property(id, 'customid')
gremlin> g.V('customid').label()
==>label1::label2
```

Eckpunkt-Eigenschaft-IDs

Eigenschaften-IDs von Knoten werden automatisch generiert und können bei Abfrage als positive oder negative Zahlen erscheinen.

Kardinalität von Eckpunkteigenschaften

Neptune unterstützt Set-Kardinalität und Single-Kardinalität. Wenn die Kardinalität nicht angegeben ist, wird die Set-Kardinalität ausgewählt. Das bedeutet, dass, wenn ein Eigenschaftswert festgelegt wird, ein neuer Wert zur Eigenschaft hinzugefügt wird, sofern dieser nicht bereits im Wertesatz enthalten ist. Dies ist der Gremlin-Aufzählungswert von Set.

`List` wird nicht unterstützt. Weitere Informationen zur Eigenschaftskardinalität finden Sie im Thema Vertex im Gremlin. JavaDoc

Aktualisieren einer Eckpunkteigenschaft

Zum Aktualisieren eines Eigenschaftswerts ohne Hinzufügung eines zusätzlichen Werts zum Wertesatz geben Sie die `single`-Kardinalität im `property`-Schritt an.

```
g.V('exampleid01').property(single, 'age', 25)
```

Dadurch werden alle vorhandenen Werte für die Eigenschaft entfernt.

Labels

Neptune unterstützt mehrere Bezeichnungen für einen Eckpunkt. Sie können mehrere Bezeichnungen angeben, indem Sie diese durch `::` trennen. Beispielsweise fügt `g.addV("Label1::Label2::Label3")` einen Knoten mit drei verschiedenen Bezeichnungen hinzu. Der `hasLabel`-Schritt entspricht dem Knoten mit allen drei Bezeichnungen: `hasLabel("Label1")`, `hasLabel("Label2")` und `hasLabel("Label3")`.

> **Important**
>
> Das `::`-Trennzeichen ist dieser Verwendung vorbehalten. Sie können im `hasLabel`-Schritt nicht mehrere Bezeichnungen angeben. Beispiel: Für `hasLabel("Label1::Label2")` gibt es keine Übereinstimmung.

Escape-Zeichen

Neptune löst alle Escape-Zeichen wie im Abschnitt Escaping Special Characters der Apache-Groovy-Dokumentation beschrieben auf.

Groovy-Einschränkungen

Neptune unterstützt keine Groovy-Befehle, die nicht mit g beginnen. Dies umfasst mathematische Zeichen (z. B.: `1+1`), Systemaufrufe (z. B: `System.nanoTime()`) und Variablendefinitionen (z. B: `1+1`).

> **Important**
>
> Neptune unterstützt keine vollständig qualifizierten Klassennamen. In Ihrer Groovy-Anforderung müssen Sie beispielsweise `single` anstelle von

`org.apache.tinkerpop.gremlin.structure.VertexProperty.Cardinality.single` verwenden.

Serialisierung

Neptune unterstützt die folgenden Serialisierungen basierend auf dem angeforderten MIME-Typ.

Neptune stellt alle Serialisierer zur Verfügung, die dies tun, und TinkerPop unterstützt die verschiedenen Versionen und Konfigurationen von GraphSON und. GraphBinary Obwohl es viele Optionen gibt, ist die Anleitung zur Verwendung einfach:

- Wenn Sie TinkerPop Apache-Treiber verwenden, bevorzugen Sie die Standardeinstellung für den Treiber, ohne einen explizit anzugeben. Sofern Sie keinen ganz bestimmten Grund dafür haben, müssen Sie den Serializer wahrscheinlich nicht bei der Treiberinitialisierung angeben. Im Allgemeinen ist die von den Treibern verwendete Standardeinstellung. `application/vnd.graphbinary-v1.0`
- Wenn Sie über HTTP eine Verbindung zu Neptune herstellen, sollten Sie der Verwendung von Priorität einräumen, `application/vnd.gremlin-v3.0+json;types=false` da die eingebetteten Typen in der alternativen Version von GraphSon 3 die Arbeit erschweren.
- Das `application/vnd.graphbinary-v1.0-stringd` ist im Allgemeinen nur nützlich, wenn es in Verbindung mit der Gremlin Console verwendet wird, da es alle Ergebnisse zur einfachen Anzeige in eine Zeichenkettendarstellung konvertiert.
- Die übrigen Formate bleiben aus veralteten Gründen weiterhin vorhanden und sollten normalerweise nicht ohne eindeutigen Grund zusammen mit Treibern verwendet werden.

MIME-Typ	Serialisierung	Konfiguration
`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` (funktioniert nur mit 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**
>
> Die hier gezeigte Serializer-Tabelle bezieht sich auf die Benennung ab 3.7.0. TinkerPop Wenn Sie mehr über diese Änderung erfahren möchten, lesen Sie bitte die TinkerPop Upgrade-Dokumentation. Die Unterstützung der Gryo-Serialisierung war in 3.4.3 veraltet und wurde in 3.6.0 offiziell entfernt. Wenn Sie Gryo explizit verwenden oder eine Treiberversion verwenden, die Gryo standardmäßig verwendet, sollten Sie zu Ihrem Treiber wechseln oder ihn aktualisieren. GraphBinary

Lambda-Schritte

Lambda-Schritte werden von Neptune nicht unterstützt.

Nicht unterstützte Gremlin-Methoden

Die folgenden Gremlin-Methoden werden von Neptune nicht unterstützt:

- `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`

Die folgende Traversierung ist beispielsweise nicht zulässig:
`g.V().addE('something').from(__.V().next()).to(__.V().next())`.

> **Important**
>
> Dies gilt nur für Methoden, bei denen die Gremlin-Abfrage als Textzeichenfolge gesendet wird.

Nicht unterstützte Gremlin-Schritte

Die folgenden Gremlin-Schritte werden von Neptune nicht unterstützt:

- Der Gremlin-Schritt io( ) wird von Neptune nur teilweise unterstützt. Er kann in einem Lesekontext verwendet werden, wie in `g.io(`*`(url)`*`).read()`, jedoch nicht zum Schreiben.

Features von Gremlin-Diagrammen in Neptune

Die Neptune-Implementierung von Gremlin legt das `graph`-Objekt nicht offen. Die folgenden Tabellen listen Gremlin-Features auf und geben an, ob Neptune sie unterstützt oder nicht.

Neptun-Unterstützung für Grafikfunktionen

Die Neptune-Diagramm-Features (wenn vorhanden) sind mit den Features identisch, die vom Befehl `graph.features()` zurückgegeben würden.

Diagramm-Feature	Aktiviert?
`Transactions`	true
`ThreadedTransactions`	false
`Computer`	false
`Persistence`	true
`ConcurrentAccess`	true

Neptune-Unterstützung für Variablen-Features

Variablen-Feature	Aktiviert?
`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

Neptune-Unterstützung für Eckpunkt-Features

Eckpunkt-Feature	Aktiviert?
`MetaProperties`	false
`DuplicateMultiProperties`	false
`AddVertices`	true
`RemoveVertices`	true
`MultiProperties`	true
`UserSuppliedIds`	true
`AddProperty`	true
`RemoveProperty`	true
`NumericIds`	false
`StringIds`	true
`UuidIds`	false
`CustomIds`	false
`AnyIds`	false

Neptune-Unterstützung für Eckpunkt-Eigenschafts-Features

Eckpunkt-Eigenschafts-Feature	Aktiviert?
`UserSuppliedIds`	false
`AddProperty`	true
`RemoveProperty`	true
`NumericIds`	true
`StringIds`	true
`UuidIds`	false
`CustomIds`	false
`AnyIds`	false
`Properties`	true
`SerializableValues`	false
UniformListValues	false
`BooleanArrayValues`	false
`DoubleArrayValues`	false
`IntegerArrayValues`	false
`StringArrayValues`	false
`BooleanValues`	true
`ByteValues`	true
`DoubleValues`	true
`FloatValues`	true
`IntegerValues`	true

`LongValues`	true
`MapValues`	false
`MixedListValues`	false
`StringValues`	true
`ByteArrayValues`	false
`FloatArrayValues`	false
`LongArrayValues`	false

Neptune-Unterstützung für Kanten-Features

Kanten-Feature	Aktiviert?
`AddEdges`	true
`RemoveEdges`	true
`UserSuppliedIds`	true
`AddProperty`	true
`RemoveProperty`	true
`NumericIds`	false
`StringIds`	true
`UuidIds`	false
`CustomIds`	false
`AnyIds`	false

Neptune-Unterstützung für Kanten-Eigenschafts-Features

Kanten-Eigenschafts-Feature	Aktiviert?
`Properties`	true
`SerializableValues`	false
`UniformListValues`	false
`BooleanArrayValues`	false
`DoubleArrayValues`	false
`IntegerArrayValues`	false
`StringArrayValues`	false
`BooleanValues`	true
`ByteValues`	true
`DoubleValues`	true
`FloatValues`	true
`IntegerValues`	true
`LongValues`	true
`MapValues`	false
`MixedListValues`	false
`StringValues`	true
`ByteArrayValues`	false
`FloatArrayValues`	false
`LongArrayValues`	false

Einhaltung von SPARQL-Standards in Amazon Neptune

Nach der Auflistung der geltenden SPARQL-Standards enthalten die folgenden Abschnitte spezifische Informationen dazu, wie die SPARQL-Implementierung von Neptune diese Standards erweitert oder von ihnen abweicht.

Themen

- Geltende Standards für SPARQL
- Standardpräfixe für Namespaces in Neptune SPARQL
- SPARQL-Standard-Graph und benannte Graphen
- Von Neptune unterstützte SPARQL XPath-Konstruktorfunktionen
- Standard-Basis-IRI für Abfragen und Updates
- xsd:dateTime Werte in Neptune
- Behandlung spezieller Gleitkommawerte in Neptune
- Neptun-Wertebegrenzung Arbitrary-Length
- Neptune erweitert den Equals-Vergleich in SPARQL
- Umgang mit Out-of-Range Literalen in Neptune SPARQL

Amazon Neptune erfüllt bei der Implementierung der SPARQL-Diagrammabfragensprache die folgenden Standards.

Geltende Standards für SPARQL

- SPARQL wird vom W3C in der Empfehlung SPARQL 1.1 Query Language vom 21. März 2013 definiert.
- Das SPARQL-Update-Protokoll und die Abfragesprache werden durch die W3C SPARQL 1.1-Update-Spezifikation definiert.
- Für numerische Formate folgt SPARQL der W3C XML Schema Definition Language (XSD) 1.1 Teil 2: Datentypen-Spezifikation, die mit der IEEE 754-Spezifikation (IEEE 754-2019 — IEEE Standard for Arithmetic) konsistent ist. Floating-Point Weitere Informationen finden Sie auch auf der Wikipedia-Seite (IEEE 754). Funktionen, die nach Version `IEEE 754-1985` eingeführt wurden, sind in der Spezifikation jedoch nicht enthalten.

Standardpräfixe für Namespaces in Neptune SPARQL

Neptune definiert standardmäßig die folgenden Präfixe für die Verwendung in SPARQL-Abfragen. Weitere Informationen finden Sie unter Präfixnamen in der SPARQL-Spezifikation.

- `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#`

SPARQL-Standard-Graph und benannte Graphen

Amazon Neptune ordnet jedem Triple einen benannten Graphen zu. Der Standard-Graph ist als die Vereinigung aller benannten Graphen definiert.

Standard-Graph für Abfragen

Wenn Sie eine SPARQL-Anfrage absenden, ohne ausdrücklich einen Graphen über das `GRAPH`-Schlüsselwort oder Konstrukte wie `FROM NAMED` anzugeben, berücksichtigt Neptune stets alle Triples in Ihrer DB-Instance. Die folgende Abfrage gibt beispielsweise alle Triples von einem Neptune-SPARQL-Endpunkt zurück:

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

Triples, die in mehr als einem Graphen vorhanden sind, werden nur einmal zurückgegeben.

Weitere Informationen über die Standard-Graph-Spezifikation finden Sie im RDF-Dataset-Abschnitt der Query Language SPARQL 1.1-Spezifikation.

Angeben des benannten Graphen für Ladevorgänge, Einfügungen oder Updates

Wenn Sie beim Laden, Einfügen oder Aktualisieren von Triples keinen benannten Graphen angeben, verwendet Neptune den benannten Fallback-Graphen, der durch die URI `http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph` definiert ist.

Wenn Sie eine Neptune-`Load`-Anforderung in einem Triple-basierten Format ausgeben, können Sie mit dem Parameter `parserConfiguration: namedGraphUri` angeben, dass der genannte Graph für alle Triples verwendet werden soll. Weitere Informationen zur `Load`-Befehlssyntax finden Sie unter the section called “Loader-Befehl”.

> **Important**
>
> Wenn Sie diesen Parameter nicht benutzen und keinen benannten Graphen angeben, wird die Fallback-URI verwendet: `http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph`.

Dieser benannte Fallback-Graph wird auch verwendet, wenn Sie Triples über `SPARQL UPDATE` laden, ohne ausdrücklich einen benannten Ziel-Graphen anzugeben.

Sie können das Quads-basierte Format verwenden, N-Quads um für jedes Tripel in der Datenbank ein benanntes Diagramm anzugeben.

> **Note**
>
> Mithilfe von N-Quads können Sie das benannte Diagramm leer lassen. In diesem Fall wird `http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph` verwendet. Sie können das standardmäßige benannte Diagramm überschreiben, um die `namedGraphUri` Parser-Konfigurationsoption zu N-Quads verwenden.

Von Neptune unterstützte SPARQL XPath-Konstruktorfunktionen

Der SPARQL-Standard ermöglicht SPARQL-Engines die Unterstützung eines erweiterbaren Satzes von XPath-Konstruktorfunktionen. Neptune unterstützt zurzeit die folgenden Konstruktorfunktionen, wobei das `xsd`-Präfix als `http://www.w3.org/2001/XMLSchema#` definiert ist:

- `xsd:boolean`
- `xsd:integer`
- `xsd:double`
- `xsd:float`
- `xsd:decimal`
- `xsd:long`
- `xsd:unsignedLong`

Standard-Basis-IRI für Abfragen und Updates

Da ein Neptune-Cluster mehrere Endpunkte besitzt, kann die Verwendung der Anfrage-URL einer Abfrage oder eines Updates als Basis-IRI zu unerwarteten Ergebnissen bei der Auflösung relativer IRIs führen.

Ab Engine-Version 1.2.1.0 verwendet Neptune `http://aws.amazon.com/neptune/default/` als Basis-IRI, wenn kein expliziter Basis-IRI Teil der Anforderung ist.

In der folgenden Anforderung ist der Basis-IRI Teil der Anforderung:

```
BASE <http://example.org/default/>
INSERT DATA { <node1> <id> "n1" }

BASE <http://example.org/default/>
SELECT * { <node1> ?p ?o }
```

Das Ergebnis wäre:

```
?p                                                  ?o
http://example.org/default/id                       n1
```

In dieser Anforderung ist jedoch kein Basis-IRI enthalten:

```
INSERT DATA { <node1> <id> "n1" }

SELECT * { <node1> ?p ?o }
```

In diesem Fall wäre das Ergebnis:

```
?p                                                  ?o
http://aws.amazon.com/neptune/default/id            n1
```

xsd:dateTime Werte in Neptune

Aus Leistungsgründen speichert Neptune date/time Werte immer als koordinierte Weltzeit (UTC). Dies macht direkte Vergleiche sehr effizient.

Es bedeutet auch, dass Neptune den Wert in UTC übersetzt und die Zeitzoneninformationen verwirft, wenn Sie einen `dateTime`-Wert eingeben, der eine bestimmte Zeitzone angibt. Wenn Sie den

`dateTime`-Wert später abrufen, wird er in UTC ausgedrückt, nicht in der Zeit der ursprünglichen Zeitzone, und Sie können nicht mehr erkennen, was die ursprüngliche Zeitzone war.

Behandlung spezieller Gleitkommawerte in Neptune

Neptune behandelt spezielle Gleitkommawerte in SPARQL wie folgt.

SPARQL-NaN-Behandlung in Neptune

SPARQL kann in Neptune einen Wert von `NaN` in einer Abfrage akzeptieren. Es gibt keine Unterscheidung zwischen signalisierenden und stillen `NaN`-Werten. Neptune behandelt alle `NaN`-Werte als stille Werte.

Semantisch ist kein Vergleich mit einem `NaN`-Wert möglich, da nichts größer als, kleiner als oder gleich einem `NaN`-Wert ist. Das bedeutet, dass ein Wert von `NaN` auf der einen Seite eines Vergleichs theoretisch mit nichts auf der anderen Seite übereinstimmen kann.

Die XSD-Spezifikation behandelt jedoch zwei `xsd:double`- oder `xsd:float`-`NaN`Werte als gleich. Neptune folgt dem für den Filter `IN`, für den Gleichheiteroperator in Filterausdrücken und für die exakte Übereinstimmungssemantik (mit einem `NaN`-Wert in der Objektposition eines Triple-Musters).

Behandlung unendlicher SPARQL-Werte in Neptune

SPARQL kann in Neptune einen Wert von `-INF` oder `INF` in einer Abfrage akzeptieren. `INF` ist in Vergleichen größer als jeder andere numerische Wert; `-INF` ist in Vergleichen kleiner als jeder andere numerische Wert.

Zwei INF-Werte mit übereinstimmenden Vorzeichen werden unabhängig vom Typ als gleich behandelt (z. B. gilt ein Float`--INF` als gleich einem Double`--INF`).

Selbstverständlich ist mit `NaN` kein Vergleich möglich, denn nichts ist größer, kleiner oder gleich `NaN`.

Behandlung einer negativen SPARQL-Null in Neptune

Neptune normalisiert einen negativen Nullwert zu einer Null ohne Vorzeichen. Sie können negative Nullwerte in einer Abfrage verwenden. Sie werden jedoch nicht als solche in der Datenbank aufgezeichnet und bei Vergleichen entsprechen sie Nullen ohne Vorzeichen.

Neptun-Wertebegrenzung Arbitrary-Length

Neptune begrenzt die Speichergröße von XSD-Ganzzahl-, Gleitkomma- und Dezimalwerten in SPARQL auf 64 Bit. Die Verwendung von größeren Werten führt zum Fehler `InvalidNumericDataException`.

Neptune erweitert den Equals-Vergleich in SPARQL

Der SPARQL-Standard definiert eine ternäre Logik für Wertausdrücke, wobei ein Wertausdruck als `true`, `false` oder `error` bewertet werden kann. Die Standardsemantik für Begriffsgleichheit (wie in der SPARQL 1.1-Spezifikation definiert), die für die Vergleiche = und `!=` in `FILTER`-Bedingungen gilt, erzeugt einen `error` beim Vergleich von Datentypen, die in der Operatoren-Tabelle nicht explizit vergleichbar sind.

Dieses Verhalten kann zu Ergebnissen, wie im folgenden Beispiel, führen, die nicht intuitiv sind.

Daten:

```
<http://example.com/Server/1> <http://example.com/ip> "127.0.0.1"^^<http://example.com/
datatype/IPAddress>
```

Abfrage 1:

```
SELECT * WHERE {
    <http://example.com/Server/1> <http://example.com/ip> ?o .
    FILTER(?o = "127.0.0.2"^^<http://example.com/datatype/IPAddress>)
}
```

Abfrage 2:

```
SELECT * WHERE {
    <http://example.com/Server/1> <http://example.com/ip> ?o .
    FILTER(?o != "127.0.0.2"^^<http://example.com/datatype/IPAddress>)
}
```

Mit der SPARQL-Standardsemantik, die Neptune vor Version 1.0.2.1 verwendet hat, würden beide Abfragen ein leeres Ergebnis zurückgeben. Der Grund hierfür ist, dass `?o = "127.0.0.2"^^<http://example.com/IPAddress>` bei der Auswertung für `?o := "127.0.0.1"^^<http://example.com/IPAddress>` einen `error` statt `false` erzeugt, weil keine expliziten Vergleichsregeln für den benutzerdefinierten Datentyp `<http://example.com/IPAddress>` angegeben sind. Infolgedessen erzeugt die negierte Version in der zweiten Abfrage auch einen `error`. In beiden Abfragen bewirkt der `error`, dass die Kandidatenlösung herausgefiltert wird.

Ab Version 1.0.2.1 erweitert Neptune den SPARQL-Ungleichheitsoperator entsprechend der Spezifikation. Weitere Informationen finden Sie im Abschnitt von SPARQL 1.1 zur Operator-

Erweiterbarkeit, in dem Engines zusätzliche Regeln für den Vergleich zwischen benutzerdefinierten und nicht vergleichbaren integrierten Datentypen definieren können.

Bei Verwendung dieser Option bewertet Neptune jetzt den Vergleich zwischen zwei in der Operator-Zuweisungstabelle nicht explizit definierten Datentypen als `true`, wenn die Literalwerte und Datentypen syntaktisch gleich sind. Andernfalls wird der Vergleich als „false“ bewertet. Ein `error` wird in keinem Fall erstellt.

Mit dieser neuen Semantik würde die zweite Abfrage `"127.0.0.1"^^<http://example.com/IPAddress>` anstelle eines leeren Ergebnisses zurückgeben.

Umgang mit Out-of-Range Literalen in Neptune SPARQL

Mit der XSD-Semantik wird jeder numerische Typ mit seinem Wertebereich definiert, abgesehen von `integer` und `decimal`. Diese Definitionen beschränken jeden Typ auf einen Wertebereich. Beispielsweise liegt der Umfang eines `xsd:byte`-Bereichs zwischen -128 und +127. Jeder Wert außerhalb dieses Bereichs gilt als ungültig.

Wenn Sie versuchen, einen Literalwert außerhalb des Wertebereichs eines Typs zuzuweisen (wenn Sie beispielsweise versuchen, ein `xsd:byte` auf den Literalwert 999 festzulegen), akzeptiert Neptune den Wert außerhalb des Bereichs unverändert, ohne zu runden oder zu verkürzen. Allerdings bleibt der Wert nicht als numerischer Wert erhalten, da der angegebene Typ ihn nicht darstellen kann.

Das heißt, Neptune akzeptiert `"999"^^xsd:byte`, obwohl es sich um einen Wert außerhalb des definierten `xsd:byte`-Wertebereichs handelt. Nachdem der Wert jedoch in der Datenbank beibehalten wurde, kann er in der exakten Übereinstimmungssemantik nur in einer Objektposition eines dreifachen Musters verwendet werden. Es kann kein Bereichsfilter ausgeführt werden, da Literale außerhalb des Bereichs nicht als numerische Werte behandelt werden.

Die SPARQL 1.1-Spezifikation definiert Bereichsoperatoren im Format `numeric`-Operator-`numeric`, `string`-Operator-`string`, `literal`-Operator-`literal` usw. Neptune kann keine Bereichsvergleichsoperatoren wie `invalid-literal`-operator-`numeric-value` ausführen.

Einhaltung der OpenCypher-Spezifikationen in Amazon Neptune

Die Amazon Neptune-Version von openCypher unterstützt im Allgemeinen die Klauseln, Operatoren, Ausdrücke, Funktionen und Syntax, die in Cypher Query Language Reference Version 9 definiert sind, der aktuellen openCypher-Spezifikation. Im Folgenden werden die Einschränkungen und Unterschiede bei der Neptune-Unterstützung für openCypher beschrieben.

Amazon Neptune unterstützt auch mehrere Funktionen, die über den Rahmen der OpenCypher-Spezifikation hinausgehen. Weitere Einzelheiten finden Sie unter OpenCypher-Erweiterungen in Amazon Neptune.

> **Note**
>
> Die aktuelle Neo4j-Implementierung von Cypher enthält Funktionen, die nicht in der oben genannten openCypher-Spezifikation enthalten sind. Wenn Sie aktuellen Cypher-Code zu Neptune migrieren, finden Sie unter Neptune-Kompatibilität mit Neo4j und Umschreiben von Cypher-Abfragen zur Ausführung in openCypher auf Neptune weitere Informationen.

Unterstützung für openCypher-Klauseln in Neptune

Neptune unterstützt die folgenden Klauseln, wenn nicht anders angegeben:

- `MATCH` – Unterstützt, jedoch werden *`shortestPath()`* und *`allShortestPaths()`* zurzeit nicht unterstützt.
- `OPTIONAL MATCH`
- *`MANDATORY MATCH`* – Zurzeit nicht in Neptune unterstützt. Neptune unterstützt jedoch benutzerdefinierte ID-WerteMATCH in -Abfragen.
- `RETURN` – Unterstützt, jedoch nicht in Verbindung mit nichtstatischen Werten für `SKIP` oder `LIMIT`. Beispielsweise funktioniert Folgendes zurzeit nicht:

```
MATCH (n)
RETURN n LIMIT toInteger(rand())    // Does NOT work!
```

- `WITH` – Unterstützt, jedoch nicht in Verbindung mit nichtstatischen Werten für `SKIP` oder `LIMIT`. Beispielsweise funktioniert Folgendes zurzeit nicht:

```
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` – Sie können mit Neptune benutzerdefinierte ID-Werte in `CREATE`-Abfragen erstellen.
- `DELETE`
- `SET`
- `REMOVE`
- `MERGE` – Neptune unterstützt benutzerdefinierte ID-Werte in `MERGE`-Abfragen.
- *`CALL[YIELD...]`* – Zurzeit nicht in Neptune unterstützt.
- `UNION, UNION ALL` – Schreibgeschützte Abfragen werden unterstützt, Mutationsabfragen werden zurzeit jedoch nicht unterstützt.
- `USING`— `USING` wird ab Engine-Version 1.3.2.0 unterstützt. Weitere Informationen finden Sie unter Abfragehinweise.

Unterstützung für openCypher-Operatoren in Neptune

Neptune unterstützt die folgenden Operatoren, wenn nicht anders angegeben:

Allgemeine Operatoren

- `DISTINCT`
- **`.`** Operator für den Zugriff auf Eigenschaften einer verschachtelten Literalmap.

Mathematische Operatoren

- **`+`** Additionsoperator.
- **`-`** Subtraktionsoperator.
- **`*`** Multiplikationsoperator.
- **`/`** Teilungsoperator.
- **`%`** Modulo-Teilungsoperator.
- Der **`^`** Potenzierungsoperator. *`is NOT supported`*

Vergleichsoperatoren

- **`=`** Additionsoperator.

- **<>** Ungleichheitsoperator.
- Der **<** Kleiner-Als-Operator wird unterstützt, es sei denn, eines der Argumente ist Path, List oder Map.
- Der **>** Größer-Als-Operator wird unterstützt, es sei denn, eines der Argumente ist Path, List oder Map.
- Der **<=** Kleiner-Als-Oder-Gleich-Operator wird unterstützt, es sei denn, eines der Argumente ist Path, List oder Map.
- Der **>=** Größer-Als-Oder-Gleich-Operator wird unterstützt, es sei denn, eines der Argumente ist Path, List oder Map.
- `IS NULL`
- `IS NOT NULL`
- `STARTS WITH` wird unterstützt, wenn es sich bei den gesuchten Daten um eine Zeichenfolge handelt.
- `ENDS WITH` wird unterstützt, wenn es sich bei den gesuchten Daten um eine Zeichenfolge handelt.
- `CONTAINS` wird unterstützt, wenn es sich bei den gesuchten Daten um eine Zeichenfolge handelt.

Boolesche Operatoren

- `AND`
- `OR`
- `XOR`
- `NOT`

Zeichenfolgenoperatoren

- **+** Verkettungsoperator.

Listenoperatoren.

- **+** Verkettungsoperator.
- `IN` (überprüft auf Vorhandensein eines Elements in der Liste)

Unterstützung für openCypher-Ausdrücke in Neptune

Neptune unterstützt die folgenden Ausdrücke, wenn nicht anders angegeben:

- `CASE`
- Der **`[]`**-Ausdruck wird zurzeit in Neptune nicht für den Zugriff auf dynamisch berechnete Eigenschaftsschlüssel innerhalb eines Knotens, einer Beziehung oder einer Map unterstützt. Beispielsweise funktioniert Folgendes nicht:

```
MATCH (n)
WITH [5, n, {key: 'value'}] AS list
RETURN list[1].name
```

Unterstützung für openCypher-Funktionen in Neptune

Neptune unterstützt die folgenden Funktionen, wenn nicht anders angegeben:

Prädikatsfunktionen

- `exists()`

Skalarfunktionen

- `coalesce()`
- `endNode()`
- `epochmillis()`
- `head()`
- `id()`
- `last()`
- `length()`
- `randomUUID()`
- `properties()`
- `removeKeyFromMap`
- `size()` – Diese Methodenüberladung funktioniert zurzeit nur für Musterausdrücke, Listen und Zeichenfolgen.

- `startNode()`
- `timestamp()`
- `toBoolean()`
- `toFloat()`
- `toInteger()`
- `type()`

Aggregationsfunktionen

- `avg()`
- `collect()`
- `count()`
- `max()`
- `min()`
- `percentileDisc()`
- `stDev()`
- `percentileCont()`
- `stDevP()`
- `sum()`

Listenfunktionen

- `join()` (verkettet Zeichenfolgen in einer Liste zu einer einzigen Zeichenfolge)
- `keys()`
- `labels()`
- `nodes()`
- `range()`
- `relationships()`
- `reverse()`
- `tail()`

Mathematische Funktionen – numerisch

- `abs()`
- `ceil()`
- `floor()`
- `rand()`
- `round()`
- `sign()`

Mathematische Funktionen – logarithmisch

- `e()`
- `exp()`
- `log()`
- `log10()`
- `sqrt()`

Mathematische Funktionen – trigonometrisch

- `acos()`
- `asin()`
- `atan()`
- `atan2()`
- `cos()`
- `cot()`
- `degrees()`
- `pi()`
- `radians()`
- `sin()`
- `tan()`

Zeichenfolgenfunktionen

- `join()` (verkettet Zeichenfolgen in einer Liste zu einer einzigen Zeichenfolge)
- `left()`
- `lTrim()`
- `replace()`
- `reverse()`
- `right()`
- `rTrim()`
- `split()`
- `substring()`
- `toLower()`
- `toString()`
- `toUpper()`
- `trim()`

User-defined Funktionen

*User-defined functions*werden derzeit in Neptune nicht unterstützt.

Neptune-specific Einzelheiten zur OpenCypher-Implementierung

Die folgenden Abschnitte beschreiben, wie sich die Neptune-Implementierung von openCypher von der openCypher-Spezifikation unterscheidet oder über diese hinausgeht.

Variable-length Pfadbewertungen (VLP) in Neptune

Evaluierungen von Pfaden (VLP) variabler Länge entdecken Pfade zwischen Knoten im Diagramm. Die Pfadlänge in einer Abfrage kann uneingeschränkt sein. Um Zyklen zu verhindern, legt die openCypher-Spezifikation fest, dass jede Kante höchstens einmal pro Lösung traversiert werden darf.

Die Neptune-Implementierung weicht bei VLPs von der openCypher-Spezifikation ab, da sie für Eigenschaftsgleichheitsfilter nur konstante Werte unterstützt. Führen Sie die folgende Abfrage aus:

```
MATCH (x)-[:route*1..2 {dist:33, code:x.name}]->(y) return x,y
```

Da der Wert des `x.name`-Eigenschaftsgleichheitsfilters keine Konstante ist, führt diese Abfrage zu einer `UnsupportedOperationException` mit der folgenden Meldung: `Property predicate over variable-length relationships with non-constant expression is not supported in this release.`

Temporale Unterstützung in der Neptune OpenCypher-Implementierung (Neptune-Datenbank 1.3.1.0 und niedriger)

Neptune stellt zurzeit eine eingeschränkte Unterstützung für temporale Funktionen in openCypher bereit. Die Lösung unterstützt den Datentyp `DateTime` für temporale Typen.

Die Funktion `datetime()` kann verwendet werden, um die aktuellen Werte für UTC-Datum und -Uhrzeit wie folgt abzurufen:

```
RETURN  datetime() as res
```

Datums- und Uhrzeitwerte können aus Zeichenfolgen im Format "dateTtime" analysiert werden, wobei date und time in einem der folgenden unterstützten Formate ausgedrückt werden:

Unterstützte Datumsformate

- `yyyy-MM-dd`
- `yyyyMMdd`
- `yyyy-MM`
- `yyyy-DDD`
- `yyyyDDD`
- `yyyy`

Unterstützte Uhrzeitformate

- `HH:mm:ssZ`
- `HHmmssZ`
- `HH:mm:ssZ`
- `HH:mmZ`
- `HHmmZ`
- `HHZ`

- `HHmmss`
- `HH:mm:ss`
- `HH:mm`
- `HHmm`
- `HH`

Beispiel:

```
RETURN datetime('2022-01-01T00:01')      // or another example:
RETURN datetime('2022T0001')
```

Beachten Sie, dass alle date/time Werte in Neptune OpenCypher als UTC-Werte gespeichert und abgerufen werden.

Neptune openCypher verwendet eine `statement`-Uhr. Das bedeutet, dass über die Dauer einer Abfrage derselbe Zeitpunkt verwendet wird. Eine andere Abfrage innerhalb derselben Transaktion verwendet möglicherweise einen anderen Zeitpunkt.

Neptune unterstützt keine Verwendung von Funktionen innerhalb von Aufrufen von `datetime()`. Beispielsweise funktioniert Folgendes nicht:

```
CREATE (:n {date:datetime(tostring(2021))})  // ---> NOT ALLOWED!
```

Neptune unterstützt die Funktion `epochmillis()`, die einen `datetime`-Wert in `epochmillis` konvertiert. Beispiel:

```
MATCH (n) RETURN epochMillis(n.someDateTime)
1698972364782
```

Neptune unterstützt zurzeit keine anderen Funktionen und Operationen für `DateTime`-Objekte, z. B. Addition und Subtraktion.

Temporale Unterstützung in der Neptune OpenCypher-Implementierung (Neptune Analytics und Neptune Database 1.3.2.0 und höher)

Die folgende Datetime-Funktionalität für OpenCypher gilt für Neptune Analytics. Alternativ können Sie den Labmode-Parameter `DatetimeMillisecond=enabled` verwenden, um die folgenden Datetime-Funktionen auf Neptune Engine-Release-Version 1.3.2.0 und höher zu aktivieren. Weitere

Informationen zur Verwendung dieser Funktionalität im Labmode finden Sie unter. Erweiterte Datetime-Unterstützung

- Support für Millisekunden. Das Datetime-Literal wird immer mit Millisekunden zurückgegeben, auch wenn Millisekunden 0 ist. (Bisheriges Verhalten bestand darin, Millisekunden zu kürzen.)

```
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 für den Aufruf der Funktion datetime () über gespeicherte Eigenschaften oder Zwischenergebnisse. Beispielsweise waren die folgenden Abfragen vor dieser Funktion nicht möglich.

  Datetime () über Eigenschaften:

```
// 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 () im Vergleich zu Zwischenergebnissen:

```
// Parse datetime from parameter
UNWIND $list as myDate
```

```
RETURN datetime(myDate) as d
```

- Es ist jetzt auch möglich, Datetime-Eigenschaften zu speichern, die in den oben genannten Fällen erstellt wurden.

  Speichern von Datetime aus der String-Eigenschaft einer Eigenschaft in eine andere:

```
// 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)
```

  Batch-Erstellung von Knoten aus einem Parameter mit einer Datetime-Eigenschaft:

```
// 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 für eine größere Teilmenge von ISO8601-Datetime-Formaten. Weitere Informationen finden Sie unter weiter unten in diesem Dokument.

Unterstützte Formate

Das Format eines Datetime-Werts ist [Date] T [Time] [Timezone], wobei T das Trennzeichen ist. Wenn keine explizite Zeitzone angegeben wird, wird UTC (Z) als Standard angenommen.

Zeitzone

Folgende Zeitzonenformate werden unterstützt:

- +/-HH:mm
- +/-hhmm
- +/-HH

Das Vorhandensein einer Zeitzone in einer Datetime-Zeichenfolge ist optional. Falls der Zeitzonen-Offset 0 ist, kann Z anstelle des obigen Zeitzonen-Postfixes verwendet werden, um die UTC-Zeit anzugeben. Der unterstützte Bereich einer Zeitzone liegt zwischen -14:00 und + 14:00 Uhr.

Date

Wenn keine Zeitzone vorhanden ist oder die Zeitzone UTC (Z) ist, lauten die unterstützten Datumsformate wie folgt:

> **Note**
>
> DDD bezieht sich auf ein Ordinaldatum, das für einen Tag des Jahres von 001 bis 365 steht (366 in Schaltjahren). 2024-002 steht beispielsweise für den 2. Januar 2024.

- `yyyy-MM-dd`
- `yyyyMMdd`
- `yyyy-MM`
- `yyyyMM`
- `yyyy-DDD`
- `yyyyDDD`
- `yyyy`

Wenn eine andere Zeitzone als Z ausgewählt wird, sind die unterstützten Datumsformate auf die folgenden beschränkt:

- `yyyy-MM-dd`
- `yyyy-DDD`
- `yyyyDDD`

Der unterstützte Datumsbereich reicht von 1400-01-01 bis 9999-12-31.

Zeit

Wenn keine Zeitzone vorhanden ist oder die Zeitzone UTC (Z) ist, werden folgende Zeitformate unterstützt:

- `HH:mm:ss.SSS`
- `HH:mm:ss`
- `HHmmss.SSS`
- `HHmmss`
- `HH:mm`
- `HHmm`
- `HH`

Wenn eine andere Zeitzone als Z ausgewählt wird, sind die unterstützten Zeitformate auf die folgenden beschränkt:

- `HH:mm:ss`
- `HH:mm:ss.SSS`

Unterschiede bei der Semantik der Neptune-openCypher-Sprache

Neptune stellt Knoten- und Beziehungs-IDs als Zeichenfolgen, nicht als Ganzzahlen dar. Die ID entspricht der ID, die über den Datenloader bereitgestellt wird. Wenn es einen Namespace für die Spalte gibt, entspricht sie dem Namespace und der ID. Daher gibt die Funktion `id` eine Zeichenfolge statt einer Ganzzahl zurück.

Der Datentyp `INTEGER` ist auf 64 Bit begrenzt. Wenn Sie mittels der Funktion `TOINTEGER` größere Fließkomma- oder Zeichenfolgenwerte in eine Ganzzahl konvertieren, werden negative Werte zu `LLONG_MIN` und positive Werte zu `LLONG_MAX` verkürzt.

Beispiel:

```
RETURN TOINTEGER(2^100)
>  9223372036854775807

RETURN TOINTEGER(-1 * 2^100)
```

```
>  -9223372036854775808
```

Multi-valued Eigenschaften

Obwohl OpenCypher CREATE keine mehrwertigen Eigenschaften erstellt, können sie in Daten vorkommen, die mit Gremlin (Neptune Database) erstellt wurden, oder beim Laden von Daten (Neptune Database und Neptune Analytics). Wenn Neptune openCypher eine Eigenschaft mit mehreren Werten findet, wird ein Wert zufällig ausgewählt, was zu einem nicht deterministischen Ergebnis führt.

Umgang mit NaN-Werten

Neptuns Umgang mit dem Vergleich von `NaN` Eigenschaftswerten ist nicht definiert. Sich auf solche Vergleiche zu verlassen, kann zu unerwarteten oder nicht deterministischen Ergebnissen führen.

Neptune-Diagrammdatenmodell

Die Grundeinheit von Amazon–Neptune-Diagrammdaten ist ein Element mit vier Positionen (Quad), ähnlich einem Resource Description Framework (RDF)-Quad. Dies sind die vier Positionen eines Neptune-Quads:

- `subject  (S)`
- `predicate  (P)`
- `object  (O)`
- `graph  (G)`

Jedes Quad ist eine Anweisung, die eine Aussage zu mindestens einer Ressource macht. Ein Statement kann beispielsweise das Vorhandensein einer Beziehung zwischen zwei Ressourcen bestätigen oder einer Ressource eine Eigenschaft (Schlüssel-Wert-Paar) anfügen. Sie können sich den Wert des Quad-Prädikats in der Regel als das Verb des Statements vorstellen. Er beschreibt den Typ der Beziehung oder Eigenschaft, die definiert wird. Das Objekt ist das Ziel der Beziehung oder der Wert der Eigenschaft. Im Folgenden sind einige Beispiele aufgeführt:

- Eine Beziehung zwischen zwei Eckpunkten kann dargestellt werden, indem die Quell-Eckpunktkennung an der `S`-Position, die Ziel-Eckpunktkennung an der `O`-Position und die Grenzbezeichnung an der `P`-Position gespeichert wird.
- Eine Eigenschaft kann dargestellt werden, indem die Element-ID an der `S`-Position, der Eigenschaftsschlüssel an der `P`-Position und der Eigenschaftswert an der `O`-Position gespeichert wird.

Die Graphposition `G` wird in den verschiedenen Stacks unterschiedlich verwendet. Hinsichtlich RDF-Daten in Neptune enthält die Position `G` eine benannte Diagramm-ID. Bei Eigenschaftsdiagrammen in Gremlin wird sie verwendet, um im Fall einer Kante den Kanten-ID-Wert zu speichern. In allen anderen Fällen wird standardmäßig ein fester Wert verwendet.

Ein Satz von Quad-Anweisungen mit freigegebenen Ressourcen-IDs erstellt ein Diagramm.

Verzeichnis mit benutzerorientierten Werten

Neptune speichert die meisten benutzerorientierten Werte nicht direkt in den Indizes, die von Neptune verwaltet werden. Stattdessen werden sie getrennt in einem Verzeichnis gespeichert und in den Indizes durch 8-Byte-IDs ersetzt.

- Alle benutzerorientierten Werte, die in S-, P- oder G-Indizes aufgenommen würden, werden auf diese Weise im Verzeichnis gespeichert.
- Im 0-Index werden numerische Werte direkt im Index gespeichert (verknüpft). Dazu gehören auch `date`- und `datetime`-Werte (dargestellt in Millisekunden aus der Epoche).
- Alle anderen benutzerorientierten Werte, die in den 0-Index aufgenommen würden, werden im Verzeichnis gespeichert und im Index mit IDs dargestellt.

Das Verzeichnis enthält eine Vorwärts-Zuordnung benutzerorientierter Werte zu 8-Byte-IDs in einem `value_to_id`-Index.

Es speichert die umgekehrte Zuordnung von 8-Byte-IDs zu Werten in einem von zwei Indizes, abhängig von der Größe der Werte:

- Der `id_to_value`-Index ordnet IDs benutzerorientierten Werten zu, die nach der internen Kodierung kleiner als 767 Byte sind.
- Der `id_to_blob`-Index ordnet IDs benutzerorientierten Werten zu, die größer sind.

Indizierung von Anweisungen in Neptune

Wenn Sie ein Quad-Diagramm abfragen, können Sie für jede Quad-Position eine Werteinschränkung angeben oder nicht. Die Abfrage gibt alle Quads zurück, die den von Ihnen angegebenen Werteinschränkungen entsprechen.

Neptune verwendet Indizes, um Muster von Graphabfragen aufzulösen. Diese Indizes beziehen sich auf die vier Hauptkomponenten einer Graphkante: Subjekt (Quellscheitelpunkt in LPG), Prädikat (RDF) oder Property or Edge Label (LPG), Objekt (Zielscheitelpunkt oder Eigenschaftswert in LPG) und Graph (RDF) oder Edge Identifier (LPG). Für diese vier Positionen der vier Komponenten gibt es 16 (2^4) mögliche Zugriffsmuster. Mithilfe von sechs Indizes können Sie alle 16 Muster effizient abfragen, ohne scannen und filtern zu müssen. Jeder Quad-Anweisungsindex verwendet einen Schlüssel, der aus den vier in einer anderen Reihenfolge verketteten Positionswerten besteht. Eine mögliche Kombination von Indizes mit vier Anweisungen, die alle 16 Zugriffspfade abdecken würden, ist:

```
     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 erstellt und verwaltet standardmäßig nur drei dieser sechs Indizes:

- `SPOG` – Verwendet einen Schlüssel aus `Subject + Predicate + Object + Graph`.
- `POGS` – Verwendet einen Schlüssel aus `Predicate + Object + Graph + Subject`.
- `GPSO` – Verwendet einen Schlüssel aus `Graph + Predicate + Subject + Object`.

Diese drei Indizes bearbeiten viele der gängigsten Zugriffsmuster. Die Verwaltung von nur 3 vollständigen Anweisungsindizes anstelle von 6 reduziert die Ressourcen, die Sie benötigen, um einen schnellen Zugriff ohne Scannen und Filtern zu unterstützen, ungemein. Der `SPOG`-Index ermöglicht beispielsweise eine effiziente Suche, wenn ein Präfix der Positionen, wie z. B. die Eckpunkt- oder die Eckpunkt- und Eigenschaft-ID gebunden ist. Der `POGS`-Index ermöglicht einen effizienten Zugriff, wenn nur die Edge- oder Eigenschaftsbezeichnung, die in der Position `P` gespeichert ist, gebunden ist.

Die Low-Level-API für die Suche nach Anweisungen verwendet ein Anweisungs-Muster, in dem einige Positionen bekannt sind. Die restlichen Positionen werden durch die Indexsuche entdeckt. Auf der Basis der Zusammenstellung der bekannten Positionen in einem Schlüsselpräfix entsprechend der Indexschlüsselreihenfolge für einen der Anweisungsindizes führt Neptune einen Bereichs-Scan aus, um alle Anweisungen abzurufen, die mit den bekannten Positionen übereinstimmen.

Ein Anweisungsindex, den Neptune jedoch nicht standardmäßig erstellt, ist der umgekehrte OSGP-Traversierungsindex, der Prädikate in Objekten und Subjekten sammeln kann. Stattdessen erfasst Neptune Distinct-Prädikate in einem getrennten Index, der für die Ausführung einheitlicher Scans von `{all P x POGS}` verwendet wird. In Gremlin entspricht ein Prädikat einer Eigenschaft oder einer Kantenbeschriftung.

Die Zugriffsstrategie von Neptune kann jedoch ineffizient werden, wenn es eine große Zahl von Distinct-Prädikaten in einem Diagramm gibt. Insbesondere kann ein `in()`-Schritt, bei dem keine Grenzbezeichnungen angegeben werden, oder ein Schritt, der `in()` intern verwendet, z. B. `both()` oder `drop()`, ineffizient sein.

Aktivieren der OSGP-Indexerstellung mittels Labormodus

Wenn Ihr Datenmodell eine große Zahl unterschiedlicher Prädikate erstellt, kann dies zu einer geringeren Leistung und höheren Betriebskosten führen. Dies kann durch die Verwendung des Labor-Modus erheblich verbessert werden, um zusätzlich zu den drei standardmäßig von Neptune verwalteten Indizes den OSGP-Index zu aktivieren.

Die Aktivierung des OSGP-Index kann einige Nachteile haben:

- Die Insert-Rate kann sich um bis zu 23 % verlangsamen.
- Der Speicherbedarf erhöht sich um bis zu 20 %.
- Leseabfragen, die alle Indizes gleichmäßig betreffen (was ziemlich selten ist), können erhöhte Latenzen aufweisen.

Im Allgemeinen lohnt es sich jedoch, den OSGP-Index für DB-Cluster mit einer großen Anzahl unterschiedlicher Prädikate zu aktivieren. Object-based Suchvorgänge werden hocheffizient (z. B. das Auffinden aller Kanten eines Scheitelpunkts oder aller Objekte, die mit einem bestimmten Objekt verbunden sind), wodurch auch das Löschen von Scheitelpunkten viel effizienter wird.

> **Important**
>
> Sie können den OSGP-Index nur in einem leeren DB-Cluster aktivieren, bevor Sie Daten in den Cluster laden.

Gremlin-Anweisungen im Neptune-Datenmodell

Gremlin-Eigenschaftsdiagrammdaten werden im SPOG-Modell mithilfe von drei Anweisungsklassen ausgedrückt:

- Eckpunktbezeichnungs-Anweisungen
- Edge-Anweisungen
- Eigenschaftenanweisungen

Die Beschreibung ihrer Verwendung in Gremlin-Abfragen finden Sie unter Grundlagen der Funktionsweise von Gremlin-Abfragen in Neptune.

Neptune-Nachschlage-Cache kann Leseabfragen beschleunigen

Amazon Neptune implementiert einen Lookup-Cache, der die NVMe-based SSD der `R5d` Instance verwendet, um die Leseleistung für Abfragen mit häufigen, sich wiederholenden Abfragen von Eigenschaftswerten oder RDF-Literalen zu verbessern. Der Nachschlage-Cache speichert diese Werte vorübergehend im NVMe-SSD-Volume, um schnell auf sie zugreifen zu können.

Leseabfragen, die die Eigenschaften einer großen Zahl von Eckpunkten und Kanten oder von zahlreichen RDF-Triples zurückgeben, können eine hohe Latenz aufweisen, wenn die Eigenschaftswerte oder Literale aus Cluster-Speichervolumes statt aus dem Arbeitsspeicher abgerufen werden müssen. Beispiele hierfür sind über eine lange Zeit ausgeführte Leseabfragen, die eine große Zahl von vollständigen Namen aus einem Identitätsdiagramm oder von IP-Adressen aus einem Diagramm zur Betrugserkennung zurückgeben. Wenn die Zahl der von Ihrer Abfrage zurückgegebenen Eigenschaftswerte oder RDF-Literale zunimmt, nimmt der verfügbare Speicher ab und die Leistung der Abfrageausführung kann sich erheblich verschlechtern.

Anwendungsfälle für den Neptune-Nachschlage-Cache

Der Nachschlage-Cache ist nur nützlich, wenn Ihre Leseabfragen die Eigenschaften einer sehr großen Zahl von Eckpunkten und Kanten oder zahlreicher RDF-Triples zurückgeben.

Um die Abfrageleistung zu optimieren, verwendet Amazon Neptune den Instance-Typ `R5d`, um einen großen Cache für solche Eigenschaftswerte oder Literale zu erstellen. Der Abruf aus dem Cache ist erfolgt sehr viel schneller als der Abruf aus Cluster-Speichervolumes.

Als Faustregel gilt, dass sich ein Nachschlage-Cache nur dann lohnt, wenn alle drei der folgenden Bedingungen erfüllt werden:

- Sie haben eine erhöhte Latenz bei Leseabfragen festgestellt.
- Sie beobachten auch einen Rückgang der `BufferCacheHitRatio` CloudWatch Metrik bei der Ausführung von Leseabfragen (siehe). Überwachung von Neptune mit Amazon CloudWatch
- Ihre Leseabfragen verbringen viel Zeit damit, Rückgabewerte zu materialisieren, bevor die Ergebnisse gerendert werden (im Gremlin-profile Beispiel unten erfahren Sie, wie viele Eigenschaftswerte für eine Abfrage materialisiert werden).

> **Note**
>
> Dieses Feature ist nur im oben beschriebenen spezifischen Szenario nützlich. Beispielsweise bietet der Nachschlage-Cache bei Aggregationsabfragen überhaupt keine Vorteile. Wenn Sie keine Abfragen ausführen, die vom Nachschlage-Cache profitieren würden, gibt es keinen Grund für die Verwendung des Instance-Typs `R5d` anstelle des gleichwertigen und kostengünstigeren Instance-Typs `R5`.

Wenn Sie Gremlin verwenden, können Sie die Materialisierungskosten einer Abfrage mit `Gremlin-profile-API` bewerten. Unter „Indexoperationen“ wird die Anzahl der Begriffe angezeigt, die während der Ausführung materialisiert wurden:

```
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
```

Die Anzahl der materialisierten nichtnumerischen Begriffe ist direkt proportional zur Anzahl der Begriffssuchen, die Neptune ausführen muss.

Verwenden des Nachschlage-Caches

Der Nachschlage-Cache ist nur für den Instance-Typ `R5d` verfügbar, für den er standardmäßig automatisch aktiviert ist. `R5d`Neptune-Instances haben dieselben Spezifikationen wie `R5` Instances und zusätzlich bis zu 1,8 TB lokalen NVMe-based SSD-Speicher. Nachschlage-Caches sind Instance-spezifisch. Workloads, die von ihnen profitieren, können speziell an `R5d`-Instances in einem Neptune-Cluster geleitet werden, während andere Workloads an `R5` oder andere Instance-Typen geleitet werden können.

Um den Nachschlage-Cache auf einer Neptune-Instance zu verwenden, aktualisieren Sie diese Instance einfach auf den Instance-Typ `R5d`. Wenn Sie dies tun, setzt Neptune den

neptune_lookup_cache DB-Cluster-Parameter automatisch auf 1 (aktiviert) und erstellt den Lookup-Cache für diese bestimmte Instance. Sie können anschließend die Instance-Status-API verwenden, um zu bestätigen, dass der Cache aktiviert wurde.

Ähnlich können Sie eine Instance von einem R5d-Instance-Typ auf einen entsprechenden R5-Instance-Typ herunterskalieren. um den Nachschlage-Cache für diese Instance zu deaktivieren.

Wenn eine R5d-Instance gestartet wird, ist der Nachschlage-Cache aktiviert und im Kaltstartmodus. Das bedeutet, dass er leer ist. Neptune sucht während der Verarbeitung von Abfragen zunächst im Nachschlage-Cache nach Eigenschaftswerten oder RDF-Literalen und fügt sie hinzu, wenn sie noch nicht vorhanden sind. Dies führt zu einer allmählichen „Aufwärmung“ des Cache.

Wenn Sie die Leseabfragen, die Eigenschaftswerte oder RDF-literal Lookups erfordern, an eine R5d-Reader-Instance weiterleiten, verschlechtert sich die Leseleistung leicht, während sich der Cache aufwärmt. Wenn der Cache aufgewärmt wird, erhöht sich die Leseleistung jedoch erheblich, und es kann auch zu einem Rückgang der I/O Kosten kommen, wenn Suchvorgänge den Cache und nicht den Clusterspeicher betreffen. Die Speicherauslastung verbessert sich ebenfalls.

Wenn es sich bei Ihrer Schreib-Instance um eine R5d-Instance handelt, wird ihr Nachschlage-Cache bei jeder Schreiboperation automatisch aufgewärmt. Dieser Ansatz erhöht zwar geringfügig die Latenz für Schreibabfragen, wärmt den Nachschlage-Cache jedoch effizienter auf. Wenn Sie dann die Leseabfragen, für die Eigenschaftswerte oder RDF-literal Lookups erforderlich sind, an die Writer-Instanz weiterleiten, wird die Leseleistung sofort verbessert, da die Werte dort bereits zwischengespeichert wurden.

Wenn Sie den Bulk-Loader auf einer R5d-Schreib-Instance ausführen, stellen Sie möglicherweise fest, dass die Leistung aufgrund des Caches leicht beeinträchtigt ist.

Da der Nachschlage-Cache für jeden Knoten spezifisch ist, wird der Cache bei Ersetzung des Hosts zum Kaltstart zurückgesetzt.

Sie können den Lookup-Cache für alle Instances in Ihrem DB-Cluster vorübergehend deaktivieren, indem Sie den neptune_lookup_cache DB-Cluster-Parameter auf 0 (deaktiviert) setzen. Im Allgemeinen ist es jedoch sinnvoller, den Cache auf bestimmten Instances zu deaktivieren, indem Sie diese von R5d auf R5 Instance-Typen herunterskalieren.

Transaktionssemantik in Neptune

Amazon Neptune wurde entwickelt, um hochgradig gleichzeitige Online-Transactional-Processing (OLAP)-Workloads über Datendiagramme zu unterstützen. Die W3C SPARQL Query Language for RDF-Spezifikation und die Apache TinkerPop Gremlin Graph Traversal Language-Dokumentation definieren keine Transaktionssemantik für die gleichzeitige Abfrageverarbeitung. Da ACID-Unterstützung und gut definierte Transaktionsgarantien sehr wichtig sein können, setzen wir eine strikte Semantik durch, um Datenanomalien zu vermeiden.

Dieser Abschnitt definiert diese Semantik und zeigt ihre Anwendung auf einige häufige Anwendungsfälle in Neptune.

Topics

- Definition von Isolationsstufen
- Transaktionsisolierungsstufen in Neptune
- Beispiele für die Neptune-Transaktionssemantik

Definition von Isolationsstufen

Das „I“ in `ACID` steht für Isolation. Der Grad der Isolation einer Transaktion bestimmt, wie stark sich andere gleichzeitige Transaktionen auf die Daten auswirken können, auf denen sie ausgeführt wird.

In der SQL:1992Norm wurde ein Vokabular zur Beschreibung von Isolationsstufen geschaffen. Er definiert drei Arten von Interaktionen (als Phänomene bezeichnet), die zwischen zwei gleichzeitigen Transaktionen auftreten können, `Tx1` und `Tx2`:

- `Dirty read` – Dies tritt auf, wenn `Tx1` ein Element ändert und `Tx2` dieses Element liest, bevor `Tx1` ein Commit für die Änderung ausgeführt hat. Wenn die `Tx1` kein Commit für die Änderung erfolgreich durchführen kann oder sie zurückabwickelt, hat `Tx2` einen Wert gelesen, der nie in die Datenbank gelangt ist.
- `Non-repeatable read` – Dies tritt auf, wenn `Tx1` ein Element liest, `Tx2` dieses Element ändert oder löscht und ein Commit für die Änderung ausführt und `Tx1` versucht, das Element erneut zu lesen. `Tx1` liest jetzt einen anderen Wert als zuvor oder stellt fest, dass das Element nicht mehr vorhanden ist.
- `Phantom read` – Dies tritt auf, `Tx1` einen Satz von Elementen liest, die ein Suchkriterium erfüllen, `Tx2` ein neues Element hinzufügt, das dem Suchkriterium entspricht, und `Tx1` die Suche wiederholt. `Tx1` erhält jetzt einen anderen Satz von Elementen als zuvor.

Jede dieser drei Interaktionsarten kann zu Inkonsistenzen in den resultierenden Daten in einer Datenbank führen.

In der SQL:1992 Norm wurden vier Isolationsstufen definiert, für die unterschiedliche Garantien in Bezug auf die drei Arten von Interaktionen und die damit verbundenen Inkonsistenzen gelten. Auf allen vier Stufen kann garantiert werden, dass eine Transaktion vollständig oder gar nicht ausgeführt wird:

- `READ UNCOMMITTED` – Unterstützt alle drei Interaktionsarten (d. h. „Dirty“-Lesevorgänge, nicht wiederholbare Lesevorgänge und „Phantomlesevorgänge“).
- `READ COMMITTED` – „Dirty“-Lesevorgänge sind nicht möglich, nicht wiederholbare und Phantomlesevorgänge sind jedoch möglich.
- `REPEATABLE READ` – Weder „Dirty“-Lesevorgänge noch nicht wiederholbare Lesevorgänge sind möglich, Phantomlesevorgänge sind jedoch möglich.
- `SERIALIZABLE` – Keine der drei Interaktionsarten ist möglich.

Multiversion Concurrency Control (MVCC) ermöglicht eine andere Art von Isolation, nämlich die SNAPSHOT-Isolation. Auf diese Weise wird sichergestellt, dass eine Transaktion mit einem Snapshot der Daten arbeitet, der beim Start der Transaktion vorhanden ist, und dass keine andere Transaktion diesen Snapshot ändern kann.

Transaktionsisolierungsstufen in Neptune

Amazon Neptune implementiert unterschiedliche Transaktionsisolationsstufen für schreibgeschützte Abfragen und Mutationsabfragen. SPARQL- und Gremlin-Abfragen werden basierend auf den folgenden Kriterien als schreibgeschützte oder als Mutationsabfragen klassifiziert:

- In SPARQL gibt es einen deutlichen Unterschied zwischen Lesevorgängen (`SELECT`, `ASK`, `CONSTRUCT` und `DESCRIBE` wie in der SPARQL 1.1 Query Language-Spezifikation definiert) und Mutationsabfragen (`INSERT` und `DELETE` wie in der SPARQL 1.1 Update-Spezifikation definiert).

  Beachten Sie, dass Neptune mehrere Mutationsabfragen, die zusammen gesendet werden (z. B. in einer `POST`-Nachricht, getrennt durch Semikolon), als eine einzige Transaktion behandelt. Sie sind gemeinsam entweder erfolgreich oder nicht erfolgreich. Wenn sie nicht erfolgreich sind, werden teilweise Änderungen zurückgesetzt.
- In Gremlin klassifiziert Neptune eine Abfrage jedoch als schreibgeschützte Abfrage oder als Mutationsabfrage, abhängig davon, ob sie Abfragepfadschritte wie `addE()`, `addV()`, `property()`

oder `drop()` enthält, die Daten manipulieren. Wenn die Abfrage einen solchen Pfadschritt enthält, wird sie als Mutationsabfrage klassifiziert und ausgeführt.

Es ist auch möglich, stehende Sitzungen in Gremlin zu verwenden. Weitere Informationen finden Sie unter Skriptbasierte Gremlin-Sitzungen. In diesen Sitzungen werden alle Abfragen (einschließlich schreibgeschützter Abfragen) mit derselben Isolierung wie Mutationsabfragen auf dem Writer-Endpunkt ausgeführt.

Bei Verwendung von Bolt-Lese-Schreib-Vorgängen in openCypher werden alle Abfragen (einschließlich schreibgeschützter Abfragen) mit derselben Isolierung wie Mutationsabfragen auf dem Writer-Endpunkt ausgeführt.

Themen

- Read-only Abfrageisolierung in Neptune
- Isolierung von Mutationsabfragen in Neptune
- Konfliktlösung mithilfe von Timeouts Lock-Wait
- Bereichssperren und falsche Konflikte

Read-only Abfrageisolierung in Neptune

Neptune evaluiert schreibgeschützte Abfragen gemäß der Snapshot-Isolationssemantik. Das bedeutet, dass eine schreibgeschützte Abfrage logisch auf einem konsistenten Snapshot der Datenbank ausgeführt wird, wenn die Abfrageevaluierung beginnt. Neptune kann dann garantieren, dass keines der folgenden Phänomene eintritt:

- `Dirty reads`— Read-only Abfragen in Neptune werden niemals unbestätigte Daten aus einer gleichzeitigen Transaktion sehen.
- `Non-repeatable reads` – Eine schreibgeschützte Transaktion, die dieselben Daten mehrmals liest, erhält stets dieselben Werte zurück.
- `Phantom reads` – Eine schreibgeschützte Transaktion liest niemals Daten, die nach Beginn der Transaktion hinzugefügt wurden.

Da die Snapshot-Isolierung durch Multiversion Concurrency Control (MVCC) erzielt wird, müssen schreibgeschützte Abfragen keine Daten sperren und blockieren daher keine Mutationsabfragen.

Read Replicas akzeptieren nur schreibgeschützte Abfragen, sodass alle Abfragen für Read Replicas gemäß der SNAPSHOT-Isolationssemantik ausgeführt werden.

Der einzige zusätzliche Aspekt bei der Abfrage einer Read Replica besteht darin, dass es eine kleine Verzögerung bei der Replikation zwischen dem Writer und den Read Replicas geben kann. Dies bedeutet, dass eine Aktualisierung, die am Writer vorgenommen wurde, einige Zeit in Anspruch nehmen kann, bis sie auf die Read Replica übertragen wird, aus der Sie lesen. Die tatsächliche Replikationszeit ist von der Schreiblast der primären Instance abhängig. Die Neptune-Architektur unterstützt die Replikation mit niedriger Latenz, und die Replikationsverzögerung wird in einer Amazon-Metrik instrumentiert. CloudWatch

Aufgrund der SNAPSHOT-Isolationsstufe wird Leseabfragen stets ein konsistenter Zustand der Datenbank angezeigt, auch wenn es sich nicht um den neuesten Zustand handelt.

Wenn Sie eine starke Garantie benötigen, dass eine Abfrage das Ergebnis einer vorherigen Aktualisierung beachtet, senden Sie die Abfrage an den Writer-Endpunkt selbst und nicht an eine Read Replica.

Isolierung von Mutationsabfragen in Neptune

Als Teil von Mutationsabfragen erstellte Lesevorgänge werden unter `READ COMMITTED`-Transaktionsisolierung ausgeführt, was die Möglichkeit von fehlerhaften Lesevorgängen ausschließt. Abgesehen von den üblichen Garantien für die `READ COMMITTED`-Transaktionsisolierung bietet Neptune eine starke Garantie, dass weder `PHANTOM`- noch `NON-REPEATABLE`- Lesevorgänge erfolgen können.

Diese starken Garantien werden erreicht, indem Datensätze und Bereiche von Datensätzen beim Lesen von Daten gesperrt werden. Dadurch wird verhindert, dass gleichzeitige Transaktionen Einfügungen oder Löschungen in Indexbereichen vornehmen, nachdem sie gelesen wurden, was wiederholbare Lesevorgänge garantiert.

> **Note**
>
> Eine gleichzeitige Mutationstransaktion `Tx2` könnte jedoch nach dem Start der Mutationstransaktion `Tx1` beginnen und das Commit einer Änderung durchführen, bevor `Tx1` Daten zum Lesen gesperrt hatte. In diesem Fall würde `Tx1` die Änderung von `Tx2` so sehen, als ob `Tx2` sie vor dem Start von `Tx1` abgeschlossen hätte. Da dies nur für Änderungen gilt, für die ein Commit durchgeführt wurde, kann ein `dirty read` nie auftreten.

Um den Sperrmechanismus zu verstehen, den Neptune für Mutationsabfragen verwendet, sollten zunächst die Details für das Diagrammdatenmodell und die Indizierungsstrategie in Neptune verstanden werden. Neptune verwaltet Daten mithilfe von drei Indizes, `SPOG`, `POGS` und `GPSO`.

Um wiederholbare Lesevorgänge für die `READ COMMITTED`-Transaktionsebene zu erzielen, nutzt Neptune Bereichssperren im verwendeten Index. Wenn beispielsweise eine Mutationsabfrage alle Eigenschaften und ausgehenden Grenzen eines Eckpunkts mit dem Namen `person1` liest, würde der Knoten den gesamten durch das Präfix `S=person1` im `SPOG`- Index definierten Bereich sperren, bevor die Daten gelesen werden.

Derselbe Mechanismus gilt bei der Verwendung anderer Indizes. Wenn beispielsweise eine Mutationstransaktion alle Quell-Ziel-Eckpunktpaare für eine bestimmte Grenzbezeichnung mit dem `POGS`-Index absucht, wäre der Bereich für die Grenzbezeichnung in der `P`-Position gesperrt. Jede gleichzeitige Transaktion, unabhängig davon, ob es sich um eine schreibgeschützte oder eine Mutationsabfrage handelt, könnte dennoch Lesevorgänge innerhalb des gesperrten Bereichs durchführen. Jede Mutation, die das Einfügen oder Löschen neuer Datensätze im gesperrten Präfixbereich beinhaltet, erfordert jedoch eine exklusive Sperre und würde verhindert werden.

Anders gesagt: Wenn ein Bereich des Indexes von einer Mutationstransaktion gelesen wurde, gibt es eine starke Garantie, dass dieser Bereich bis zum Ende der Lesetransaktion nicht durch gleichzeitige Transaktionen geändert wird. Auf diese Weise wird sichergestellt, dass kein `non-repeatable reads` auftritt.

Konfliktlösung mithilfe von Timeouts Lock-Wait

Wenn eine zweite Transaktion versucht, einen Datensatz in einem Bereich zu ändern, den eine erste Transaktion gesperrt hat, erkennt Neptune den Konflikt sofort und blockiert die zweite Transaktion.

Wenn keine Abhängigkeitssperre erkannt wird, wendet Neptune automatisch eine Sperrwartezeitüberschreitung an, bei dem die blockierte Transaktion bis zu 60 Sekunden wartet, bis die sperrende Transaktion abgeschlossen ist und die Sperre aufgehoben wurde.

- Wenn die Sperrwartezeit abläuft, bevor die Sperre aufgehoben wird, wird die blockierte Transaktion rückabgewickelt.
- Wenn die Sperre innerhalb des Sperrwartezeitraums aufgehoben wird, wird die zweite Transaktion entsperrt und kann erfolgreich abgeschlossen werden, ohne dass ein erneuter Versuch erfolgen muss.

Wenn Neptune jedoch eine Abhängigkeitssperre zwischen den beiden Transaktionen erkennt, ist eine automatische Beseitigung des Konflikts nicht möglich. In diesem Fall bricht Neptune eine der beiden Transaktionen sofort ab und führt ein Rollback für diese Transaktion aus, ohne eine Sperrwartezeitüberschreitung zu initiieren. Neptune versucht dabei, die Transaktion rückgängig zu machen, in der die wenigsten Datensätze eingefügt oder gelöscht wurden.

Messung der Wartezeit auf Sperren (Engine ≥ 1.4.5.0)

Ab Engine-Version 1.4.5.0 können Sie mithilfe von zwei Protokollzählern für langsame Abfragen genau beobachten, wie lange eine Mutationsabfrage blockiert wurde:

Zähler	Description
`sharedLocksWaitTimeMillis`	Wartezeit auf gemeinsame Sperren (S), die mehrere Lesegeräte zulassen, aber Schreibvorgänge blockieren.
`exclusiveLocksWaitTimeMillis`	Wartezeit auf exklusive Sperren (X), die jeden anderen Zugriff blockieren.

Diese beiden Felder werden im `storageCounters` Objekt nur angezeigt, wenn Sie den `debug` Anmeldemodus für langsame Abfragen aktivieren ()`neptune_enable_slow_query_log=debug`.

> **Tip**
>
> `sharedLocksWaitTimeMillis + exclusiveLocksWaitTimeMillis`Nähert man sich dem der Abfrage`overallRunTimeMs`, entsteht bei der Abfrage ein Engpass durch einen Sperrkonflikt und nicht durch CPU, Netzwerk oder. I/O

Praktische Tipps zur Reduzierung von Konflikten:

- Staffelung widersprüchlicher Jobs — Führen Sie umfangreiche Batch-Mutationen in Zeiten geringerer Benutzeraktivität durch.
- Teilen Sie große Mutationen in kleinere Teile auf — Kleinere Transaktionen enthalten Sperren für kürzere Zeit, wodurch die Wahrscheinlichkeit von Timeouts verringert wird.

Bereichssperren und falsche Konflikte

Neptune führt Bereichssperren mithilfe von Gap-Sperren aus. Eine Gap-Sperre ist eine Sperre für eine Lücke zwischen Indexdatensätzen oder eine Sperre für eine Lücke vor dem ersten oder nach dem letzten Indexdatensatz.

Neptune verwendet eine sogenannte Verzeichnistabelle, um bestimmten Zeichenkettenliteralen numerische ID-Werte zuzuordnen. Dies ist ein Beispielstatus eines solchen Neptun-Verzeichnisses: Tabelle:

Zeichenfolge	ID (ID)
type	1
default_graph	2
person_3	3
person_1	5
knows	6
person_2	7
age	8
edge_1	9
lives_in	10
New York	11
Person	12
Place	13
edge_2	14

Die Zeichenfolgen oben gehören zu einem Eigenschaftsdiagrammmodell. Die Konzepte gelten jedoch für alle RDF-Diagrammmodelle.

Der entsprechende Status des SPOG-Index (Subject-Predicate-Object_Graph) ist unten links dargestellt. Rechts werden die entsprechenden Zeichenfolgen gezeigt, um die Bedeutung der Indexdaten zu vermitteln.

S (ID)	P (ID)	O (ID)	G (ID)		S (Zeichenfolge)	P (Zeichenfolge)	O (Zeichenfolge)	G (Zeichenfolge)
3	1	12	2		person_3	type	Person	default_graph
5	1	12	2		person_1	type	Person	default_graph
5	6	3	9		person_1	knows	person_3	edge_1
5	8	40	2		person_1	age	40	default_graph
5	10	11	14		person_1	lives_in	New York	edge_2
7	1	12	2		person_2	type	Person	default_graph
11	1	13	2		New York	type	Place	default_graph

Wenn beispielsweise eine Mutationsabfrage alle Eigenschaften und ausgehenden Kanten eines Eckpunkts mit dem Namen `person_1` liest, würde der Knoten den gesamten durch das Präfix `S=person_1` im SPOG-Index definierten Bereich sperren, bevor die Daten gelesen werden.
Die Bereichssperre würde Gap-Sperren für alle übereinstimmenden Datensätze und den ersten Datensatz einrichten, der nicht übereinstimmt. Übereinstimmende Datensätze würden gesperrt. Nicht übereinstimmende Datensätze würden nicht gesperrt. Neptune würde die Gap-Sperren wie folgt platzieren:

- `5 1 12 2` (Lücke 1)
- `5 6 3 9` (Lücke 2)

- `5 8 40 2` (Lücke 3)
- `5 10 11 14` (Lücke 4)
- `7 1 12 2` (Lücke 5)

Dies sperrt die folgenden Datensätze:

- `5 1 12 2`
- `5 6 3 9`
- `5 8 40 2`
- `5 10 11 14`

In diesem Zustand werden die folgenden Operationen berechtigterweise blockiert:

- Einfügen einer neuen Eigenschaft oder Kante für `S=person_1`. Eine neue Eigenschaft, die sich von `type` unterscheidet, oder eine neue Kante müsste in Lücke 2, Lücke 3, Lücke 4 oder Lücke 5 eingefügt werden, die alle gesperrt sind.
- Löschen eines der vorhandenen Datensätze.

Gleichzeitig würden einige gleichzeitige Operationen fälschlicherweise blockiert (was zu falschen Konflikten führen würde):

- Alle Eigenschaften oder Kanteneinfügungen für `S=person_3` werden blockiert, da sie in Lücke 1 platziert werden müssten.
- Jede neue Eckpunkteinfügung, der eine ID zwischen 3 und 5 zugewiesen wird, würde blockiert, da sie in Lücke 1 eingefügt werden müsste.
- Jede neue Eckpunkteinfügung, der eine ID zwischen 5 und 7 zugewiesen wird, würde blockiert, da sie in Lücke 5 eingefügt werden müsste.

Gap-Sperren sind nicht präzise genug, um die Lücke für ein einzelnes bestimmtes Prädikat zu sperren (um z. B. Lücke 5 für Prädikat `S=5` zu sperren).

Die Bereichssperren werden nur in dem Index platziert, in dem der Lesevorgang stattfindet.
Im Beispiel oben sind Datensätze nur im SPOG-Index gesperrt, nicht in POGS oder GPSO.
Lesevorgänge für eine Abfrage können für alle Indizes ausgeführt werden, abhängig von den

Zugriffsmustern, die mithilfe der `explain`-APIs (für Sparql und für Gremlin) aufgelistet werden können.

> **Note**
>
> Gap-Sperren können auch als sichere gleichzeitige Aktualisierungen der zugrunde liegenden Indizes verstanden werden, was ebenfalls zu falschen Konflikten führen kann. Diese Gap-Sperren werden unabhängig von der Isolationsstufe oder den von der Transaktion ausgeführten Lesevorgängen platziert.

Falsche Konflikte können nicht nur auftreten, wenn gleichzeitige Transaktionen aufgrund von Lückensperren kollidieren, sondern in einigen Fällen auch, wenn eine Transaktion nach einem Fehler erneut versucht wird. Wenn das durch den Fehler ausgelöste Rollback noch nicht abgeschlossen ist und die Sperren, die zuvor auf die Transaktion angewendet wurden, noch nicht vollständig aufgehoben wurden, stößt der erneute Versuch auf einen falschen Konflikt und schlägt fehl.

Bei hoher Auslastung stellen Sie in der Regel fest, dass 3 bis 4 % der Schreibabfragen aufgrund falscher Konflikte fehlschlagen. Für einen externen Client sind diese falschen Konflikte schwer vorherzusagen und sollten mithilfe von Wiederholungsversuchen behandelt werden.

Beispiele für die Neptune-Transaktionssemantik

Die folgenden Beispiele zeigen verschiedene Anwendungsfälle für die Transaktionssemantik in Amazon Neptune.

Themen

- Beispiel 1 – Einfügen einer Eigenschaft nur in dem Fall, dass sie nicht vorhanden ist
- Beispiel 2 – Feststellung, dass ein Eigenschaftswert global eindeutig ist
- Beispiel 3 – Ändern einer Eigenschaft, wenn eine andere Eigenschaft einen bestimmten Wert hat
- Beispiel 4 – Ersetzen einer vorhandenen Eigenschaft
- Beispiel 5 – Vermeiden hängender Eigenschaften oder Kanten

Beispiel 1 – Einfügen einer Eigenschaft nur in dem Fall, dass sie nicht vorhanden ist

Angenommen, Sie möchten sicherstellen, dass eine Eigenschaft nur einmal angegeben wird. Nehmen Sie beispielsweise an, mehrere Abfragen versuchen, einer Person gleichzeitig eine

Kreditbewertung zuzuweisen. Sie möchten nur eine Instance der Eigenschaft einfügen und die anderen Abfragen fehlschlagen lassen, da die Eigenschaft bereits festgelegt wurde.

```
# 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 .} }
```

Der Gremlin `property()`-Schritt fügt eine Eigenschaft mit dem angegebenen Schlüssel und Wert ein. Der `coalesce()`-Schritt führt das erste Argument im ersten Schritt aus, und wenn dies fehlschlägt, wird der zweite Schritt ausgeführt:

Bevor der Wert für die `creditScore`-Eigenschaft für einen bestimmten `person1`-Eckpunkt eingefügt wird, muss eine Transaktion versuchen, den möglicherweise nicht vorhandenen `creditScore`-Wert für `person1` zu lesen. Dieser versuchte Lesevorgang sperrt den SP- Bereich für `S=person1` und `P=creditScore` im SPOG-Index, in dem der `creditScore`-Wert entweder vorhanden ist oder geschrieben werden wird.

Durch diese Bereichssperre wird verhindert, dass gleichzeitige Transaktionen gleichzeitig einen `creditScore`- Wert einfügen. Wenn es mehrere parallele Transaktionen gibt, kann höchstens eine von ihnen den Wert zu einem bestimmten Zeitpunkt aktualisieren. Dies schließt die Anomalie der Erstellung mehr als einer `creditScore`-Eigenschaft aus.

Beispiel 2 – Feststellung, dass ein Eigenschaftswert global eindeutig ist

Angenommen, Sie möchten eine Person mit einer Sozialversicherungsnummer als Primärschlüssel einfügen. Sie möchten, dass Ihre Mutationsabfrage sicherstellt, dass auf globaler Ebene niemand sonst in der Datenbank dieselbe Sozialversicherungsnummer hat:

```
# 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 } }
```

Dieses Beispiel ähnelt dem vorherigen. Der Hauptunterschied besteht darin, dass die Bereichssperre für den POGS-Index und nicht für den SPOG-Index gesetzt wird.

Die Transaktion, die die Abfrage ausführt, muss das Muster, `?person :ssn 123456789`, lesen, in dem die Positionen P und 0 gebunden sind. Die Bereichssperre wird für den POGS-Index für `P=ssn` und `O=123456789` gesetzt.

- Wenn das Muster vorhanden ist, wird keine Aktion ausgeführt.
- Wenn es nicht vorhanden ist, verhindert die Sperre, dass gleichzeitige Transaktionen auch diese Sozialversicherungsnummer einfügen.

Beispiel 3 – Ändern einer Eigenschaft, wenn eine andere Eigenschaft einen bestimmten Wert hat

Nehmen wir an, dass verschiedene Ereignisse in einem Spiel eine Person von Level 1 auf Level 2 verschieben und ihr eine neue `level2Score`-Eigenschaft zuweisen, die auf Null gesetzt ist. Sie müssen sicherstellen, dass nicht mehrere gleichzeitige Instances einer solchen Transaktion mehrere Instances der Score-Eigenschaft von Level 2 erstellen können. Die Abfragen in Gremlin und SPARQL können wie folgt aussehen.

```
# 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 .}
```

Wenn in Gremlin `Cardinality.single` angegeben ist, fügt der `property()`- Schritt entweder eine neue Eigenschaft hinzu oder ersetzt einen vorhandenen Eigenschaftswert durch den neuen angegebenen Wert.

Jede Aktualisierung eines Eigenschaftswerts, z. B. das Erhöhen `level` von 1 auf 2, wird als Löschung des aktuellen Datensatzes und Einfügen eines neuen Datensatzes mit dem neuen Eigenschaftswert implementiert. In diesem Fall wird der Datensatz für Level 1 gelöscht, und ein Datensatz für Level 2 wird neu gesetzt.

Damit die Transaktion `level2Score` hinzufügen und `level` von 1 auf 2 aktualisieren kann, muss sie zunächst überprüfen, ob der `level`-Wert derzeit gleich 1 ist. Dafür wird eine Bereichssperre für das SPO-Präfix für `S=person1`, `P=level`, und `O=1` im SPOG-Index gesetzt. Diese Sperre verhindert, dass gleichzeitige Transaktionen das Version 1-Tripel löschen. Daher können keine widersprüchlichen gleichzeitigen Aktualisierungen stattfinden.

Beispiel 4 – Ersetzen einer vorhandenen Eigenschaft

Bestimmte Ereignisse können die Kreditbewertungen einer Person auf einen neuen Wert aktualisieren (hier BBB). Sie möchten jedoch sicherstellen, dass gleichzeitige Ereignisse dieses Typs nicht mehrere Kreditbewertungseigenschaften für eine Person erstellen können.

```
# 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 .}
```

Dieser Fall ist Beispiel 3 ähnlich, mit der Ausnahme, dass Neptune nicht das SPO-Präfixe sperrt, sondern das Präfix SP nur bei `S=person1` und `P=creditScore` sperrt. Dadurch wird verhindert, dass gleichzeitige Transaktionen Tripel mit der `creditScore`-Eigenschaft für das `person1`-Subjekt einfügen oder löschen.

Beispiel 5 – Vermeiden hängender Eigenschaften oder Kanten

Die Aktualisierung einer Entität sollte kein hängendes Element zurücklassen, d. h. eine Eigenschaft oder Grenze, die einer nicht typisierten Entität zugeordnet ist. Dies ist nur ein Problem in SPARQL, da Gremlin über integrierte Einschränkungen verfügt, die hängende Elemente verhindern.

```
# SPARQL:
```

```
tx1: INSERT { :person1 :age 23 } WHERE { :person1 rdf:type :Person }
tx2: DELETE { :person1 ?p ?o }
```

Die INSERT-Abfrage muss das SPO-Präfix mit S=person1, P=rdf:type und O=Person im SPOG-Index lesen und sperren. Diese Sperre verhindert, dass die DELETE-Abfrage parallel erfolgreich ist.

Beim „Rennen“ zwischen dem Versuch der DELETE- Abfrage, den :person1 rdf:type :Person-Datensatz zu löschen, und dem der INSERT-Abfrage, den Datensatz zu lesen und eine Bereichssperre für ihren SPO im SPOG-Index zu setzen, sind die folgenden Ergebnisse möglich:

- Wenn die INSERT-Abfrage ein Commit durchführt, bevor die DELETE-Abfrage alle Datensätze für :person1 liest und löscht, wird :person1 vollständig aus der Datenbank entfernt, einschließlich des neu eingefügten Datensatzes.
- Wenn die DELETE-Abfrage ein Commit durchführt, bevor die INSERT-Abfrage versucht, den :person1 rdf:type :Person-Datensatz zu lesen, beachtet der Lesevorgang die festgeschriebene Änderung. Das heißt, er findet keinen :person1 rdf:type :Person-Datensatz und wird daher nicht aktiv („No-op“).
- Wenn die INSERT-Abfrage liest, bevor die DELETE-Abfrage dies tut, wird das :person1 rdf:type :Person-Tripel gesperrt, und die DELETE- Abfrage wird blockiert, bis die INSERT-Abfrage ein Commit durchführt, wie im ersten Fall zuvor.
- Wenn DELETE vor der INSERT-Abfrage liest und die INSERT-Abfrage versucht, zu lesen und eine Sperre auf das SPO-Präfix für den Datensatz zu setzen, wird ein Konflikt erkannt. Der Grund hierfür ist, dass das Tripel zum Entfernen markiert wurde und der INSERT dann fehlschlägt.

Alle diese verschiedenen möglichen Ereignissequenzen führen nicht zum Erstellen hängender Grenzen.

Amazon-Neptune-DB-Cluster und -Instances

Ein Amazon Neptune-DB-Cluster verwaltet den Zugriff auf Ihre Daten über Abfragen. Ein Cluster besteht aus:

- Einer primären DB-Instance.
- Bis zu 15 Read-Replica-DB-Instances.

Alle Instances in einem Cluster nutzen dasselbe zugrunde liegende verwaltete Speichervolumen, das auf Zuverlässigkeit und hohe Verfügbarkeit ausgelegt ist.

Sie stellen die Verbindungen zu den DB-Instances in Ihrem DB-Cluster über Neptune-Endpunkte her.

Primäre DB-Instance in einem Neptune-DB-Cluster

Die primäre DB-Instance koordiniert alle Schreiboperationen auf dem Speichervolume, der dem DB-Cluster zugrunde liegt. Sie unterstützt auch Leseoperationen.

In einem Neptune-DB-Cluster kann es nur eine primäre DB-Instance geben. Wenn die primäre Instance nicht verfügbar ist, wechselt Neptune automatisch zu einer der Read-Replica-Instances mit einer Priorität, die Sie angeben können.

Read-replica DB-Instances in einem Neptune-DB-Cluster

Nach der Erstellung der primären Instance für einen DB-Cluster können Sie bis zu 15 Read-Replica-Instances im DB-Cluster erstellen, um schreibgeschützte Abfragen zu unterstützen.

Neptune-Read-Replica-DB-Instances sind gut für die Skalierung der Lesekapazitäten geeignet, da sie im Cluster-Volume vollständig für Leseoperationen dediziert sind. Alle Schreiboperationen werden von der primären Instance verwaltet. Jede Read-Replica-DB-Instance hat einen eigenen Endpunkt.

Da das Cluster-Speichervolume von allen Instances in einem Cluster gemeinsam genutzt wird, geben alle Read-Replica-Instances dieselben Daten für Abfrageergebnisse mit einer sehr geringen Replikationsverzögerung zurück. Diese Verzögerung beträgt in der Regel sehr viel weniger als 100 Millisekunden nach dem Schreiben eines Updates durch die primäre Instance. Sie kann jedoch etwas länger sein, wenn es sehr viele Schreiboperationen gibt.

Wenn eine oder mehrere Read-Replica-Instances in verschiedenen Availability Zones verfügbar sind, kann dies die Verfügbarkeit erhöhen, da Read-Replicas als Failover-Ziele für die primäre

Instance dienen. Wenn die primäre Instance ausfällt, stuft Neptune eine Read-Replica-Instance zur primären Instance herauf. Es gibt eine kurze Unterbrechung, während die heraufgestufte Instance neu gestartet wird. Während dieser Zeit können Lese- und Schreibanfragen für die primäre Instance mit einer Ausnahmemeldung fehlschlagen.

Wenn Ihr DB-Cluster dagegen keine Read-Replica-Instances enthält, bleibt Ihr DB-Cluster auch dann nicht verfügbar, wenn die primäre Instance ausfällt, bis sie neu erstellt wurde. Re-creating Die primäre Instance dauert erheblich länger als das Heraufstufen einer Read-Replica.

Um eine hohe Verfügbarkeit zu gewährleisten, sollten Sie eine oder mehrere Read-Replica-Instances erstellen, die dieselbe DB-Instance-Klasse wie die primäre Instance besitzen, sich jedoch in anderen Availability Zones befinden. Siehe Fehlertoleranz für einen Neptune-DB-Cluster.

Mithilfe der Konsole können Sie ein Multi-AZ Deployment erstellen, indem Sie Multi-AZ bei der Erstellung eines DB-Clusters einfach angeben. Wenn sich ein DB-Cluster in einer einzelnen Availability Zone befindet, können Sie ihn zu einem Multi-AZ DB-Cluster machen und ein Neptune-Replikat in einer anderen Availability Zone hinzufügen.

> **Note**
>
> Sie können keine verschlüsselte Read-Replica-Instance für einen unverschlüsselten Neptune-DB-Cluster oder eine unverschlüsselte Read-Replica-Instance für einen verschlüsselten Neptune-DB-Cluster erstellen.

Informationen zum Erstellen einer Read-Replica-DB-Instance in Neptune finden Sie unter Erstellen einer Neptune-Reader-Instance über die Konsole.

Dimensionieren von DB-Instances in einem Neptune-DB-Cluster

Dimensionieren Sie die Instances in Ihrem Neptune-DB-Cluster entsprechend Ihren CPU- und Speicheranforderungen. Die Anzahl der vCPUs in einer Instance bestimmt die Anzahl der Abfrage-Threads, die eingehende Abfragen verarbeiten. Die Größe des Arbeitsspeichers einer Instance bestimmt die Größe des Puffer-Caches, der zum Speichern von Kopien der Datenseiten verwendet wird, die aus dem zugrunde liegenden Speichervolume abgerufen werden.

Jede Neptune-DB-Instance verfügt über eine Anzahl von Abfrage-Threads, die dem 2-Fachen der Zahl der vCPUs auf dieser Instance entspricht. Beispielsweise verfügbt eine `r5.4xlarge`-Instance mit 16 vCPUs über 32 Abfrage-Threads und kann daher 32 Abfragen gleichzeitig verarbeiten.

Zusätzliche Abfragen, die eingehen, wenn alle Abfrage-Threads belegt sind, werden in eine serverseitige Warteschlange eingereiht und nach dem FIFO-Prinzip verarbeitet, sobald Abfrage-Threads verfügbar werden. Diese serverseitige Warteschlange kann ungefähr 8 000 ausstehende Anfragen aufnehmen. Sobald sie voll ist, reagiert Neptune auf weitere Anfragen mit einer `ThrottlingException`. Sie können die Anzahl der ausstehenden Anfragen anhand der `MainRequestQueuePendingRequests` CloudWatch Metrik oder mithilfe des Status-Endpunkts der Gremlin-Abfrage mit dem Parameter überwachen. `includeWaiting`

Aus Client-Sicht umfasst die Ausführungszeit von Abfragen die gesamte Zeit, die diese in der Warteschlange verbringen, sowie die Zeit, die für die tatsächliche Ausführung der Abfrage benötigt wird.

Eine anhaltende gleichzeitige Schreiblast, die alle Abfrage-Threads auf der primären DB-Instance nutzt, weist idealerweise eine CPU-Auslastung von mindestens 90 % auf. Dies zeigt an, dass alle Abfrage-Threads auf dem Server aktiv sinnvolle Arbeit leisten. Die tatsächliche CPU-Auslastung ist jedoch häufig etwas niedriger, selbst bei anhaltender gleichzeitiger Schreiblast. Dies liegt in der Regel daran, dass Abfrage-Threads auf den Abschluss von I/O Vorgängen auf dem zugrunde liegenden Speichervolume warten. Neptune verwendet Quorum-Schreibvorgänge, bei denen sechs Kopien Ihrer Daten in drei Availability Zones erstellt werden. Vier dieser sechs Speicherknoten müssen einen Schreibvorgang bestätigen, damit dieser als dauerhaft angesehen wird. Wenn ein Abfrage-Thread auf dieses Quorum vom Speichervolume wartet, ist er blockiert, was die CPU-Auslastung reduziert.

Wenn Sie eine serielle Schreiblast ausführen, bei der Sie einen Schreibvorgang nach dem anderen ausführen und warten, bis der erste Schreibvorgang abgeschlossen ist, bevor Sie mit dem nächsten beginnen, können Sie davon ausgehen, dass die CPU-Auslastung noch geringer sein wird. Die genaue Menge hängt von der Anzahl der vCPUs und Abfragethreads ab (je mehr Abfrage-Threads, desto weniger CPU-Gesamtanzahl pro Abfrage), wobei eine gewisse Reduzierung durch das Warten verursacht wird. I/O

Weitere Informationen zur optimalen Dimensionierung von DB-Instances finden Sie unter Instance-Typen für Amazon Neptune auswählen. Die Preise für die einzelnen Instance-Typen finden Sie auf der Seite für Neptune-Preise.

Überwachen der DB-Instance-Leistung in Neptune

Sie können CloudWatch Metriken in Neptune verwenden, um die Leistung Ihrer DB-Instances zu überwachen und die vom Client beobachtete Abfragelatenz zu verfolgen. Siehe Wird CloudWatch zur Überwachung der DB-Instance-Leistung in Neptune verwendet.

Speicher, Zuverlässigkeit und Verfügbarkeit von Amazon Neptune

Amazon Neptune verwendet eine verteilte und gemeinsam genutzte Speicherarchitektur, die automatisch entsprechend Ihren Anforderungen an den Datenbankspeicher skaliert wird.

Neptune-Daten werden in einem Cluster-Volume gespeichert, bei dem es sich um ein einzelnes virtuelles Volume handelt, das Non-Volatile Memory Express (NVMe) -Laufwerke verwendet. SSD-based Der Cluster-Volume besteht aus einer Sammlung logischer Blöcke, die als „Segmente“ bezeichnet werden. Jedem dieser Segmente sind 10 Gigabyte (GB) Speicher zugewiesen. Die Daten in jedem Segment werden in sechs Kopien repliziert, die dann auf drei Availability Zones (AZs) in der AWS -Region verteilt werden, in der sich das DB-Cluster befindet.

Wenn ein Neptune-DB-Cluster erstellt wird, wird ihm ein einzelnes Segment mit 10 GB zugeteilt. Wenn die Datenmenge zunimmt und den aktuell zugeteilten Speicherplatz überschreitet, erweitert Neptune das Cluster-Volume automatisch, indem neue Segmente hinzugefügt werden. Ein Neptun-Cluster-Volume kann in allen unterstützten Regionen mit Ausnahme von China auf eine maximale Größe von 128 Tebibyte (TiB) anwachsen GovCloud, wo es auf 64 TiB begrenzt ist. Für Engine-Versionen vor Release: 1.0.2.1.R6 (22.04.2020) ist die Größe der Cluster-Volumes jedoch in allen Regionen auf 64 TiB begrenzt.

Das DB-Cluster-Volume enthält alle Benutzerdaten, Indizes und Verzeichnisse (wie im Abschnitt Neptune-Diagrammdatenmodell beschrieben) sowie interne Metadaten, z. B. interne Transaktionsprotokolle. Diese Diagrammdaten, einschließlich Indizes und interner Protokolle, dürfen die maximale Größe des Cluster-Volumes nicht überschreiten.

I/O— Optimierte Speicheroption

Neptune bietet zwei Preismodelle für Speicher an:

- Standardspeicher — Standardspeicher bietet kostengünstigen Datenbankspeicher für Anwendungen mit mäßiger bis geringer I/O Nutzung.
- I/O—Optimierter Speicher — Mit I/O —Optimized Storage zahlen Sie nur für den Speicher, den Sie tatsächlich nutzen. Die Kosten sind höher als bei Standardspeicher, und Sie zahlen nichts für den I/O tatsächlich genutzten Speicher.

  I/O—Optimierter Speicher wurde entwickelt, um die Anforderungen von I/O — intensiven Graph-Workloads zu vorhersehbaren Kosten, geringer I/O Latenz und gleichbleibendem Durchsatz zu erfüllen. I/O

Weitere Informationen finden Sie unter —Optimierter Speicher. I/O

Neptune-Speicherzuteilung

Auch wenn ein Neptune-Cluster-Volume auf bis zu 128 TiB (oder 64 TiB in einigen wenigen Regionen) vergrößert werden kann, werden Ihnen nur die Kosten für den tatsächlich zugeteilten Speicherplatz berechnet. Der gesamte zugeteilte Speicherplatz wird durch die Speicherhöchstgrenze festgelegt. Dies ist der maximale Speicherplatz, der dem Cluster-Volume zu einem beliebigen Zeitpunkt während seines Bestehens zugeteilt wird.

Das bedeutet, dass selbst dann, wenn Benutzerdaten aus einem Cluster-Volume entfernt werden, z. B. durch eine Drop-Abfrage wie `g.V().drop()`, der gesamte zugeteilte Speicherplatz unverändert bleibt. Neptune optimiert automatisch den ungenutzten zugeteilten Speicherplatz für die zukünftige Wiederverwendung.

Zusätzlich zu den Benutzerdaten verbrauchen zwei weitere Inhaltstypen internen Speicherplatz: Verzeichnisdaten und interne Transaktionsprotokolle. Verzeichnisdaten werden zwar zusammen mit Diagrammdaten gespeichert, bleiben jedoch unbegrenzte Zeit bestehen, auch wenn die unterstützten Diagrammdaten gelöscht wurden. Daher können Einträge wiederverwendet werden, wenn erneut Daten eingeführt werden. Interne Protokolldaten werden in einem getrennten internen Speicherbereich gespeichert, der über einen eigenen Höchstwert verfügt. Wenn ein internes Protokoll abläuft, kann der belegte Speicherplatz für andere Protokolle wiederverwendet werden, jedoch nicht für Diagrammdaten. Die Menge an internem Speicherplatz, der für Protokolle zugewiesen wurde, ist im Gesamtspeicher enthalten, der von der `VolumeBytesUsed` CloudWatch Metrik gemeldet wird.

In Bewährte Methoden für Speicher finden Sie Möglichkeiten, den zugeteilten Speicherplatz auf ein Minimum zu beschränken und Speicherplatz wiederzuverwenden.

Neptune-Speicher-Fakturierung

Die Speicherkosten werden auf Grundlage des Speicherhöchstwerts abgerechnet, wie oben beschrieben. Ihre Daten werden zwar in sechs Kopien repliziert, Ihnen wird jedoch nur eine Kopie der Daten berechnet.

Sie können den aktuellen Speichergrenzwert Ihres DB-Clusters ermitteln, indem Sie die `VolumeBytesUsed` CloudWatch Metrik überwachen (siehe Überwachung von Neptune mit Amazon CloudWatch).

Andere Faktoren, die sich auf die Neptune-Speicherkosten auswirken können, sind Datenbank-Snapshots und Backups, die getrennt als Backup-Speicher berechnet werden und auf den Neptune-Speicherkosten basieren (siehe CloudWatch Metriken, die für die Verwaltung des Neptune-Backup-Speichers nützlich sind).

Wenn Sie jedoch einen Klon Ihrer Datenbank erstellen, verweist der Klon auf dasselbe Cluster-Volume, das Ihr DB-Cluster selbst verwendet, sodass keine zusätzlichen Speichergebühren für die Originaldaten anfallen. Nachträgliche Änderungen am Klon verwenden das Copy-On-Write-Protokoll und führen zu zusätzlichen Speicherkosten.

Weitere Informationen zu Neptune-Preisen finden Sie unter Amazon-Neptune-Preise.

Bewährte Methoden für Neptune-Speicher

Da bestimmte Datentypen permanenten Speicherplatz in Neptune verbrauchen, sollten Sie diese bewährten Methoden anwenden, um große Spitzen beim Speicherwachstum zu vermeiden:

- Vermeiden Sie bei der Entwicklung Ihres Diagrammdatenmodells temporäre Eigenschaftsschlüssel und benutzerorientierte Werte, soweit möglich.
- Wenn Sie Ihr Datenmodell ändern möchten, dürfen Sie Daten erst dann in einen vorhandenen DB-Cluster unter Verwendung des neuen Modells laden, bis Sie die Daten in diesem DB-Cluster mithilfe der Fast-Reset-API gelöscht haben. Häufig ist es am besten, Daten, die ein neues Modell verwenden, in einen neuen DB-Cluster zu laden.
- Transaktionen für große Datenmengen erzeugen entsprechend umfangreiche interne Protokolle, wodurch die Höchstgrenze für den internen Protokollbereich dauerhaft erhöht werden kann. Beispielsweise könnte eine einzelne Transaktion, die alle Daten in Ihrem DB-Cluster löscht, ein riesiges internes Protokoll erzeugen, das die Zuteilung eines großen Teils des internen Speichers erfordern und damit den für Diagrammdaten verfügbaren Speicherplatz dauerhaft reduzieren würde.

  Um dies zu vermeiden, sollten Sie große Transaktionen in kleinere Transaktionen aufteilen und genügend Zeit zwischen ihnen einplanen, sodass die entsprechenden internen Protokolle ablaufen können und ihr interner Speicher für die Wiederverwendung durch nachfolgende Protokolle freigegeben wird.
- Um das Wachstum Ihres Neptun-Clustervolumens zu überwachen, können Sie einen CloudWatch Alarm für die `VolumeBytesUsed` CloudWatch Metrik einstellen. Dies kann besonders nützlich sein, wenn Ihre Daten die maximale Größe des Cluster-Volumes erreichen. Weitere Informationen finden Sie unter CloudWatch Amazon-Alarme verwenden.

Die einzige Möglichkeit, den von Ihrem DB-Cluster verwendeten Speicherplatz zu verringern, wenn Sie über eine große Menge an ungenutztem zugeteiltem Speicherplatz verfügen, besteht darin, alle Daten in Ihrem Diagramm zu exportieren und sie dann erneut in einen neuen DB-Cluster zu laden. Eine einfache Möglichkeit, Daten aus einem DB-Cluster zu exportieren, finden Sie unter Neptune-Service und -Hilfsprogramm für den Datenexport. Eine einfache Möglichkeit, Daten wieder in Neptune zu importieren, finden Sie unter Neptune-Bulk-Loader.

> **Note**
>
> Die Erstellung und Wiederherstellung eines Snapshots reduziert nicht die Menge des Speichers, der Ihrem DB-Cluster zugeteilt ist, da ein Snapshot das ursprüngliche Image des dem Cluster zugrunde liegenden Speichers beibehält. Wenn eine große Menge des zugeteilten Speichers nicht verwendet wird, können Sie den zugeteilten Speicher nur reduzieren, indem Sie die Diagrammdaten exportieren und in einen neuen DB-Cluster laden.

Speicherzuverlässigkeit und hohe Verfügbarkeit von Neptune

Amazon Neptune ist auf Zuverlässigkeit, Beständigkeit und Fehlertoleranz ausgelegt.

Da sechs Kopien Ihrer Neptune-Daten in drei Availability Zones (AZs) aufbewahrt werden, werden die Daten hoch dauerhaft gespeichert und die Wahrscheinlichkeit eines Datenverlusts ist sehr gering. Die Daten werden automatisch über die Availability Zones repliziert, unabhängig davon, ob es in ihnen DB-Instances gibt. Der Umfang der Replikation ist unabhängig von der Anzahl der DB-Instances in Ihrem Cluster.

Das bedeutet, dass Sie eine Read-Replica schnell hinzufügen können, da Neptune keine neue Kopie der Diagrammdaten erstellt. Stattdessen wird die Read-Replica mit dem Cluster-Volume verbunden, das Ihre Daten bereits enthält. Ebenso werden beim Entfernen einer Read-Replica die zugrunde liegenden Daten nicht entfernt.

Sie können das Cluster-Volume und dessen Daten erst löschen, wenn Sie alle zugehörigen DB-Instances gelöscht haben.

Neptune erkennt automatisch Fehler in den Segmenten, aus denen das Cluster-Volume besteht. Wenn eine Kopie der Daten in einem Segment beschädigt ist, repariert Neptune dieses Segment sofort und verwendet andere Kopien der Daten innerhalb desselben Segments, um sicherzustellen, dass die reparierten Daten aktuell sind. Auf diese Weise vermeidet Neptune Datenverluste

und reduziert die Notwendigkeit für zeitpunktbezogene Wiederherstellungen nach einem Laufwerksausfall.

Verbinden mit Amazo-Neptune-Endpunkten

Amazon Neptune verwendet einen Cluster von DB-Instances anstelle einer einzelnen Instance. Jede Neptune-Verbindung wird von einer spezifischen DB-Instance verarbeitet. Wenn Sie eine Verbindung zu einem Neptune-Cluster herstellen, verweisen der von Ihnen angegebene Hostname und Port auf einen zwischengeschalteten Handler, der als Endpunkt bezeichnet wird. Ein Endpunkt ist eine URL, die eine Host-Adresse und einen Port enthält. Neptune-Endpunkte verwenden verschlüsselte Transport Layer Security/Secure Sockets Layer () TLS/SSL -Verbindungen.

Neptune verwendet den Endpunkt-Mechanismus, um diese Verbindungen zu abstrahieren, sodass Sie die Hostnamen nicht fest kodieren oder eine eigene Logik schreiben müssen, um Verbindungen umzuleiten, wenn einige DB-Instances nicht verfügbar sind.

Ihrem Anwendungsfall entsprechend können Sie mit Endpunkten jede Verbindung der entsprechenden Instance oder Gruppe von Instances zuordnen. Mit benutzerdefinierten Endpunkten können Sie eine Verbindung zu Teilmengen von DB-Instances herstellen. Die folgenden Endpunkte sind in einem Neptune-DB-Cluster verfügbar:

Neptune-Cluster-Endpunkte

Ein Cluster-Endpunkt ist ein Endpunkt für einen Neptune-DB-Cluster, der eine Verbindung zur aktuellen primären DB-Instance für diesen DB-Cluster herstellt. Jeder Neptune-DB-Cluster verfügt über einen Cluster-Endpunkt und eine einzelne primäre DB-Instance.

Der Cluster-Endpunkt bietet Failover-Unterstützung für read/write Verbindungen zum DB-Cluster. Verwenden Sie den Cluster-Endpunkt für alle Schreibvorgänge auf dem DB-Cluster, darunter Einfügungs-, Aktualisierungs- und Löschvorgänge sowie DDL-Änderungen (Data Definition Language). Sie können den Cluster-Endpunkt auch für Lesevorgänge nutzen, beispielsweise Abfragen.

Wenn die aktuelle primäre DB-Instance eines DB-Clusters ausfällt, führt Neptune automatisch ein Failover zu einer neuen primären DB-Instance durch. Während eines Failovers bedient der DB-Cluster weiterhin Verbindungsanfragen von der neuen primären DB-Instance an den Cluster-Endpunkt mit minimaler Serviceunterbrechung.

Das folgende Beispiel zeigt einen Cluster-Endpunkt für einen Neptune-DB-Cluster.

```
mydbcluster.cluster-123456789012.us-east-1.neptune.amazonaws.com:8182
```

Neptune-Reader-Endpunkte

Ein Reader-Endpunkt ist ein Endpunkt für einen Neptune-DB-Cluster, der eine Verbindung zu einer verfügbaren Neptune-Replica für diesen DB-Cluster herstellt. Jeder Neptune-DB-Cluster verfügt über einen Reader-Endpunkt. Wenn es mehr als eine Neptune-Replica gibt, leitet der Reader-Endpunkt jede Verbindungsanforderung an eine der Neptune-Replicas weiter.

Der Reader-Endpunkt bietet Round-Robin-Routing für schreibgeschützte Verbindungen zum DB-Cluster. Verwenden Sie den Reader-Endpunkt für Lesevorgänge, beispielsweise -Abfragen.

Sie können den Reader-Endpunkt nur für Schreibvorgänge verwenden, wenn Sie über einen Einzel-Instance-Cluster (einen Cluster ohne Read Replicas) verfügen. Nur in diesem Fall kann der Reader sowohl für Schreib- als auch für Leseoperationen verwendet werden.

Das Round-Robin-Routing für den Reader-Endpunkt funktioniert durch Ändern des Hosts, auf den der DNS-Eintrag verweist. Mit jeder Auflösung des DNS erhalten Sie eine andere IP-Adresse und Verbindungen werden anhand dieser IP-Adressen geöffnet. Nachdem eine Verbindung hergestellt wurde, werden alle Anfragen nach dieser Verbindung an denselben Host gesendet. Der Client muss eine neue Verbindung herstellen und den DNS-Datensatz erneut auflösen, um eine Verbindung zu einem potenziell anderen Read Replica herzustellen.

> **Note**
>
> WebSockets Verbindungen werden oft über lange Zeiträume aufrechterhalten. Zum Anfordern verschiedener Read Replicas gehen Sie wie folgt vor:
>
> - Sicherstellen, dass Ihr Client den DNS-Eintrag jedes Mal auflöst, wenn er eine Verbindung herstellt.
> - Trennen Sie die Verbindung und stellen Sie sie weider her.

Verschiedene Client-Software löst den DNS möglicherweise unterschiedlich auf. Beispiel: Wenn Ihr Client den DNS auflöst und dann die IP-Adresse für jede Verbindung verwendet, werden alle Anfragen an einen einzelnen Host geleitet.

DNS-Caching für Clients oder Proxys löst den DNS-Namen auf denselben Endpunkt aus dem Cache auf. Dies ist ein Problem für Round-Robin-Routing- und Failover-Szenarien.

> **Note**
>
> Deaktivieren Sie alle DNS-Caching-Einstellungen, um die DNS-Auflösung jedes Mal zu erzwingen.

Der DB-Cluster verteilt Verbindungsanforderungen an den Reader-Endpunkt unter den verfügbaren Neptune-Replicas. Wenn der DB-Cluster lediglich eine primäre DB-Instance enthält, bedient der Reader-Endpunkt Verbindungsanforderungen über die primäre DB-Instance. Wenn eine Neptune-Replica für diesen DB-Cluster erstellt wird, bedient der Reader-Endpunkt bei nur minimaler Service-Unterbrechung Verbindungsanforderungen für den Reader-Endpunkt über die neue Neptune-Replica.

Das folgende Beispiel zeigt einen Reader-Endpunkt für einen Neptune-DB-Cluster.

```
mydbcluster.cluster-ro-123456789012.us-east-1.neptune.amazonaws.com:8182
```

Neptune-Instance-Endpunkte

Ein Instance-Endpunkt ist ein Endpunkt für eine DB-Instance in einem Neptune-DB-Cluster, der eine Verbindung zu dieser spezifischen DB-Instance herstellt. Jede DB-Instance in einem DB-Cluster hat unabhängig vom Instance-Typ einen eigenen eindeutigen Instance-Endpunkt. Daher ist ein Instance-Endpunkt für die aktuelle primäre DB-Instance des DB-Clusters vorhanden. Es ist auch ein einzelner Instance-Endpunkt für jede Neptune-Replica im DB-Cluster vorhanden.

Der Instance-Endpunkt ermöglicht die direkte Kontrolle über Verbindungen zum DB-Cluster in Szenarien, in denen die Verwendung des Cluster-Endpunkts oder des Reader-Endpunkts möglicherweise nicht geeignet ist. Beispiel: Ihre Client-Anwendung erfordert möglicherweise einen detaillierten Lastenausgleich je nach Workload-Typ. In diesem Fall können Sie mehrere Clients so konfigurieren, dass sie Verbindungen zu verschiedenen Neptune-Replicas in einem DB-Cluster herstellen, um die Lese-Workloads zu verteilen.

Das folgende Beispiel zeigt einen Instance-Endpunkt für eine DB-Instance in einem Neptune-DB-Cluster.

```
mydbinstance.123456789012.us-east-1.neptune.amazonaws.com:8182
```

Benutzerdefinierte Neptune-Endpunkte

Ein benutzerdefinierter Endpunkt für einen Neptune-Cluster ist ein von Ihnen gewählter Satz von DB-Instances. Wenn Sie eine Verbindung zum Endpunkt herstellen, wählt Neptune eine Instance in der

Gruppe aus, um die Verbindung zu ermöglichen. Sie bestimmen, auf welche Instances sich dieser Endpunkt bezieht, und Sie entscheiden auch, welchen Zweck der Endpunkt erfüllen soll.

Ein Neptune-DB-Cluster verfügt erst dann über benutzerdefinierte Endpunkte, wenn Sie diese erstellen. Sie können für jeden bereitgestellten Neptune-Cluster bis zu fünf benutzerdefinierte Endpunkte erstellen.

Der benutzerdefinierte Endpunkt bietet Datenbankverbindungen mit Lastenausgleich, die auf anderen Kriterien als dem Schreibschutz oder der read/write Fähigkeit der DB-Instances basieren. Da die Verbindung mit jeder DB-Instance erfolgen kann, die mit dem Endpunkt verknüpft ist, müssen alle Instances innerhalb dieser Gruppe dieselben Leistungs- und Speicherkapazitätsmerkmale besitzen. Wenn Sie benutzerdefinierte Endpunkten verwenden, wird der Reader-Endpunkt für diesen Cluster in der Regel nicht verwendet.

Dieses Feature ist für fortgeschrittene Benutzer mit spezialisierten Workloads vorgesehen, bei denen es nicht sinnvoll ist, wenn alle Neptune-Replicas im Cluster identisch sind. Mit benutzerdefinierten Endpunkten können Sie die Kapazität der DB-Instances vorhersagen, die für die einzelnen Verbindungen verwendet werden.

Wenn Sie beispielsweise mehrere benutzerdefinierte Endpunkte definieren, die eine Verbindung zu Gruppen von Instances mit unterschiedlichen Instance-Klassen herstellen, können Sie Benutzer mit unterschiedlichen Leistungsanforderungen zu den Endpunkten leiten, die für ihre Anwendungsfälle jeweils am besten geeignet sind.

Das folgende Beispiel zeigt einen benutzerdefinierten Endpunkt für eine DB-Instance in einem Neptune-DB-Cluster.

```
myendpoint.cluster-custom-123456789012.us-east-1.neptune.amazonaws.com:8182
```

Weitere Informationen finden Sie unter Arbeiten mit benutzerdefinierten Endpunkten.

Überlegungen zu Neptune-Endpunkten

Berücksichtigen Sie bei der Arbeit mit Neptune-Endpunkten Folgendes:

- Bevor Sie einen Instance-Endpunkt verwenden, um eine Verbindung zu einer bestimmten DB-Instance in einem DB-Cluster herzustellen, sollten Sie die Verwendung des Cluster-Endpunkts oder des Leser-Endpunkts für den DB-Cluster in Betracht ziehen.

  Cluster-Endpunkt und Reader-Endpunkt bieten Unterstützung für Hochverfügbarkeitsszenarien. Wenn die primäre DB-Instance eines DB-Clusters ausfällt, führt Neptune automatisch

einen Failover zu einer neuen primären DB-Instance durch. Dies geschieht entweder durch Heraufstufung einer vorhandenen Neptune-Replica zu einer neuen primären DB-Instance oder durch Erstellung einer neuen primären DB-Instance. Bei einem Failover können Sie den Cluster-Endpunkt verwenden, um die Verbindung zu der neu heraufgestuften oder erstellten primären DB-Instance wiederherzustellen, oder den Reader-Endpunkt verwenden, um eine Verbindung zu einer anderen Neptune-Replica im DB-Cluster wiederherzustellen.

Wenn Sie diesen Ansatz nicht verwenden, können Sie trotzdem sicherstellen, dass Sie sich mit der richtigen DB-Instance im DB-Cluster für die vorgesehene Operation verbinden. Dazu können Sie manuell oder programmatisch den Satz an verfügbaren DB-Instances im DB-Cluster ermitteln und deren Instance-Typen nach dem Failover bestätigen, bevor Sie den Instance-Endpunkt einer bestimmten DB-Instance verwenden.

Weitere Informationen zu Failovers finden Sie unter Fehlertoleranz für einen Neptune-DB-Cluster.

- Der Reader-Endpunkt leitet Verbindungen nur an verfügbare Neptune-Replicas in einem Neptune-DB-Cluster weiter. Es werden keine spezifischen Abfragen weitergeleitet.

  > **Important**
  >
  > Neptune führt keinen Lastausgleich durch.

  Wenn Sie einen Lastausgleich für Abfragen zur Verteilung des Lese-Workloads für einen DB-Cluster ausführen möchten, müssen Sie dies in Ihrer Anwendung durchführen. Sie müssen Instance-Endpunkte verwenden, um eine direkte Verbindung mit Neptune-Replicas zum Lastausgleich herzustellen.

- Das Round-Robin-Routing für den Reader-Endpunkt funktioniert durch Ändern des Hosts, auf den der DNS-Eintrag verweist. Der Client muss eine neue Verbindung herstellen und den DNS-Datensatz erneut auflösen, um eine Verbindung zu einer potenziell neuen Read Replica herzustellen.

- Während eines Failovers könnte der Reader-Endpunkt für kurze Zeit Verbindungen an die neue primäre DB-Instance eines DB-Clusters leiten, wenn eine Neptune-Replica zur neuen primären DB-Instance heraufgestuft wird.

Arbeiten mit benutzerdefinierten Endpunkten in Neptune

Wenn Sie eine DB-Instance zu einem benutzerdefinierten Endpunkt hinzufügen oder aus einem benutzerdefinierten Endpunkt entfernen, bleiben alle vorhandenen Verbindungen zu dieser DB-Instance bestehen.

Sie können eine Liste mit DB-Instances definieren, die in einem benutzerdefinierten Endpunkt enthalten sein sollen (statische Liste) oder eine Liste mit DB-Instances, die von einem benutzerdefinierten Endpunkt ausgeschlossen werden sollen (Ausschlussliste). Sie können den inclusion/exclusion Mechanismus verwenden, um die DB-Instances in Gruppen zu unterteilen und sicherzustellen, dass die benutzerdefinierten Endpunkte alle DB-Instances im Cluster abdecken. Jeder benutzerdefinierte Endpunkt kann jeweils nur einen dieser Listentypen enthalten.

In der AWS-Managementkonsole wird die Auswahl durch das Kontrollkästchen future zu diesem Cluster hinzugefügte Instanzen anhängen dargestellt. Wenn Sie das Kontrollkästchen deaktivieren, verwendet der benutzerdefinierte Endpunkt eine statische Liste, die nur die im Dialogfeld angegebenen DB-Instances enthält. Wenn Sie das Kontrollkästchen aktivieren, verwendet der benutzerdefinierte Endpunkt eine Ausschlussliste. In diesem Fall repräsentiert der benutzerdefinierte Endpunkt alle DB-Instances im Cluster (einschließlich zukünftig hinzugefügter DB-Instances) außer den Instances, die im Dialogfeld nicht ausgewählt wurden.

Neptune ändert die in statischen Listen oder Ausschlusslisten angegebenen DB-Instances nicht, wenn die DB-Instances aufgrund eines Failovers oder einer Heraufstufung ihre Rollen zwischen primärer Instance und Neptune-Replica wechseln.

Sie können eine DB-Instance mit mehreren benutzerdefinierten Endpunkten verknüpfen. Angenommen, Sie fügen einem Cluster eine neue DB-Instance hinzu. In diesen Fällen wird die DB-Instance allen benutzerdefinierten Endpunkten hinzugefügt, für die sie infrage kommt. Die jeweils definierte statische Liste oder Ausschlussliste bestimmt, welche DB-Instance hinzugefügt werden kann.

Wenn der Endpunkt eine statische Liste von DB-Instances enthält, werden keine neu hinzugefügten Neptune-Replicas hinzugefügt. Wenn der Endpunkt eine Ausschlussliste enthält, werden neu hinzugefügte Neptune-Replicas jedoch hinzugefügt, sofern sie nicht in der Ausschlussliste aufgeführt sind.

Wenn eine Neptune-Replica nicht mehr verfügbar ist, bleibt sie weiterhin mit ihren benutzerdefinierten Endpunkten verknüpft. Dies gilt unabhängig davon, ob sie beschädigt ist, angehalten wurde, neu gestartet wird oder aus einem anderen Grund nicht verfügbar ist. Solange sie jedoch nicht verfügbar ist, können Sie über keinen Endpunkt eine Verbindung zu ihr herstellen.

Da neu erstellte Neptune-Cluster über keine benutzerdefinierten Endpunkte verfügen, müssen Sie diese selbst erstellen und verwalten. Dies gilt auch für Neptune-Cluster, die aus Snapshots wiederhergestellt werden, da benutzerdefinierte Endpunkte nicht im Snapshot enthalten sind. Sie müssen diese nach der Wiederherstellung erneut erstellen und neue Endpunktnamen auswählen, wenn sich der wiederhergestellte Cluster in der gleichen Region wie der ursprüngliche Cluster befindet.

Erstellen eines benutzerdefinierten Endpunkts

Sie verwalten benutzerdefinierte Endpunkte über die Neptune-Konsole. Navigieren Sie hierzu zur Detailseite für Ihren Neptune-Cluster und verwenden Sie die Steuerelemente im Abschnitt Benutzerdefinierte Endpunkte.

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die Amazon Neptune Neptune-Konsole unter. https://console.aws.amazon.com/neptune/home
2. Navigieren Sie zur Cluster-Detailseite.
3. Wählen Sie die `Create custom endpoint`-Aktion im Abschnitt Endpunkte aus.
4. Wählen Sie einen Namen für den benutzerdefinierten Endpunkt aus, der für Ihre Benutzer-ID und Region eindeutig ist. Der Name darf höchstens 63 Zeichen enthalten und muss das folgende Format besitzen:

   endpointName`.cluster-custom-`*customerDnsIdentifier*`.`*dnsSuffix*

   Da die Namen benutzerdefinierter Endpunkte den Namen Ihres Clusters nicht enthalten, müssen Sie diese Namen nicht ändern, wenn Sie einen Cluster umbenennen. Sie können den Namen eines benutzerdefinierten Endpunkts jedoch nicht für mehr als einen Cluster in derselben Region verwenden. Geben Sie jedem benutzerdefinierten Endpunkt einen Namen, der unter den Clustern, die zu Ihrer Benutzer-ID innerhalb einer bestimmten Region gehören, nur einmal vorkommt.
5. Um eine Liste von DB-Instances auszuwählen, die auch bei einer Erweiterung des Clusters gleich bleibt, lassen Sie das Kontrollkästchen Attach future instances added to this cluster (Zukünftige zu diesem Cluster hinzugefügte Instances anhängen) deaktiviert. Wenn Sie dieses Kontrollkästchen aktivieren, fügt der benutzerdefinierte Endpunkt dynamisch alle neuen Instances hinzu, die zum Cluster hinzugefügt werden.

Anzeigen benutzerdefinierter Endpunkte

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die Amazon Neptune Neptune-Konsole unter. https://console.aws.amazon.com/neptune/home
2. Navigieren Sie zur Cluster-Detailseite Ihres DB-Clusters.
3. Der Abschnitt Endpunkte enthält lediglich Informationen zu benutzerdefinierten Endpunkten. (Details zu integrierten Endpunkten finden Sie im Hauptabschnitt zu Details). Um die Details für einen spezifischen benutzerdefinierten Endpunkt anzuzeigen, wählen Sie dessen Namen aus. Anschließend wird die Detailseite für diesen Endpunkt geöffnet.

Bearbeiten eines benutzerdefinierten Endpunkts

Sie können die Eigenschaften eines benutzerdefinierten Endpunkts bearbeiten, um die mit ihm verknüpften DB-Instances zu ändern. Sie können auch zwischen einer statischen Liste und einer Ausschlussliste wechseln.

Sie können keinen benutzerdefinierten Endpunkt verwenden oder eine Verbindung zu diesem herstellen, wenn die Änderungen einer Bearbeitungsaktion noch verarbeitet werden. Es kann einige Minuten dauern, bis der Status des Endpunkts wieder in Verfügbar geändert wird und Sie erneut eine Verbindung herstellen können.

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die Amazon Neptune Neptune-Konsole unter. https://console.aws.amazon.com/neptune/home
2. Navigieren Sie zur Cluster-Detailseite.
3. Wählen Sie im Abschnitt Endpunkte den Namen des benutzerdefinierten Endpunkts aus, den Sie bearbeiten möchten.
4. Wählen Sie auf der Detailseite für diesen Endpunkt die Aktion Bearbeiten aus.

Löschen eines benutzerdefinierten Endpunkts

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die Amazon Neptune Neptune-Konsole unter. https://console.aws.amazon.com/neptune/home
2. Navigieren Sie zur Cluster-Detailseite.
3. Wählen Sie im Abschnitt Endpunkte den Namen des benutzerdefinierten Endpunkts aus, den Sie löschen möchten.
4. Wählen Sie auf der Detailseite für diesen Endpunkt die Aktion Löschen aus.

Neptune-Labor-Modus

Sie können den Labor-Modus von Amazon Neptune verwenden, um neue Features zu aktivieren, die in der aktuellen Neptune-Engine-Version enthalten sind, aber noch nicht für die Produktion bereit sind und daher standardmäßig nicht aktiviert sind. Auf diese Weise können Sie diese Features in Ihren Entwicklungs- und Testumgebungen ausprobieren.

Verwenden des Neptune-Labor-Modus

Verwenden Sie den `neptune_lab_mode-DB-Cluster-Parameter`, um Features zu aktivieren oder zu deaktivieren. Dazu schließen Sie *`(feature name)`*`=enabled` oder *`(feature name)`*`=disabled` in den Wert des `neptune_lab_mode`-Parameters in der DB-Cluster-Parametergruppe ein.

In dieser Engine-Version können Sie beispielsweise den `neptune_lab_mode`-Parameter auf `Streams=disabled, ReadWriteConflictDetection=enabled` festlegen.

Weitere Informationen zum Bearbeiten der DB-Cluster-Parametergruppe für Ihre Datenbank finden Sie unter Bearbeiten einer Parametergruppe. Beachten Sie, dass Sie die Standard-DB-Cluster-Parametergruppe nicht bearbeiten können. Wenn Sie die Standardgruppe verwenden, müssen Sie eine neue DB-Cluster-Parametergruppe erstellen, bevor Sie den `neptune_lab_mode`-Parameter festlegen können.

> **Note**
>
> Wenn Sie einen statischen DB-Cluster-Parameter wie `neptune_lab_mode` ändern, müssen Sie die primäre (Writer-)Instance des Clusters neu starten, damit die Änderung wirksam wird. Vor Release: 1.2.0.0 (21.07.2022) wurden alle Read-Replicas in einem DB-Cluster automatisch neu gestartet, wenn die primäre Instance neu gestartet wurde.
> Ab Release: 1.2.0.0 (21.07.2022) führt ein Neustart der primären Instance nicht dazu, dass Replicas neu gestartet werden. Das bedeutet, dass Sie jede Instance getrennt neu starten müssen, um die Änderung eines DB-Cluster-Parameters zu übernehmen (siehe Parametergruppen).

> **Important**
>
> Wenn Sie die falschen Labor-Modus-Parameter angeben oder Ihre Anfrage aus einem anderen Grund fehlschlägt, werden Sie zurzeit möglicherweise nicht über den Fehler

informiert. Sie sollten stets überprüfen, ob eine Änderungsanforderung im Labor-Modus erfolgreich war, indem Sie die Status-API aufrufen, wie unten gezeigt:

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

Die Statusergebnisse enthalten Labor-Modus-Informationen, aus denen hervorgeht, ob die von Ihnen angeforderten Änderungen ausgeführt wurden oder nicht:

```
{
  "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"}
}
```

Sie können im Labor-Modus auf die folgenden Features zugreifen:

Der OSGP-Index

Neptune kann nun einen vierten Index verwalten, den OSGP-Index, der für Datensätze mit einer großen Anzahl von Prädikaten nützlich ist (siehe Aktivieren eines OSGP-Indexes).

Sie können einen OSGP-Index in einem neuen, leeren Neptune-DB-Cluster aktivieren, indem Sie im `neptune_lab_mode`-DB-Cluster-Parameter `ObjectIndex=enabled` festlegen. Ein OSGP-Index kann nur in einem neuen, leeren DB-Cluster aktiviert werden.

Standardmäßig ist der OSPG-Index deaktiviert.

> **Note**
>
> Nachdem Sie den DB-Cluster-Parameter `neptune_lab_mode` für die Unterstützung des OSGP-Index eingerichtet haben, müssen Sie die Writer-Instance des Clusters neu starten, damit die Änderung wirksam wird.

> **Warning**
>
> Wenn Sie einen aktivierten OSGP-Index durch eine Einstellung deaktivieren `ObjectIndex=disabled` und ihn später wieder aktivieren, nachdem Sie weitere Daten hinzugefügt haben, wird der Index nicht korrekt erstellt. On-demand Die Neuerstellung des Indexes wird nicht unterstützt, daher sollten Sie den OSGP-Index nur aktivieren, wenn die Datenbank leer ist.

Garbage-Collection im Wörterbuch

Die Garbage-Collection aus dem Wörterbuch kann für Eigenschaftsdiagrammdaten aktiviert werden, wenn Neptune-Streams nicht über den Parameter aktiviert ist. `DictionaryGCMode` Die Parallelität kann über den Parameter gesteuert werden. `DictionaryGCConcurrency` Weitere Details finden Sie unter Müllabfuhr im Wörterbuch.

Formalisierte Transaktionssemantik

Neptune hat die formale Semantik für gleichzeitige Transaktionen aktualisiert (siehe Transaktionssemantik in Neptune).

Verwenden Sie `ReadWriteConflictDetection` als Namen in dem `neptune_lab_mode`-Parameter, der die formalisierte Transaktionssemantik aktiviert oder deaktiviert.

Standardmäßig ist die formalisierte Transaktionssemantik bereits aktiviert. Wenn Sie zum früheren Verhalten zurückkehren möchten, nehmen Sie `ReadWriteConflictDetection=disabled` in den Wert für den DB-Cluster-`neptune_lab_mode`-Parameter auf.

Erweiterte Datetime-Unterstützung

Neptune hat die Unterstützung für die Datetime-Funktionalität erweitert. Um Datetime mit erweiterten Formaten zu aktivieren, fügen Sie diese Option `DatetimeMillisecond=enabled` in den Wertesatz für den DB-Cluster-Parameter ein. `neptune_lab_mode`

StrictTimeoutValidation

> **Note**
>
> Diese Funktion ist ab Version 1.3.2.0 der Neptune Engine verfügbar.

Standardwert: aktiviert (vor der Neptune-Engine-Version 1.4.0.0 standardmäßig deaktiviert)

Zulässige Werte: enabled/disabled

Wenn dieser Parameter aktiviert ist`enabled`, darf ein Timeout-Wert pro Abfrage, der als Anforderungsoption oder Abfragehinweis angegeben wurde, den global in der `neptune_query_timeout`Parametergruppeneinstellung festgelegten Wert nicht überschreiten. Wenn das Timeout pro Abfrage die globale Einstellung überschreitet, gibt Neptune eine aus. `InvalidParameterException` In Engine-Versionen vor 1.4.0.0 war dieser Parameter `disabled` standardmäßig und musste explizit aktiviert werden.

Diese Einstellung kann in einer Antwort auf dem `/status` Endpunkt bestätigt werden, wenn der Wert lautet. `disabled`

Weitere Informationen finden Sie unter Zeitüberschreitungen pro Abfrage.

AccurateQRCMemoryEstimation

> **Note**
>
> Diese Funktion ist ab Version 1.4.0.0 der Neptune-Engine verfügbar.

Standardwert: deaktiviert

Zulässige Werte: enabled/disabled

Der Gremlin-Abfrageergebnis-Cache ermöglicht, wenn er aktiviert ist, das Zwischenspeichern von Abfrageergebnissen in der Datenbank. Standardmäßig wird die ungefähre Schätzung verwendet, um die Größe des zwischengespeicherten Ergebnisses zu bestimmen. Wenn dieser Parameter im Labormodus `AccurateQRCMemoryEstimation` aktiviert ist, verwendet die Größenschätzung für zwischengespeicherte Ergebnisse genaue Größenschätzungen anstelle von Näherungswerten. Dieser Labmode-Parameter ist ab Neptune Engine Release-Version 1.4.0.0 verfügbar.

Die alternative Abfrage-Engine (DFE) von Amazon Neptune

Amazon Neptune verfügt über eine alternative Abfrage-Engine namens DFE, die DB-Instance-Ressourcen wie CPU-Kerne und Speicher I/O effizienter als die ursprüngliche Neptune-Engine nutzt.

> **Note**
>
> Bei großen Datensätzen wird die DFE-Engine auf T3-Instances möglicherweise nicht einwandfrei ausgeführt.

Die DFE-Engine führt SPARQL-, Gremlin- und openCypher-Abfragen aus und unterstützt eine Vielzahl von Plantypen, darunter Left-Deep-, Bushy- und Hybrid-Tarife. Planbetreiber können sowohl Rechenoperationen aufrufen, die auf einer reservierten Gruppe von Rechenkernen ausgeführt werden, als auch I/O Operationen, von denen jeder in einem eigenen Thread in einem Thread-Pool ausgeführt wird. I/O

Die DFE-Engine verwendet vorgenerierte Statistiken zu Ihren Neptune-Diagrammdaten, um informierte Entscheidungen zur Strukturierung von Abfragen zu treffen. Informationen dazu, wie diese Statistiken generiert werden, finden Sie unter DFE-Statistiken.

Die Wahl des Plantyps und der Anzahl der verwendeten Rechen-Threads erfolgt automatisch auf der Grundlage vorgenerierter Statistiken und der Ressourcen, die im Neptune-Hauptknoten verfügbar sind. Die Reihenfolge der Ergebnisse ist für Pläne mit interner Rechenparallelität nicht vorgegeben.

Steuern der Verwendung der Neptune-DFE-Engine

Standardmäßig ist der Instance-Parameter neptune_dfe_query_engine einer Instance auf `viaQueryHint` festgelegt. Dies führt dazu, dass die DFE-Engine nur für openCypher-Abfragen sowie für Gremlin- und SPARQL-Abfragen verwendet wird, die explizit den auf `true` festgelegten Abfragehinweis `useDFE` enthalten.

Sie können die DFE-Engine vollständig aktivieren, sodass sie verwendet wird, wann immer möglich, indem Sie den Instance-Parameter `neptune_dfe_query_engine` auf `enabled` festlegen.

Sie können die DFE-Engine auch deaktivieren, indem Sie den Abfragehinweis `useDFE` für eine bestimmte Gremlin-Abfrage oder SPARQL-Abfrage einfügen. Mit diesem Abfragehinweis können Sie verhindern, dass die DFE-Engine diese bestimmte Abfrage ausführt.

Sie können mithilfe eines Instance-Status-Aufrufs feststellen, ob die DFE-Engine in einer Instance aktiviert ist oder nicht:

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

Die Statusantwort gibt dann an, ob die DFE-Engine aktiviert ist oder nicht:

```
{
  "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"}
}
```

Der Gremlin-Ergebnisse für `explain` und `profile` geben an, ob eine Abfrage von der DFE-Engine ausgeführt wird. Siehe `Informationen, die in einem Gremlin Explain-Bericht enthalten sind` für `explain` und DFE-profile-Berichte für `profile`.

Ähnlich teilt SPARQL `explain` Ihnen mit, ob eine SPARQL-Abfrage von der DFE-Engine ausgeführt wird. Weitere Informationen finden Sie unter Beispiel für die SPARQL `Explain-Ausgabe, wenn` DFE aktiviert ist und `DefNode-Operator`.

Von der Neptune-DFE-Engine unterstützte Abfragekonstrukte

Zurzeit unterstützt die Neptune-DFE-Engine eine Teilmenge von SPARQL- und Gremlin-Abfragekonstrukten.

Für SPARQL ist dies die Teilmenge der konjunktiven Basisdiagrammmuster.

Für Gremlin ist dies im Allgemeinen die Teilmenge von Abfragen, die eine Kette von Traversierungen enthalten, die einige der komplexeren Schritte nicht enthalten.

Sie können wie folgt herausfinden, ob eine Ihrer Abfragen ganz oder teilweise von der DFE-Engine ausgeführt wird:

- In Gremlin teilen Ihnen die Ergebnisse für `explain` und `profile` mit, welche Teile einer Abfrage von der DFE-Engine ausgeführt werden, wenn zutreffend. Siehe `Informationen, die in einem Gremlin Explain-Bericht enthalten sind` für `explain` und DFE-profile-Berichte für `profile`. Weitere Informationen finden Sie auch in `Optimieren von Gremlin-Abfragen mithilfe von Explain und Profile`.

  Einzelheiten zur Neptune-Engine-Unterstützung für einzelne Gremlin-Schritte sind in Unterstützung für Gremlin-Schritte dokumentiert.
- Ähnlich teilt SPARQL `explain` Ihnen mit, ob eine SPARQL-Abfrage von der DFE-Engine ausgeführt wird. Weitere Informationen finden Sie unter Beispiel für die SPARQL `Explain`-Ausgabe, wenn DFE aktiviert ist und `DefNode-Operator`.

Verwalten von Statistiken, die die Neptune-DFE-Engine verwenden soll

> **Note**
>
> Die Unterstützung von openCypher ist von der DFE-Abfrage-Engine in Neptune abhängig. Die DFE-Engine wird mithilfe des neptune_dfe_query_engine Instance-Parameters in der DB-Parametergruppe einer Instanz gesteuert.

Die DFE-Engine verwendet Informationen zu den Daten in Ihrem Neptune-Diagramm, um bei der Planung der Abfrageausführung effektive Kompromisse einzugehen. Bei diesen Informationen handelt es sich um Statistiken, die sogenannte Merkmalssätze und Prädikatstatistiken umfassen, die als Anleitung für die Abfrageplanung dienen können.

Ab Engine-Version 1.2.1.0 können Sie mithilfe der GetGraphSummaryAPI oder des Endpunkts zusammenfassende Informationen zu Ihrem Diagramm aus diesen Statistiken abrufen. `summary`

Diese DFE-Statistiken werden derzeit neu generiert, wenn entweder mehr als 10 % der Daten in Ihrem Diagramm geändert wurden oder wenn die neuesten Statistiken älter als 10 Tage sind. Diese Auslöser können sich in der Zukunft jedoch ändern.

> **Note**
>
> Die Statistikgenerierung ist für `T3`- und `T4g`-Instances deaktiviert, da sie die Arbeitsspeicherkapazität dieser Instance-Typen überschreiten kann.

Sie können die Generierung von DFE-Statistiken über einen der folgenden Endpunkte verwalten:

- `https://`*`your-neptune-host`*`:`*`port`*`/rdf/statistics` (für SPARQL).
- `https://`*`your-neptune-host`*`:`*`port`*`/propertygraph/statistics` (für Gremlin und openCypher) und die alternative Version: `https://`*`your-neptune-host`*`:`*`port`*`/pg/statistics`.

> **Note**
>
> Ab Engine-Version 1.1.1.0 wird der Gremlin-Statistik-Endpunkt (`https://your-neptune-host:port/gremlin/statistics`) zugunsten des `propertygraph`- oder pg-Endpunkts außer Betrieb genommen. Aus Gründen der Abwärtskompatibilität wird er weiter unterstützt, wird jedoch in zukünftigen Versionen möglicherweise entfernt.
> Ab Engine-Version 1.2.1.0 wird der SPARQL-Statistik-Endpunkt (`https://your-neptune-host:port/sparql/statistics`) zugunsten des `rdf`-Endpunkts außer Betrieb genommen. Aus Gründen der Abwärtskompatibilität wird er weiter unterstützt, wird jedoch in zukünftigen Versionen möglicherweise entfernt.

> **Note**
>
> Wenn sich ein DFE-Statistikendpunkt auf einer Reader-Instance befindet, kann er nur Statusanfragen verarbeiten. Andere Anfragen schlagen mit einer `ReadOnlyViolationException` fehl.

Größenbeschränkungen für die DFE-Statistikgenerierung

Derzeit wird die Generierung von DFE-Statistiken angehalten, wenn eine der folgenden Größenbeschränkungen erreicht wird:

- Die Anzahl der generierten Merkmalsätze darf 50 000 nicht überschreiten.
- Die Anzahl der generierten Prädikatstatistiken darf eine Million nicht überschreiten.

Diese Grenzwerte unterliegen Änderungen.

Aktueller Status der DFE-Statistiken

Sie können den aktuellen Status der DFE-Statistiken wie folgt überprüfen:

> **Note**
>
> In den folgenden Beispielen werden der Endpunkt und die Befehle der Eigenschaft graph verwendet. Verwenden Sie für RDF-Daten `get-sparql-statistics` für die AWS CLI und `/rdf/statistics` für REST-Endpunkte.

AWS CLI

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

> **Note**
>
> Ersetze es *us-east-1* durch die Region deines Neptun-Clusters. Sie können weglassen, `--region` wenn Sie eine Standardregion konfiguriert haben.

Weitere Informationen finden Sie unter get-propertygraph-statistics in der Befehlsreferenz. 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**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

Weitere Informationen zur Verwendung awscurl mit der IAM-Authentifizierung finden Sie unter. Verwenden von `awscurl` mit temporären Anmeldeinformationen, um eine sichere Verbindung zu einem DB-Cluster mit aktivierter IAM-Authentifizierung herzustellen

curl

```
curl -G https://your-neptune-endpoint:port/propertygraph/statistics
```

Die Antwort auf eine Statusanforderung enthält die folgenden Felder:

- `status` – HTTP-Rückgabecode der Anforderung. Wenn die Anforderung erfolgreich ist, lautet der Code `200`. Eine Liste mit häufigen Fehlern finden Sie unter Häufige Fehler.
- `payload`:
  - `autoCompute` – (Boolean) Gibt an, ob die automatische Generierung von Statistiken aktiviert ist oder nicht.
  - `active` – (Boolean) Gibt an, ob die automatische Generierung von DFE-Statistiken aktiviert ist oder nicht.
  - `statisticsId` – Meldet die ID der aktuellen Statistikgenerierungsausführung. Der Wert `-1` gibt an, dass keine Statistiken generiert wurden.
  - `date` – Die UTC-Zeit, zu der die DFE-Statistiken zuletzt generiert wurden (im Format ISO 8601).

> **Note**
>
> Vor Engine-Version 1.2.1.0 wurde dies mit Minutengenauigkeit dargestellt. Ab Engine-Version 1.2.1.0 wird dies mit Millisekundengenauigkeit dargestellt (zum Beispiel `2023-01-24T00:47:43.319Z`).

- `note` – Ein Hinweis zu Problemen in dem Fall, dass Statistiken ungültig sind.
- `signatureInfo` – Enthält Informationen zu den in der Statistik generierten Merkmalsätzen (vor Engine-Version 1.2.1.0 hatte dieses Feld den Namen `summary`). Dies ist im Allgemeinen nicht direkt umsetzbar:
  - `signatureCount` – Gesamtzahl der Signaturen für alle Merkmalsätze.
  - `instanceCount` – Gesamtzahl der Merkmalsatz-Instances.
  - `predicateCount` – Gesamtzahl der eindeutigen Prädikate.

Die Antwort auf eine Statusanforderung, wenn keine Statistiken generiert wurden, sieht wie folgt aus:

```
{
  "status" : "200 OK",
  "payload" : {
    "autoCompute" : true,
    "active" : false,
    "statisticsId" : -1
   }
}
```

Wenn DFE-Statistiken verfügbar sind, sieht die Antwort wie folgt aus:

```
{
  "status" : "200 OK",
  "payload" : {
    "autoCompute" : true,
    "active" : true,
    "statisticsId" : 1588893232718,
    "date" : "2020-05-07T23:13Z",
    "summary" : {
      "signatureCount" : 5,
      "instanceCount" : 1000,
```

```
      "predicateCount" : 20
    }
  }
}
```

Wenn die Generierung von DFE-Statistiken fehlgeschlagen ist, weil beispielsweise die Größenbeschränkung für Statistiken überschritten wurde, sieht die Antwort wie folgt aus:

```
{
  "status" : "200 OK",
  "payload" : {
    "autoCompute" : true,
    "active" : false,
    "statisticsId" : 1588713528304,
    "date" : "2020-05-05T21:18Z",
    "note" : "Limit reached: Statistics are not available"
  }
}
```

Deaktivieren der automatischen Generierung von DFE-Statistiken

Standardmäßig ist die automatische Generierung von DFE-Statistiken aktiviert, wenn Sie DFE aktivieren.

Sie können die automatische Generierung wie folgt deaktivieren:

> **Note**
>
> In den folgenden Beispielen werden der Endpunkt und die Befehle der Eigenschaft Graph verwendet. Verwenden Sie für RDF-Daten `manage-sparql-statistics` für die AWS CLI und `/rdf/statistics` für REST-Endpunkte.

AWS CLI

```
aws neptunedata manage-propertygraph-statistics \
  --endpoint-url https://your-neptune-endpoint:port \
  --mode disableAutoCompute
```

Weitere Informationen finden Sie unter manage-propertygraph-statistics in der Befehlsreferenz. 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**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

Weitere Informationen zur Verwendung awscurl mit der IAM-Authentifizierung finden Sie unter. Verwenden von `awscurl` mit temporären Anmeldeinformationen, um eine sichere Verbindung zu einem DB-Cluster mit aktivierter IAM-Authentifizierung herzustellen

curl

```
curl -X POST -d '{ "mode" : "disableAutoCompute" }' https://your-neptune-
endpoint:port/propertygraph/statistics
```

Wenn die Anforderung erfolgreich ist, lautet der HTTP-Antwortcode `200` und die Antwort ist:

```
{
  "status" : "200 OK"
}
```

Sie können überprüfen, ob die automatische Generierung deaktiviert ist, indem Sie eine Statusanforderung ausgeben und überprüfen, ob das Feld `autoCompute` in der Antwort auf `false` festgelegt ist.

Durch das Deaktivieren der automatischen Generierung von Statistiken wird eine laufende Statistikberechnung nicht beendet.

Wenn Sie eine Anforderung zur Deaktivierung der automatischen Generierung an eine Reader-Instance Ihres DB-Clusters statt an die Writer-Instance senden, schlägt die Anforderung mit dem HTTP-Rückgabecode 400 und einer Ausgabe wie der folgenden fehl:

```
{
  "detailedMessage" : "Writes are not permitted on a read replica instance",
  "code" : "ReadOnlyViolationException",
  "requestId":"8eb8d3e5-0996-4a1b-616a-74e0ec32d5f7"
}
```

Eine Liste mit weiteren häufigen Fehlern finden Sie unter Häufige Fehler.

Re-enabling automatische Generierung von DFE-Statistiken

Standardmäßig ist die automatische Generierung von DFE-Statistiken bereits aktiviert, wenn Sie DFE aktivieren. Wenn Sie die automatische Generierung deaktivieren, können Sie diese später wie folgt erneut aktivieren:

> **Note**
>
> In den folgenden Beispielen werden der Endpunkt und die Befehle der Eigenschaft graph verwendet. Verwenden Sie für RDF-Daten `manage-sparql-statistics` für die AWS CLI und `/rdf/statistics` für REST-Endpunkte.

AWS CLI

```
aws neptunedata manage-propertygraph-statistics \
  --endpoint-url https://your-neptune-endpoint:port \
  --mode enableAutoCompute
```

Weitere Informationen finden Sie unter manage-propertygraph-statistics in der Befehlsreferenz. 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**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

Weitere Informationen zur Verwendung awscurl mit der IAM-Authentifizierung finden Sie unter. Verwenden von `awscurl` mit temporären Anmeldeinformationen, um eine sichere Verbindung zu einem DB-Cluster mit aktivierter IAM-Authentifizierung herzustellen

curl

```
curl -X POST -d '{ "mode" : "enableAutoCompute" }' https://your-neptune-endpoint:port/propertygraph/statistics
```

Wenn die Anforderung erfolgreich ist, lautet der HTTP-Antwortcode `200` und die Antwort ist:

```
{
  "status" : "200 OK"
}
```

Sie können überprüfen, ob die automatische Generierung aktiviert ist, indem Sie eine Statusanforderung ausgeben und prüfen, ob das Feld `autoCompute` in der Antwort auf `true` festgelegt ist.

Manuelles Auslösen der Generierung von DFE-Statistiken

Sie können die Generierung von DFE-Statistiken wie folgt manuell starten:

> **Note**
>
> In den folgenden Beispielen werden der Endpunkt und die Befehle der Eigenschaft Graph verwendet. Verwenden Sie für RDF-Daten `manage-sparql-statistics` für die AWS CLI und `/rdf/statistics` für REST-Endpunkte.

AWS CLI

```
aws neptunedata manage-propertygraph-statistics \
  --endpoint-url https://your-neptune-endpoint:port \
  --mode refresh
```

Weitere Informationen finden Sie unter manage-propertygraph-statistics in der Befehlsreferenz. 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**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

Weitere Informationen zur Verwendung awscurl mit der IAM-Authentifizierung finden Sie unter. Verwenden von awscurl mit temporären Anmeldeinformationen, um eine sichere Verbindung zu einem DB-Cluster mit aktivierter IAM-Authentifizierung herzustellen

curl

```
curl -X POST -d '{ "mode" : "refresh" }' https://your-neptune-endpoint:port/
propertygraph/statistics
```

Wenn die Anforderung erfolgreich ist, sieht die Ausgabe wie folgt aus und der HTTP-Rückgabecode ist 200:

```
{
  "status" : "200 OK",
  "payload" : {
    "statisticsId" : 1588893232718
  }
}
```

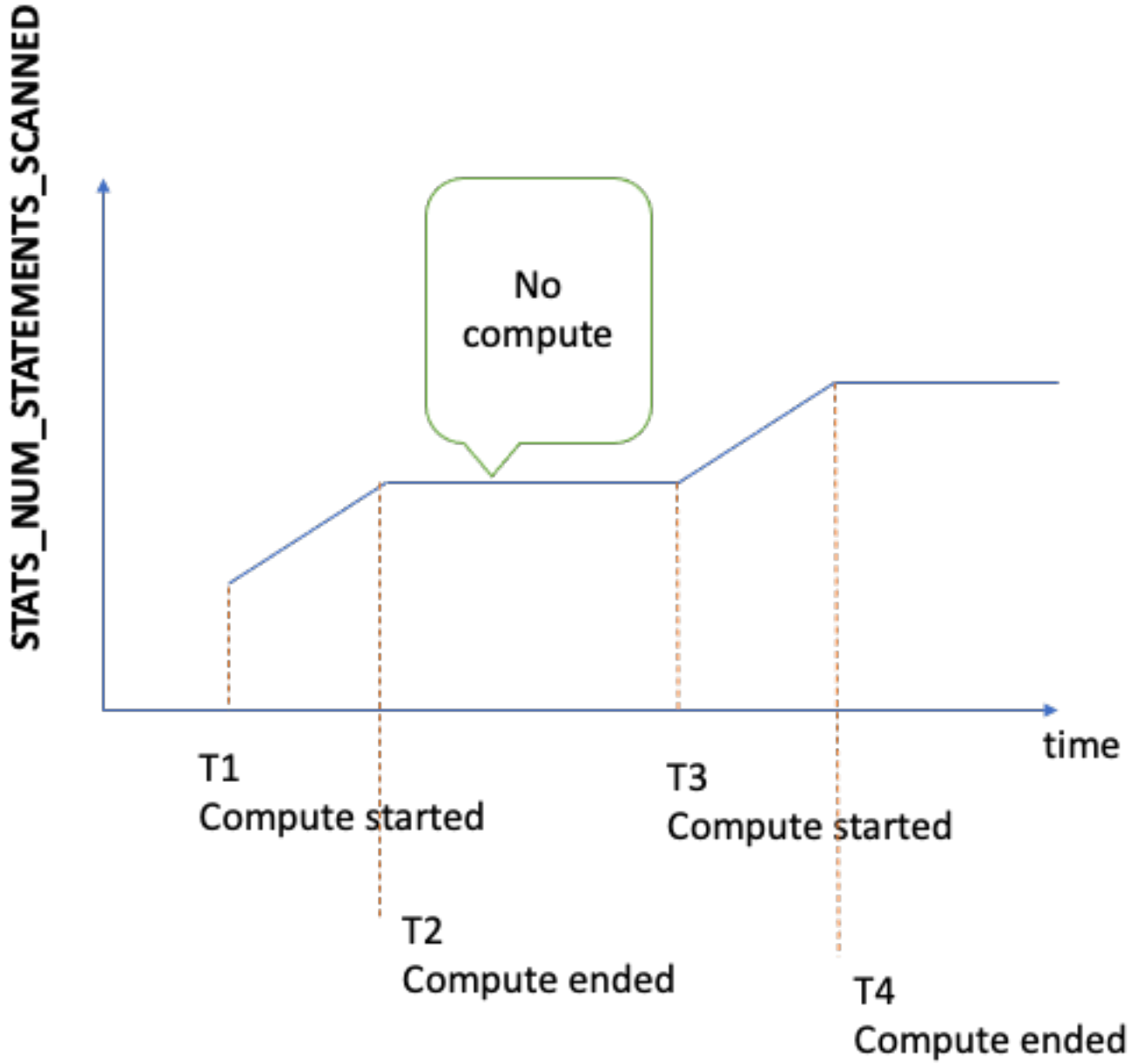
Die `statisticsId` in der Ausgabe ist die ID der Statistikgenerierung, die zurzeit ausgeführt wird. Wenn zum Zeitpunkt der Anforderung bereits eine Ausführung bearbeitet wird, gibt die Anforderung die ID dieser Ausführung zurück, statt eine neue zu initiieren. Sie können jeweils nur eine Statistikgenerierung gleichzeitig ausführen.

Wenn während der Generierung der DFE-Statistiken ein Failover auftritt, nimmt der neue Writer-Knoten den zuletzt verarbeiteten Prüfpunkt auf und setzt die Statistikausführung ab dort fort.

Verwendung der **`StatsNumStatementsScanned`** CloudWatch Metrik zur Überwachung der Statistikberechnung

Die `StatsNumStatementsScanned` CloudWatch Metrik gibt die Gesamtzahl der Anweisungen zurück, die seit dem Start des Servers für statistische Berechnungen gescannt wurden. Sie wird bei jedem Slice der Statistikberechnung aktualisiert.

Bei jeder Auslösung einer Statistikberechnung nimmt diese Zahl zu. Wenn keine Berechnung stattfindet, bleibt sie konstant. Wenn Sie ein Diagramm mit `StatsNumStatementsScanned`-Werten über die Zeit betrachten, erhalten Sie daher ein ziemlich klares Bild davon, wann und wie schnell Statistikberechnungen ausgeführt wurden:



Während der Berechnung zeigt Ihnen die Steigung im Diagramm, wie schnell diese ausgeführt wird. (Je steiler die Steigung, desto schneller werden Statistiken berechnet.)

Wenn das Diagramm einfach eine flache Linie bei 0 ist, wurde das Statistik-Feature aktiviert, es wurden jedoch keine Statistiken berechnet. Wenn das Statistik-Feature deaktiviert wurde oder wenn Sie eine Engine-Version verwenden, die keine Statistikberechnung unterstützt, ist `StatsNumStatementsScanned` nicht vorhanden.

Wie bereits erwähnt, können Sie die Statistikberechnung über die Statistik-API deaktivieren. Wenn Sie diese jedoch deaktiviert lassen, sind Statistiken möglicherweise nicht aktuell, was zu einer mangelhaften Abfrageplangenerierung für die DFE-Engine führen kann.

Informationen Überwachung von Neptune mit Amazon CloudWatch zur Verwendung finden Sie unter CloudWatch.

Verwenden AWS Identity and Access Management (IAM-) Authentifizierung mit DFE-Statistikendpunkten

Sie können mit der IAM-Authentifizierung sicher auf DFE-Statistikendpunkte zugreifen, indem Sie awscurl oder ein anderes Tool verwenden AWS CLI, das mit HTTPS und IAM funktioniert. Informationen zum Einrichten der richtigen Anmeldeinformationen finden Sie unter. Verwenden von `awscurl` mit temporären Anmeldeinformationen, um eine sichere Verbindung zu einem DB-Cluster mit aktivierter IAM-Authentifizierung herzustellen

Die folgenden Beispiele zeigen, wie eine authentifizierte Statusanfrage gestellt wird:

> **Note**
>
> In den folgenden Beispielen werden der Endpunkt und die Befehle des Eigenschaftsdiagramms verwendet. Verwenden Sie für RDF-Daten die entsprechenden SPARQL-Befehle (`get-sparql-statistics`,`manage-sparql-statistics`) für die AWS CLI und `/rdf/statistics` für REST-Endpunkte.

AWS CLI

```
aws neptunedata get-propertygraph-statistics \
  --endpoint-url https://your-neptune-endpoint:port
```

Der AWS CLI verarbeitet die IAM-Authentifizierung automatisch anhand Ihrer konfigurierten Anmeldeinformationen.

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))
```

Das SDK verarbeitet die IAM-Authentifizierung automatisch mithilfe Ihrer konfigurierten Anmeldeinformationen.

awscurl

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

> **Note**
>
> Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

curl

```
curl -G https://your-neptune-endpoint:port/propertygraph/statistics
```

> **Note**
>
> Dieses Beispiel beinhaltet keine IAM-Authentifizierung. Um die IAM-Authentifizierung mit zu verwendencurl, müssen Sie die Anfrage manuell signieren. Erwägen Sie AWS CLI, awscurl stattdessen das SDK oder zu verwenden.

In den folgenden Beispielen wird die Statistikgenerierung manuell mit Authentifizierung initiiert:

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

Erstellen Sie eine JSON-Datei mit dem Namen`request.json`, die Folgendes enthält:

```
{ "mode" : "refresh" }
```

Dann führe Folgendes aus:

```
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**
>
> Dieses Beispiel beinhaltet keine IAM-Authentifizierung. Um die IAM-Authentifizierung mit zu verwendencurl, müssen Sie die Anfrage manuell signieren. Erwägen Sie AWS CLI, awscurl stattdessen das SDK oder zu verwenden.

Löschen von DFE-Statistiken

Sie können alle Statistiken in der Datenbank wie folgt löschen:

> **Note**
>
> In den folgenden Beispielen werden der Endpunkt und die Befehle der Eigenschaft graph verwendet. Verwenden Sie für RDF-Daten `delete-sparql-statistics` für die AWS CLI und `/rdf/statistics` für REST-Endpunkte.

AWS CLI

```
aws neptunedata delete-propertygraph-statistics \
  --endpoint-url https://your-neptune-endpoint:port
```

Weitere Informationen finden Sie unter delete-propertygraph-statistics in der Befehlsreferenz. 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**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

Weitere Informationen zur Verwendung awscurl mit der IAM-Authentifizierung finden Sie unter. Verwenden von `awscurl` mit temporären Anmeldeinformationen, um eine sichere Verbindung zu einem DB-Cluster mit aktivierter IAM-Authentifizierung herzustellen

curl

```
curl -X DELETE https://your-neptune-endpoint:port/propertygraph/statistics
```

Gültige HTTP-Rückgabecodes sind:

- `200` – Die Statistiken wurden erfolgreich gelöscht.

  In diesem Fall würde eine typische Antwort wie folgt aussehen:

```
{
  "status" : "200 OK",
  "payload" : {
      "active" : false,
      "statisticsId" : -1
  }
}
```

- `204` – Es gab keine Statistiken, die gelöscht werden konnten.

In diesem Fall ist die Antwort leer (keine Antwort).

Wenn Sie eine Löschanforderung an einen Statistikendpunkt auf einem Reader-Knoten senden, wird eine `ReadOnlyViolationException` ausgelöst.

Häufige Fehlercodes für DFE-Statistikanforderungen

Im Folgenden finden Sie eine Liste häufiger Fehler, die auftreten können, wenn Sie eine Anforderung an einen Statistikendpunkt senden:

- `AccessDeniedException` – Rückgabecode: `400`. Meldung: `Missing Authentication Token`.
- `BadRequestException` (für Gremlin und openCypher) – Rückgabecode: `400`. Meldung: `Bad route: /pg/statistics`.
- `BadRequestException` (für RDF-Daten) – Rückgabecode: `400`. Meldung: `Bad route: /rdf/statistics`.
- `InvalidParameterException` – Rückgabecode: `400`. Meldung: `Statistics command parameter 'mode' has unsupported value '`*`the invalid value`*`'`.
- `MissingParameterException` – Rückgabecode: `400`. Meldung: `Content-type header not specified.`.
- `ReadOnlyViolationException` – Rückgabecode: `400`. Meldung: `Writes are not permitted on a read replica instance`.

Wenn Sie beispielsweise eine Anforderung senden, wenn DFE und Statistiken nicht aktiviert sind, erhalten Sie eine Antwort wie die folgende:

```
{
  "code" : "BadRequestException",
  "requestId" : "b2b8f8ee-18f1-e164-49ea-836381a3e174",
  "detailedMessage" : "Bad route: /sparql/statistics"
}
```

Abrufen eines kurzen Übersichtsberichts zu Ihrem Diagramm

Die Neptune-Diagrammübersichts-API ruft die folgenden Informationen zu Ihrem Diagramm ab:

- Bei Eigenschaftsdiagrammen (PG-Diagrammen) gibt die Diagrammübersichts-API eine schreibgeschützte Liste von Knoten- und Kantenbezeichnungen und Eigenschaftsschlüsseln sowie die Anzahl der Knoten, Kanten und Eigenschaften zurück.
- Bei Resource-Description-Framework-Diagrammen (RDF-Diagrammen) gibt die Diagrammübersichts-API eine schreibgeschützte Liste von Klassen und Prädikatschlüsseln sowie die Anzahl der Quads, Subjekte und Prädikate zurück.

> **Note**
>
> Die Diagrammübersichts-API wurde mit der Engine-Version 1.2.1.0 von Neptune eingeführt.

Mit der Diagrammübersichts-API erhalten Sie schnell eine Übersicht über die Größe und den Inhalt Ihrer Grafikdaten. Sie können die API mithilfe von `%summary`Neptune Workbench Magic auch interaktiv in einem Neptune-Notebook verwenden. In einer Diagrammanwendung kann die API verwendet werden, um die Suchergebnisse zu verbessern, indem im Rahmen der Suche Bezeichnungen für entdeckte Knoten oder Kanten bereitgestellt werden.

Die Diagrammübersichtsdaten stammen aus den DFE-Statistiken, die von der Neptune-DFE-Engine während der Laufzeit berechnet wurden, und sind verfügbar, wenn DFE-Statistiken verfügbar sind. Statistiken sind standardmäßig aktiviert, wenn Sie einen neuen Neptune-DB-Cluster erstellen.

> **Note**
>
> Die Statistikgenerierung ist für die Instance-Typen `t3` und `t4` (d. h. für die Instance-Typen `db.t3.medium` und `db.t4g.medium`) deaktiviert, um Speicherplatz zu sparen. Daher sind für diese Instance-Typen keine Diagrammübersichtsdaten verfügbar.

Sie können den Status der DFE-Statistiken mithilfe der Statistikstatus-API überprüfen. Solange die automatische Generierung von Statistiken nicht deaktiviert wurde, werden Statistiken regelmäßig automatisch aktualisiert.

Wenn Sie bei der Anforderung einer Diagrammübersicht sicherstellen möchten, dass die Statistiken so aktuell wie möglich sind, können Sie unmittelbar vor dem Abrufen der Übersicht manuell eine Statistikaktualisierung auslösen. Wenn das Diagramm während der Berechnung der Statistiken geändert wird, ist die Übersicht zwangsläufig leicht veraltet, aber nicht sehr.

Verwenden der Diagrammübersicht-API zum Abrufen von Diagrammübersichtinformationen

Die Diagrammübersicht eines Eigenschaftsdiagramms, das Sie mit Gremlin oder openCypher abfragen, können Sie vom Endpunkt der Eigenschaftsdiagramm-Übersichtsendpunkt abrufen. Für diesen Endpunkt gibt es sowohl einen langen als auch einen kurzen URI:

- `https://your-neptune-host:port/propertygraph/statistics/summary`
- `https://your-neptune-host:port/pg/statistics/summary`

Die Diagrammübersicht eines RDF-Diagramms, das Sie mit SPARQL abfragen, können Sie vom RDF-Übersichtsendpunkt abrufen:

- `https://your-neptune-host:port/rdf/statistics/summary`

Diese Endpunkte sind schreibgeschützt und unterstützen nur HTTP–`GET`Operationen. Wenn $GRAPH_SUMMARY_ENDPOINT auf die Adresse des Endpunkts festgelegt ist, den Sie abfragen möchten, können Sie die Übersichtsdaten über `curl` und HTTP-`GET` wie folgt abrufen:

```
curl -G "$GRAPH_SUMMARY_ENDPOINT"
```

Wenn beim Versuch, eine Diagrammübersicht abzurufen, keine Statistiken verfügbar sind, sieht die Antwort wie folgt aus:

```
{
  "detailedMessage": "Statistics are not available. Summary can only be generated after
 statistics are available.",
  "requestId": "48c1f788-f80b-b69c-d728-3f6df579a5f6",
  "code": "StatisticsNotAvailableException"
}
```

Der **`Modus-URL-Abfrageparameter`** für die Graph-Zusammenfassungs-API

Die Diagrammübersichts-API akzeptiert einen URL-Abfrageparameter mit dem Namen `mode`, der einen von zwei Werten annehmen kann, `basic` (Standard) und `detailed`. Die Diagrammübersichtantwort im `detailed`-Modus eines RDF-Diagramms enthält ein zusätzliches `subjectStructures`-Feld. Die detaillierte Diagrammübersichtantwort eines Eigenschaftsdiagramms enthält zwei zusätzliche Felder, `nodeStructures` und `edgeStructures`.

Um eine `detailed` Diagrammübersichtantwort anzufordern, geben Sie den Parameter `mode` wie folgt an:

```
curl -G "$GRAPH_SUMMARY_ENDPOINT?mode=detailed"
```

Wenn der Parameter `mode` nicht vorhanden ist, wird standardmäßig der `basic`-Modus verwendet. Daher ist eine ausdrückliche Angabe von `?mode=basic` möglich, aber nicht erforderlich.

Diagrammübersichtantwort für ein Eigenschaftsdiagramm (PG)

Bei einem leeren Eigenschaftsdiagramm sieht die detaillierte Diagrammübersichtantwort wie folgt aus:

```
{
  "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" : [ ]
    }
  }
}
```

Die Übersichtantwort eines Eigenschaftsdiagramms (PG) enthält die folgenden Felder:

- **`status`** – HTTP-Rückgabecode der Anforderung. Wenn die Anforderung erfolgreich ist, lautet der Code 200.

  Eine Liste mit häufigen Fehlern finden Sie unter Häufige Diagrammübersichtfehler.
- **`payload`**
  - **`version`** – Vrsion dieser Diagrammübersichtantwort.
  - **`lastStatisticsComputationTime`** – Zeitstempel nach ISO 8601 für den Zeitpunkt, an dem Neptune die Statistiken zuletzt berechnet hat.
  - **`graphSummary`**
    - **`numNodes`** – Anzahl der Knoten im Diagramm.
    - **`numEdges`** – Anzahl der Kanten im Diagramm.
    - **`numNodeLabels`** – Anzahl der eindeutigen Knotenbezeichnungen im Diagramm.
    - **`numEdgeLabels`** – Anzahl der eindeutigen Kantenbezeichnungen im Diagramm.
    - **`nodeLabels`** – Liste der eindeutigen Knotenbezeichnungen im Diagramm.
    - **`edgeLabels`** – Liste der eindeutigen Kantenbezeichnungen im Diagramm.
    - **`numNodeProperties`** – Anzahl der eindeutigen Knoteneigenschaften im Diagramm.
    - **`numEdgeProperties`** – Anzahl der eindeutigen Kanteneigenschaften im Diagramm.
    - **`nodeProperties`** – Liste der eindeutigen Knoteneigenschaften im Diagramm zusammen mit der Zahl der Knoten, bei denen die einzelnen Eigenschaften jeweils verwendet werden.
    - **`edgeProperties`** – Liste der eindeutigen Kanteneigenschaften im Diagramm zusammen mit der Zahl der Kanten, bei denen die einzelnen Eigenschaften jeweils verwendet werden.
    - **`totalNodePropertyValues`** – Gesamtzahl der Nutzungen aller Knoteneigenschaften.
    - **`totalEdgePropertyValues`** – Gesamtzahl der Nutzungen aller Kanteneigenschaften.
    - **`nodeStructures`** – Dieses Feld ist nur vorhanden, wenn in der Anforderung angegeben ist. Es enthält eine Liste von Knotenstrukturen, die jeweils die folgenden Felder enthalten:
      - **`count`** – Anzahl der Knoten, die diese spezifische Struktur aufweisen.

    - **`nodeProperties`** – Liste der Knoteneigenschaften, die in dieser spezifischen Struktur vorhanden sind.
    - **`distinctOutgoingEdgeLabels`** – Liste der eindeutigen Kantenbezeichnungen, die in dieser spezifischen Struktur vorhanden sind.
  - **`edgeStructures`** – Dieses Feld ist nur vorhanden, wenn in der Anforderung angegeben ist. Es enthält eine Liste von Kantenstrukturen, die jeweils die folgenden Felder enthalten:
    - **`count`** – Anzahl der Kanten, die diese spezifische Struktur aufweisen.
    - **`edgeProperties`** – Liste der Kanteneigenschaften, die in dieser spezifischen Struktur vorhanden sind.

Diagrammübersichtantwort für ein RDF-Diagramm

Bei einem leeren RDF-Diagramm sieht die detaillierte Diagrammübersichtantwort wie folgt aus:

```
{
  "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" : [ ]
    }
  }
}
```

Eine RDF-Diagrammübersichtantwort enthält die folgenden Felder:

- **`status`** – HTTP-Rückgabecode der Anforderung. Wenn die Anforderung erfolgreich ist, lautet der Code 200.

  Eine Liste mit häufigen Fehlern finden Sie unter Häufige Diagrammübersichtfehler.
- **`payload`**

- **`version`** – Vrsion dieser Diagrammübersichtantwort.
- **`lastStatisticsComputationTime`** – Zeitstempel nach ISO 8601 für den Zeitpunkt, an dem Neptune die Statistiken zuletzt berechnet hat.
- **`graphSummary`**
  - **`numDistinctSubjects`** – Anzahl der eindeutigen Subjekte im Diagramm.
  - **`numDistinctPredicates`** – Anzahl der eindeutigen Prädikate im Diagramm.
  - **`numQuads`** – Anzahl der Quads im Diagramm.
  - **`numClasses`** – Anzahl der Klassen im Diagramm.
  - **`classes`** – Liste der Klassen im Diagramm.
  - **`predicates`** – Liste der Prädikate im Diagramm zusammen mit der Zahl der Prädikate.
  - **`subjectStructures`** – Dieses Feld ist nur vorhanden, wenn in der Anforderung angegeben ist. Es enthält eine Liste von Subjektstrukturen, die jeweils die folgenden Felder enthalten:
    - **`count`** – Anzahl der Vorkommen dieser spezifischen Struktur.
    - **`predicates`** – Liste der Prädikate, die in dieser spezifischen Struktur vorhanden sind.

Beispiel für eine Übersichtantwort für ein Eigenschaftsdiagramm (PG)

Dies ist die detaillierte Übersichtantwort für ein Eigenschaftsdiagramm, das ein Beispiel-Eigenschaftsdiagramm für einen Flugroutendatensatz enthält:

```
{
  "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"
          ]
        }
      ]
    }
  }
}
```

Beispiel für eine RDF-Übersichtantwort

Dies ist die detaillierte Übersichtantwort für ein RDF-Diagramm, das einen Beispiel-RDF-Flugroutendatensatz enthält:

```
{
  "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"
          ]
        }
      ]
    }
  }
}
```

Verwenden AWS Identity and Access Management (IAM) -Authentifizierung mit Endpunkten für die Graphzusammenfassung

Sie können mit einer IAM-Authentifizierung sicher auf Diagrammübersicht-Endpunkte zugreifen, indem Sie awscurl oder ein anderes Tool verwenden, das mit HTTPS und IAM funktioniert. Weitere Informationen dazu, wie Sie korrekte Anmeldeinformationen einrichten, finden Sie unter Verwenden von `awscurl` mit temporären Anmeldeinformationen, um eine sichere Verbindung zu einem DB-Cluster mit aktivierter IAM-Authentifizierung herzustellen. Anschließend können Sie Anforderungen wie die folgenden senden:

```
awscurl "$GRAPH_SUMMARY_ENDPOINT" \
    --region (your region) \
    --service neptune-db
```

> **Important**
>
> Der IAM-Identität oder Rolle, die die temporären Anmeldeinformationen erstellt, muss eine IAM-Richtlinie angehängt sein, die die IAM-Aktion zulässt. GetGraphSummary

Eine Liste häufiger IAM-Fehler, die auftreten können, finden Sie unter IAM-Authentifizierungsfehler.

Häufige Fehlercodes, die eine Diagrammübersicht-Anforderung möglicherweise zurückgibt

Neptune-Servicefehlercode	HTTP Status	Fehlermeldung	Fehlerszenario	Abhilfe
`AccessDeniedException`	403	Fehlendes Authentifizierungstoken.	Eine unsignierte oder falsch signierte Anforderung wurde bei aktiviertem IAM an die Neptune-Datenbank gesendet.	Signieren Sie die Anfrage vor dem Senden mit SigV4 (siehe IAM und Diagrammübersichten).

Neptune-Servicefehlercode	HTTP Status	Fehlermeldung	Fehlerszenario	Abhilfe
	403	Benutzer: *(user ARN)* ist nicht berechtigt, Folgendes auszuführen: neptune-db: auf der Ressource:. GetGraphSummary *(resource ARN)*	Die IAM-Richtlinie erlaubt keine Aktion, GetGraphSummarywenn die Anfrage zur Graphzusammenfassung mit aktiviertem IAM an die Neptune-Datenbank gesendet wurde.	Stellen Sie sicher, dass die IAM-Richtlinie, die dem Benutzer/der Rolle angefügt ist, der/die die Anforderung stellt, die Aktion `GetGraphSummary` zulässt.
`BadRequestException`	400	Statistiken sind deaktiviert, daher ist auch die Diagrammübersicht deaktiviert.	Es wird versucht, eine Übersicht über Burstable-Instance-Typen (`t3` oder`t4g`) abzurufen, während Statistiken deaktiviert sind.	Verwenden Sie einen Instance-Typ, für den die Statistikgenerierung aktiviert ist (alle unterstützten Instances außer `t3` und `t4g`).
	400	Schlechte Route: */rdf/statistics/summarypathapi*	Die Anforderung wurde an einen ungültigen Pfad gesendet.	Verwenden Sie die korrekte Route für den Diagrammübersicht-Endpunkt.
`InvalidParameterException`	400	Die Anfrage enthält unbekannte Parameter: '*(unknown parameter or parameters)* '.	Wenn ein ungültiger Parameter in der Anforderung angegeben ist.	Verwenden Sie in der Anforderung nur gültige Parameter (z. B. `mode`).

Neptune-Servicefehlercode	HTTP Status	Fehlermeldung	Fehlerszenario	Abhilfe
`InvalidParameterException`	400	Der URI-Abfrageparameter 'mode' hat den nicht unterstützten Wert '*`(invalid value)`*'.	Wenn auf den URL-Parameter 'mode' in der Anforderung ein ungültiger Wert folgt.	Verwenden Sie gültige Werte (wie `basic` oder `detailed`), wenn Sie den URL-Parameter 'mode' angeben.
`MethodNotAllowedException`	405	Methode nicht zulässig.	Aufruf des Übersichtsendpunkts mit einer anderen HTTP-Methode als `GET` (z. B. `POST` oder `DELETE`).	Verwenden Sie die HTTP-Methode `GET`, wenn Sie den Übersichtsendpunkt aufrufen.
`StatisticsNotAvailableException`	400	Die Statistiken wurden noch nicht berechnet. Die Diagrammübersicht ist nach Abschluss der Statistikberechnung verfügbar.	Es sind keine Statistiken verfügbar, wenn die Anforderung an den Übersichtsendpunkt gesendet wird.	Warten Sie, bis die Statistikgenerierung abgeschlossen ist. Sie können den Status der Statistikgenerierung mithilfe der Statistikstatus-API überprüfen.
	400	Das Statistiklimit wurde erreicht. Daher ist keine Diagrammübersicht verfügbar.	Die Statistikgenerierung wurde beendet, da die Statistik-Größenbeschränkungen erreicht wurden.	Für dieses Diagramm ist keine Diagrammübersicht verfügbar.

Wenn Sie beispielsweise eine Anforderung an einen Diagrammübersicht-Endpunkt in einer Neptune-Datenbank senden, für die die IAM-Authentifizierung aktiviert ist, und die notwendigen

Berechtigungen sind nicht in der IAM-Richtlinie der anfordernden Person enthalten, erhalten Sie eine Antwort wie die folgende:

```
{
  "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"
}
```

Amazon-Neptune-Neptune-Konnektivität

Amazon Neptune hat einen Open-Source-JDBC-Treiber veröffentlicht, der OpenCypher-, Gremlin - und SPARQL-Abfragen unterstützt. SQL-Gremlin Die JDBC-Konnektivität vereinfacht die Herstellung von Verbindungen mit Neptune über Business-Intelligence (BI)-Tools wie Tableau. Für die Verwendung des JDBC-Treibers mit Neptune fallen keine zusätzlichen Kosten an. Sie zahlen weiter nur für die verbrauchten Neptune-Ressourcen.

Der Treiber ist mit JDBC 4.2 kompatibel und erfordert mindestens Java 8. Informationen zur Verwendung eines JDBC-Treibers finden Sie in der JDBC-API-Dokumentation.

Das GitHub Projekt, in dem Sie Probleme einreichen und Funktionsanfragen öffnen können, enthält eine ausführliche Dokumentation für den Treiber:

JDBC-Treiber für Amazon Neptune

- Verwenden von SQL mit dem JDBC-Treiber
- Verwenden von Gremlin mit dem JDBC-Treiber
- Verwenden von openCypher mit dem JDBC-Treiber
- Verwenden von SPARQL mit dem JDBC-Treiber

Erste Schritte mit dem Neptune-JDBC-Treiber

Um den Neptune-JDBC-Treiber für die Verbindung mit einer Neptune-Instance zu verwenden, muss entweder der JDBC-Treiber auf einer Amazon-EC2-Instance in derselben VPC wie Ihr Neptune-DB-Cluster bereitgestellt werden oder die Instance muss über einen SSH-Tunnel oder Load Balancer verfügbar sein. Ein SSH-Tunnel kann intern im Treiber oder extern eingerichtet werden.

Sie können den Treiber hier herunterladen. Der Treiber wird als einzelne JAR-Datei mit einem Namen wie `neptune-jdbc-1.0.0-all.jar` bereitgestellt. Um ihn zu verwenden, platzieren Sie die JAR-Datei im `classpath` Ihrer Anwendung. Wenn Ihre Anwendung Maven oder Gradle verwendet, können Sie die entsprechenden Maven- oder Gradle-Befehle verwenden, um den Treiber aus der JAR-Datei zu installieren.

Der Treiber benötigt eine JDBC-Verbindungs-URL, um eine Verbindung mit Neptune herzustellen. Das Format ist wie folgt:

```
jdbc:neptune:(connection
 type)://(host);property=value;property=value;...;property=value
```

In den Abschnitten für jede Abfragesprache im GitHub Projekt werden die Eigenschaften beschrieben, die Sie in der JDBC-Verbindungs-URL für diese Abfragesprache festlegen können.

Wenn sich die JAR-Datei im `classpath` Ihrer Anwendung befindet, ist keine weitere Konfiguration notwendig. Sie können den Treiber über die JDBC-`DriverManager`-Schnittstelle und eine Neptune-Verbindungszeichenfolge verbinden. Wenn Ihr Neptune-DB-Cluster beispielsweise über den Endpunkt `neptune-example.com` auf Port 8182 zugänglich ist, könnten Sie wie folgt eine Verbindung zu openCypher herstellen:

```
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();
}
```

In den Dokumentationsabschnitten des GitHub Projekts für jede Abfragesprache wird beschrieben, wie die Verbindungszeichenfolge erstellt wird, wenn diese Abfragesprache verwendet wird.

Verwenden von Tableau mit dem Neptune-JDBC-Treiber

Um Tableau mit dem Neptune-JDBC-Treiber zu verwenden, laden Sie zunächst die neueste Version von Tableau Desktop herunter und installieren diese. Laden Sie die JAR-Datei für den Neptune-JDBC-Treiber sowie die Neptune-Tableau-Connector-Datei (eine Datei mit der Erweiterung `.taco`) herunter.

Herstellen einer Verbindung zu Tableau for Neptune auf einem Mac-Computer

1. Platzieren Sie die JAR-Datei mit dem Neptune-JDBC-Treiber im Ordner `/Users/(your user name)/Library/Tableau/Drivers`.

2. Platzieren Sie die Neptune-Tableau-Connector-Datei `.taco` im Ordner `/Users/(your user name)/Documents/My Tableau Repository/Connectors`.
3. Wenn die IAM-Authentifizierung aktiviert ist, richten Sie die Umgebung für diese ein. Beachten Sie, dass Umgebungsvariablen, die in `.zprofile/`, `.zshenv/`, `.bash_profile` usw. festgelegt sind, nicht funktionieren. Die Umgebungsvariablen müssen so festgelegt werden, dass sie von einer GUI-Anwendung geladen werden können.

   Eine Möglichkeit für die Einrichtung Ihrer Anmeldeinformationen besteht darin, Ihren Zugriffsschlüssel und Geheimschlüssel in der Datei `/Users/(your user name)/.aws/credentials` zu platzieren.

   Eine einfache Möglichkeit für die Einrichtung der Serviceregion besteht in der Öffnung eines Terminals und der Eingabe des folgenden Befehls unter Verwendung der Region Ihrer Anwendung (z. B. `us-east-1`):

   ```
   launchctl setenv SERVICE_REGION region name
   ```

   Es gibt weitere Möglichkeiten für die Einrichtung von Umgebungsvariablen, die nach einem Neustart beibehalten werden. Unabhängig von der verwendeten Technik müssen jedoch Variablen festgelegt werden, auf die eine GUI-Anwendung zugreifen kann.
4. Um Umgebungsvariablen in eine Mac-GUI zu laden, geben Sie diesen Befehl auf einem Terminal ein:

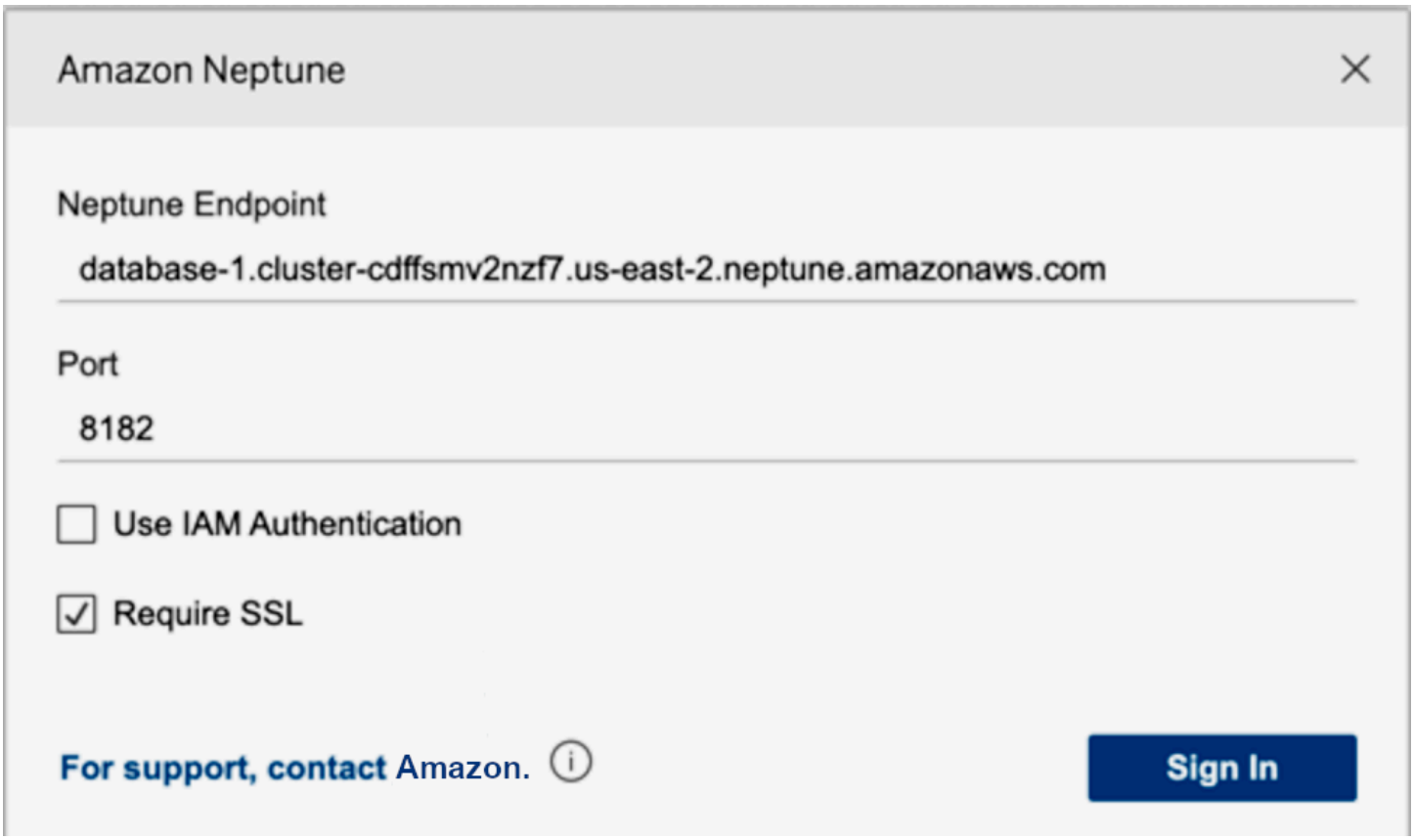
   ```
   /Applications/Tableau/Desktop/2021.1.app/Contents/MacOS/Tableau
   ```

Herstellen einer Verbindung zu Tableau for Neptune auf einem Windows-Computer

1. Platzieren Sie die JAR-Datei mit dem Neptune-JDBC-Treiber im Ordner `C:\Program Files\Tableau\Drivers`.
2. Platzieren Sie die Neptune-Tableau-Connector-Datei `.taco` im Ordner `C:\Users\(your user name)\Documents\My Tableau Repository\Connectors`.
3. Wenn die IAM-Authentifizierung aktiviert ist, richten Sie die Umgebung für diese ein.

   Dies kann so einfach wie die Einrichtung wie das Einstellen von der Umgebungsvariablen `ACCESS_KEY`, `SECRET_KEY` und `SERVICE_REGION` für Benutzer.

Wählen Sie bei geöffnetem Tableau links im Fenster Mehr aus. Wenn die Tableau-Connector-Datei gefunden wurde, können Sie Amazon Neptune by AWS in der angezeigten Liste auswählen:



Sie sollten den Port nicht bearbeiten oder Verbindungsoptionen hinzufügen müssen. Geben Sie Ihren Neptune-Endpunkt ein und legen Sie Ihre IAM- und SSL-Konfiguration fest. (Sie müssen SSL aktivieren, wenn Sie IAM verwenden.)

Wenn Sie Anmelden auswählen, kann es länger als 30 Sekunden dauern, bis die Verbindung hergestellt ist, wenn Ihr Diagramm groß ist. Tableau sammelt Eckpunkt- und Kantentabellen, verbindet Eckpunkte an Kanten und erstellt Visualisierungen.

Fehlerbehebung für JDBC-Treiberverbindungen

Wenn der Treiber keine Verbindung zum Server herstellen kann, überprüfen Sie mithilfe der Funktion `isValid` des JDBC-Objekts `Connection`, ob die Verbindung gültig ist. Wenn die Funktion `false` zurückgibt, d. h. wenn die Verbindung ungültig ist, überprüfen Sie, ob der Endpunkt korrekt ist, mit dem die Verbindung hergestellt werden soll, ob Sie sich in der VPC Ihres Neptune-DB-Clusters befinden oder ob es einen gültigen SSH-Tunnel zum Cluster gibt.

Wenn Sie die Antwort `No suitable driver found for (connection string)` für den Aufruf `DriverManager.getConnection` erhalten, liegt wahrscheinlich ein Problem zu Beginn der Verbindungszeichenfolge vor. Stellen Sie sicher, dass die Verbindungszeichenfolge wie folgt beginnt:

```
jdbc:neptune:opencypher://...
```

Um weitere Informationen zur Verbindung zu erhalten, können Sie der Verbindungszeichenfolge wie folgt ein `LogLevel` hinzufügen:

```
jdbc:neptune:opencypher://(JDBC URL):(port);logLevel=trace
```

Alternativ können Sie in Ihren Eingabeeigenschaften `properties.put("logLevel", "trace")` hinzufügen, um Ablaufverfolgungsinformationen zu protokollieren.

Updates der Amazon-Neptune-Engine

Amazon Neptune veröffentlicht regelmäßig Updates der Engine. Sie können mithilfe der Instance-Status-API die aktuell installierte Engine-Version ermitteln.

Engine-Versionen werden unter Engine-Versionen für Amazon Neptune und Patches unter Neueste Updates aufgelistet.

Weitere Informationen dazu, wie Updates veröffentlicht werden und wie Sie die Neptune-Engine in Ihrer Datenbank aktualisieren können, finden Sie unter Clusterwartung. Die Versionsnummerierung wird beispielsweise unter Engine-Versionsnummern erläutert.

Ausnahmebehandlung und Wiederholungen

Die Entwicklung robuster Anwendungen auf Neptune bedeutet oft, sich auf das Unerwartete vorzubereiten, insbesondere wenn es um die Behandlung von Fehlern geht, die von der Datenbank zurückgegeben werden. Eine der häufigsten Reaktionen auf serverseitige Ausnahmen besteht darin, den fehlgeschlagenen Vorgang erneut zu versuchen. Die Wiederholungslogik ist für ausfallsichere Systeme zwar unerlässlich, Sie müssen sich jedoch darüber im Klaren sein, dass nicht alle Fehler auf die gleiche Weise behandelt werden sollten. Anstatt sich auf generisches Wiederholungsverhalten zu verlassen, kann Ihnen ein durchdachter Ansatz dabei helfen, zuverlässigere und effizientere Anwendungen zu entwickeln.

Warum Wiederholungslogik wichtig ist

Die Wiederholungslogik ist eine wichtige Komponente jeder verteilten Anwendung. Vorübergehende Probleme wie Netzwerkinstabilität, vorübergehende Ressourcenbeschränkungen oder gleichzeitige Änderungskonflikte können dazu führen, dass Operationen fehlschlagen. In vielen Fällen deuten diese Fehler nicht auf ein dauerhaftes Problem hin und können behoben werden, indem Sie warten und es erneut versuchen. Durch die Implementierung einer soliden Wiederholungsstrategie wird der Realität unvollständiger Umgebungen in verteilten Systemen Rechnung getragen und eine höhere Zuverlässigkeit und Kontinuität gewährleistet, ohne dass manuelle Eingriffe erforderlich sind.

Die Risiken wahlloser Wiederholungsversuche

Wenn standardmäßig jeder Fehler wiederholt wird, kann dies mehrere unbeabsichtigte Folgen haben:

- Vermehrter Konflikt — Wenn Operationen, die aufgrund hoher Parallelität fehlschlagen, wiederholt wiederholt werden, kann sich der Konflikt insgesamt noch verschlimmern. Dies kann zu einem Zyklus fehlgeschlagener Transaktionen und Leistungseinbußen führen.
- Erschöpfung der Ressourcen — Wahllose Wiederholungsversuche können zusätzliche Systemressourcen verbrauchen, sowohl auf der Client- als auch auf der Serverseite. Dies kann möglicherweise zu einer Drosselung oder sogar zu einer Verschlechterung des Dienstes führen.
- Höhere Latenz für Clients — Übermäßige Wiederholungsversuche können zu erheblichen Verzögerungen bei Client-Anwendungen führen, insbesondere wenn jede Wiederholung mit Wartezeiten verbunden ist. Dies kann sich negativ auf die Benutzererfahrung und nachgelagerte Prozesse auswirken.

Entwicklung einer praktischen Wiederholungsstrategie

Um eine robuste und effiziente Anwendung zu erstellen, sollten Sie eine Strategie für Wiederholungsversuche entwickeln, die auf die spezifischen Fehlerbedingungen zugeschnitten ist, auf die Ihre Anwendung stoßen kann. Im Folgenden finden Sie einige Überlegungen, die Sie bei Ihrem Vorgehen berücksichtigen sollten:

- Identifizieren Sie Fehler, die wiederholt werden können — Nicht alle Ausnahmen sollten erneut versucht werden. Beispielsweise sollten Syntaxfehler, Authentifizierungsfehler oder ungültige Abfragen keinen erneuten Versuch auslösen. Neptune bietet Fehlercodes und allgemeine Empfehlungen, bei denen Fehler sicher wiederholt werden können. Sie müssen jedoch die Logik implementieren, die zu Ihrem Anwendungsfall passt.
- Implementieren Sie exponentielles Backoff — Verwenden Sie bei vorübergehenden Fehlern eine exponentielle Backoff-Strategie, um die Wartezeit zwischen Wiederholungen schrittweise zu verlängern. Dies trägt dazu bei, Konflikte zu vermeiden und das Risiko kaskadierender Ausfälle zu verringern.
- Berücksichtigen Sie die Länge der ersten Pause: Wenn Sie den ersten Versuch zu schnell ausführen, wird möglicherweise der gleiche Fehler angezeigt, wenn dem Server nicht genügend Zeit zur Freigabe von Ressourcen eingeräumt wurde, die für eine erfolgreiche Abfrage erforderlich sind. Eine längere Pause in den richtigen Situationen könnte ungenutzte Anfragen und die Serverbelastung verringern.
- Hinzufügen von Jitter zum Backoff — Exponentielles Backoff ist zwar effektiv, kann aber dennoch zu synchronisierten Wiederholungsstürmen führen, wenn viele Clients gleichzeitig ausfallen und es dann gemeinsam erneut versuchen. Das Hinzufügen von Jitter, einer kleinen zufälligen Variation der Backoff-Verzögerung, trägt dazu bei, die Anzahl der Wiederholungsversuche zu verteilen, wodurch die Wahrscheinlichkeit verringert wird, dass alle Clients es gleichzeitig erneut versuchen und es zu einer weiteren Lastspitze kommt.
- Wiederholungsversuche einschränken — Legen Sie eine angemessene Höchstzahl von Wiederholungsversuchen fest, um Endlosschleifen und eine Erschöpfung der Ressourcen zu verhindern.
- Überwachen und anpassen — Überwachen Sie kontinuierlich die Fehlerquote Ihrer Anwendung und passen Sie Ihre Wiederholungsstrategie nach Bedarf an. Wenn Sie eine hohe Anzahl von Wiederholungsversuchen für einen bestimmten Vorgang feststellen, sollten Sie überlegen, ob der Vorgang optimiert oder serialisiert werden kann.

Eine eingehendere Erläuterung von exponentiellem Backoff und Jitter als allgemeinem Cloud-Entwurfsmuster finden Sie unter Wiederholungsversuch mit Backoff in Prescriptive Guidance. AWS

Beispielszenarien

Die richtige Wiederholungsstrategie hängt von der Art des Fehlers, der Arbeitslast und den beobachteten Fehlermustern ab. In der folgenden Tabelle sind einige der häufigsten Fehlerszenarien zusammengefasst und es wird erläutert, wie sich die Überlegungen zur Wiederholungsstrategie auf die einzelnen Szenarien auswirken. Für zusätzlichen Kontext folgen erläuternde Absätze.

Szenario	Wiederholbar?	Backoff & Jitter	Erste Pause	Limit für Wiederholungsversuche	Überwachen und anpassen
Gelegentliche CME bei kurzen Anfragen	Ja	Kurzer Backoff, Jitter hinzufügen	Kurz (z. B. 100 ms)	Hoch	Achten Sie auf steigende CME-Raten
Häufige CME-Anfragen Longer-Running	Ja	Längerer Backoff, zusätzlicher Jitter	Länger (zum Beispiel 2 Sekunden)	Mittel	Untersuchen Sie Konflikte und reduzieren Sie sie
Speicherlimits für teure Abfragen	Ja	Langer Rückzieher	Lang (zum Beispiel 5-10 Sekunden)	Niedrig	Abfrage optimieren, Warnung, wenn sie persistent ist
Timeout bei moderaten Abfragen	Vielleicht	Backoff moderieren, Jitter hinzufügen	Mäßig (z. B. 1s)	Gering bis Mäßig	Beurteilen Sie die Serverlast und den Abfrageentwurf

Szenario 1: Gelegentliche CME bei kurzen Abfragen

Bei einem Workload, der bei kurzen, einfachen Updates selten `ConcurrentModificationException` auftritt, sind diese Fehler in der Regel vorübergehend und können problemlos wiederholt werden. Machen Sie vor dem ersten Versuch eine kurze Anfangspause (z. B. 100 Millisekunden). In dieser Zeit kann jede kurzzeitige Sperre aufgehoben werden. Kombinieren Sie dies mit einem kurzen exponentiellen Backoff und Jitter, um synchronisierte Wiederholungsversuche zu vermeiden. Da die Kosten für Wiederholungsversuche gering sind, ist ein höheres Wiederholungslimit sinnvoll. Behalten Sie dennoch die CME-Rate im catch, um jeden Trend zu einer zunehmenden Konkurrenz in Ihren Daten zu erkennen.

Szenario 2: Häufiges CME bei Abfragen mit langer Laufzeit

Wenn in Ihrer Anwendung häufig CMEs bei Abfragen mit langer Laufzeit auftreten, deutet dies auf schwerwiegendere Konflikte hin. Beginnen Sie in diesem Fall mit einer längeren Anfangspause (z. B. 2 Sekunden), um der aktuellen Abfrage, für die die Sperre gilt, genügend Zeit zum Abschluss zu geben. Verwenden Sie einen längeren exponentiellen Backoff und fügen Sie Jitter hinzu. Begrenzen Sie die Anzahl der Wiederholungen, um übermäßige Verzögerungen und eine übermäßige Ressourcennutzung zu vermeiden. Wenn der Konflikt weiterhin besteht, überprüfen Sie Ihren Workload auf Muster und erwägen Sie, Updates zu serialisieren oder die Parallelität zu reduzieren, um die Ursache zu beheben.

Szenario 3: Speicherlimits für teure Abfragen

Wenn während einer bekanntermaßen ressourcenintensiven Abfrage speicherbasierte Fehler auftreten, können Wiederholungsversuche sinnvoll sein, allerdings erst nach einer langen Anfangspause (z. B. 5 bis 10 Sekunden oder länger), damit der Server Ressourcen freigeben kann. Verwenden Sie eine lange Backoff-Strategie und legen Sie ein niedriges Wiederholungslimit fest, da wiederholte Fehler ohne Änderungen an der Abfrage oder der Arbeitslast kaum behoben werden können. Dauerhafte Fehler sollten Warnmeldungen auslösen und eine Überprüfung der Abfragekomplexität und der Ressourcennutzung veranlassen.

Szenario 4: Timeout bei moderaten Abfragen

Ein Timeout bei einer mäßig teuren Abfrage ist ein zweideutigerer Fall. Manchmal kann ein erneuter Versuch erfolgreich sein, wenn der Timeout auf einen vorübergehenden Anstieg der Serverlast oder der Netzwerkbedingungen zurückzuführen ist. Beginnen Sie mit einer moderaten Anfangspause (z. B. 1 Sekunde), um dem System die Möglichkeit zu geben, sich zu erholen. Wenden Sie einen

moderaten Back-Off an und fügen Sie Jitter hinzu, um synchronisierte Wiederholungsversuche zu vermeiden. Halten Sie das Wiederholungslimit niedrig bis moderat, da wiederholte Timeouts auf ein tieferes Problem mit der Abfrage oder der Serverkapazität hindeuten können. Achten Sie auf Muster: Prüfen Sie bei häufigen Timeouts, ob die Abfrage optimiert werden muss oder ob der Neptune-Cluster zu wenig bereitgestellt ist.

Überwachung und Beobachtbarkeit

Die Überwachung ist ein wichtiger Bestandteil jeder Wiederholungsstrategie. Effektive Beobachtbarkeit hilft Ihnen zu verstehen, wie gut Ihre Wiederholungslogik funktioniert, und gibt frühzeitig Hinweise, wenn etwas in Ihrer Workload- oder Clusterkonfiguration behoben werden muss.

MainRequestQueuePendingRequests

Diese CloudWatch Metrik verfolgt die Anzahl der Anfragen, die in der Eingabewarteschlange von Neptune warten. Ein steigender Wert weist darauf hin, dass sich Anfragen häufen. Dies kann ein Zeichen für übermäßige Konflikte, unzureichend bereitgestellte Ressourcen oder Wiederholungsversuche sein. Durch die Überwachung dieser Kennzahl können Sie erkennen, wann Ihre Wiederholungsstrategie Warteschlangenprobleme verursacht oder verschärft, und Sie können dazu veranlassen, Ihre Vorgehensweise anzupassen, bevor Fehler eskalieren.

Andere Metriken CloudWatch

Andere Neptune-Metriken wie `CPUUtilization`, `TotalRequestsPerSecond`, und Abfragelatenz bieten zusätzlichen Kontext. Ein hoher CPU-Wert in I/O Kombination mit einer wachsenden Warteschlangenlänge kann beispielsweise darauf hindeuten, dass Ihr Cluster überlastet ist oder dass Abfragen zu groß oder zu häufig sind. CloudWatch Für diese Messwerte können Alarme eingerichtet werden, um Sie auf abnormales Verhalten aufmerksam zu machen und Ihnen zu helfen, Spitzenwerte bei Fehlern oder Wiederholungsversuchen mit den zugrunde liegenden Ressourcenbeschränkungen zu korrelieren.

Neptun-Status- und Abfrage-APIs

Die Neptune Status API für Gremlin und ihre analogen APIs für und SPARQL bieten eine Echtzeitansicht der akzeptierten OpenCypherund auf dem Cluster ausgeführten Abfragen, was nützlich ist, um Engpässe zu diagnostizieren oder die Auswirkungen der Wiederholungslogik in Echtzeit zu verstehen.

Durch die Kombination dieser Überwachungstools können Sie:

- Erkennen Sie, wann Wiederholungsversuche zu Warteschlangen und Leistungseinbußen beitragen.
- Finden Sie heraus, wann Sie Ihren Neptun-Cluster skalieren oder Abfragen optimieren sollten.
- Stellen Sie sicher, dass Ihre Wiederholungsstrategie vorübergehende Fehler behebt, ohne tiefere Probleme zu maskieren.
- Lassen Sie sich frühzeitig vor neu auftretenden Konflikten oder Ressourcenerschöpfung warnen.

Proaktive Überwachung und Warnmeldungen sind für die Aufrechterhaltung einer fehlerfreien Neptune-Bereitstellung unerlässlich, insbesondere wenn die Parallelität und Komplexität Ihrer Anwendung zunehmen.

Sicherung Ihrer Amazon Neptune Neptune-Datenbank

Cloud-Sicherheit hat höchste AWS Priorität. Als AWS Kunde profitieren Sie von einer Rechenzentrums- und Netzwerkarchitektur, die darauf ausgelegt sind, die Anforderungen der sicherheitssensibelsten Unternehmen zu erfüllen.

Sicherheit ist eine gemeinsame Verantwortung von Ihnen AWS und Ihnen. Das Modell der geteilten Verantwortung beschreibt dies als Sicherheit der Cloud und Sicherheit in der Cloud:

- Sicherheit der Cloud — AWS ist verantwortlich für den Schutz der Infrastruktur, die AWS Dienste in der AWS Cloud ausführt. AWS bietet Ihnen auch Dienste, die Sie sicher nutzen können. Third-party Auditoren testen und verifizieren regelmäßig die Wirksamkeit unserer Sicherheitsmaßnahmen im Rahmen der AWS Compliance-Programme. Informationen zu den für Amazon Neptune geltenden Compliance-Programmen finden Sie unter Im Rahmen des Compliance-Programms zugelassene - Services.
- Sicherheit in der Cloud — Ihre Verantwortung richtet sich nach dem AWS Service, den Sie nutzen. Sie sind auch für andere Faktoren verantwortlich, etwa für die Vertraulichkeit Ihrer Daten, für die Anforderungen Ihres Unternehmens und für die geltenden Gesetze und Vorschriften.

Diese Dokumentation hilft Ihnen, zu verstehen, wie Sie das Modell der geteilten Verantwortung bei der Verwendung von Neptune anwenden. Die folgenden Themen zeigen Ihnen, wie Sie Neptune zur Erfüllung Ihrer Sicherheits- und Compliance-Ziele konfigurieren können. Sie lernen auch, wie Sie andere AWS Dienste nutzen können, die Ihnen helfen, Ihre Neptune-Ressourcen zu überwachen und zu sichern.

Topics

- Upgrades des Amazon Neptune Neptune-Betriebssystems
- Schutz von Daten in Ihrer Amazon Neptune Neptune-Datenbank
- Authentifizieren Sie Ihre Amazon Neptune Neptune-Datenbank mit AWS Identity and Access Management
- Aktivierung der IAM-Datenbankauthentifizierung in Amazon Neptune
- Herstellen einer Verbindung zu Ihrer Amazon Neptune Neptune-Datenbank mit AWS Identity and Access Management Authentifizierung
- Verwaltung des Zugriffs auf Amazon Neptune Neptune-Datenbanken mithilfe von IAM-Richtlinien
- Verwenden von serviceverknüpften Rollen für Amazon Neptune

- Temporäre Anmeldeinformationen verwenden, um eine Verbindung zu Amazon Neptune herzustellen
- Protokollierung und Überwachung von Nutzung und Leistung in Amazon Neptune
- Amazon Neptune- und Schnittstellen-VPC-Endpunkte (AWS PrivateLink)
- Überlegungen zur Einhaltung von Vorschriften für Amazon Neptune
- Aufbau robuster und katastrophentoleranter Amazon Neptune Neptune-Bereitstellungen

Upgrades des Amazon Neptune Neptune-Betriebssystems

Amazon Neptune sorgt durch regelmäßige Betriebssystem-Upgrades für kontinuierliche Verbesserungen der Datenbankleistung, Sicherheit und Stabilität. Diese Upgrades werden nach einem strukturierten Aktualisierungsprozess sowohl auf Neptune Databases als auch auf Neptune Analytics angewendet. Amazon Neptune veröffentlicht mindestens einmal pro Monat Betriebssystem-Upgrades.

Neptune Database OS-Updates, für die keine Ausfallzeiten erforderlich sind, werden während des Wartungsfensters automatisch angewendet. Bestimmte Betriebssystemupdates (z. B. ein Upgrade der Kernelversion) erfordern einen Neustart der Instanz. Diese Updates sind optional und haben kein festes Datum, an dem sie angewendet werden sollen. Wenn Sie diese Updates jedoch nicht installieren, werden sie möglicherweise irgendwann erforderlich und werden während des Wartungsfensters Ihrer Instanz automatisch angewendet.

Neptune Analytics — Die Betriebssystem-Upgrades von Neptune Analytics sind nahtlos und erfordern keine Kundenaktion. Diese Updates werden automatisch und ohne Ausfallzeiten im Hintergrund angewendet.

Um Sicherheit und Compliance zu gewährleisten, empfehlen wir Ihnen, alle von Amazon Neptune bereitgestellten Updates während Ihres Wartungsfensters routinemäßig zu installieren. Wenn Sie bei allen optionalen und obligatorischen Updates auf dem Laufenden bleiben, können Sie wichtige Sicherheitspatches integrieren und die Einhaltung verschiedener Compliance-Verpflichtungen sicherstellen. Veraltete Betriebssystemversionen können zur Nichteinhaltung gesetzlicher Anforderungen führen.

Minimierung der Ausfallzeiten für Betriebssystem-Upgrades, die einen Neustart erfordern

Für Betriebssystem-Upgrades, die einen Neustart erfordern, empfehlen wir, dass Sie zuerst die Reader-Instances in einem Cluster und dann die Writer-Instanz aktualisieren, um die Verfügbarkeit Ihres Clusters zu maximieren. Es wird nicht empfohlen, die Reader- und Writer-Instanzen gleichzeitig zu aktualisieren, da es bei einem Failover zu längeren Ausfallzeiten kommen könnte.

Betriebssystem-Upgrades auf Ihre Neptune-DB-Instance anwenden

Neptune-DB-Instances erfordern gelegentlich Betriebssystemupdates. Amazon Neptune aktualisiert das Betriebssystem auf eine neuere Version, um die Datenbankleistung und die allgemeine Sicherheitslage der Kunden zu verbessern. In der Regel dauern die Updates etwa 10 Minuten. Betriebssystemupdates ändern nicht die DB-Engine-Version oder die DB-Instance-Klasse einer DB-Instance.

Um benachrichtigt zu werden, wenn ein neues optionales Update verfügbar ist, können Sie es `RDS-EVENT-0230` in der Kategorie Sicherheitspatching-Ereignisse abonnieren. Informationen zum Abonnieren von Amazon Neptune Neptune-Veranstaltungen finden Sie unter Neptune-Ereignisbenachrichtigung abonnieren.

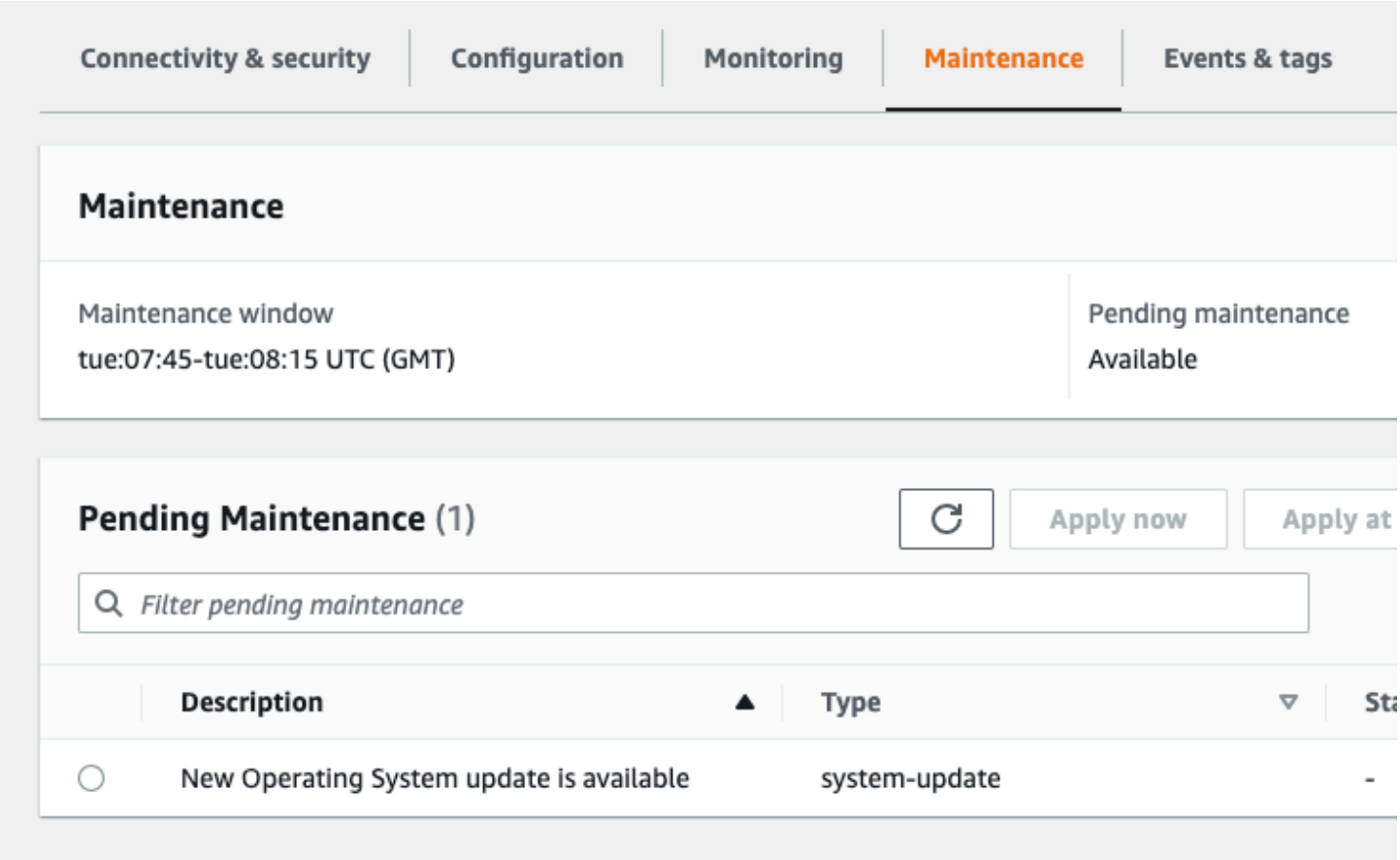
> **Important**
>
> Ihre Amazon Neptune DB-Instance wird während des Betriebssystem-Upgrades offline geschaltet. Sie können die Cluster-Ausfallzeiten minimieren, indem Sie einen Cluster mit mehreren Instanzen einrichten. Wenn Sie keinen Cluster mit mehreren Instanzen haben, können Sie vorübergehend einen erstellen, indem Sie sekundäre Instanzen hinzufügen, um diese Wartung durchzuführen, und dann die zusätzlichen Reader-Instances löschen, sobald die Wartung abgeschlossen ist (es fallen reguläre Gebühren für die sekundäre Instanz an).

Sie können die AWS-Managementkonsole oder die AWS CLI verwenden, um festzustellen, ob ein Update verfügbar ist.

Verwendung der AWS-Managementkonsole

Um festzustellen, ob ein Update verfügbar ist, verwenden Sie AWS-Managementkonsole:

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die Amazon Neptune Neptune-Konsole unter. https://console.aws.amazon.com/neptune/home
2. Wählen Sie im Navigationsbereich Clusters und dann die Instance aus.
3. Wählen Sie Maintenance aus.
4. Suchen Sie im Abschnitt „Ausstehende Wartung“ nach dem Betriebssystem-Update.



Sie können das Betriebssystem-Update auswählen und im Bereich Ausstehende Wartung auf Jetzt anwenden oder Beim nächsten Wartungsfenster anwenden klicken. Wenn der Wartungswert „Nächstes Fenster“ lautet, verschieben Sie die Wartungsarbeiten, indem Sie „Upgrade aufschieben“ wählen. Sie können eine Wartungsaktion nicht verschieben, wenn sie bereits gestartet wurde.

Alternativ können Sie die Instanz aus einer Liste von Clustern auswählen, indem Sie im Navigationsbereich auf Cluster klicken und im Menü Aktionen die Option Jetzt anwenden oder Beim nächsten Wartungsfenster anwenden auswählen.

Verwendung der AWS CLI

Rufen Sie den folgenden Befehl auf, um festzustellen, ob ein Update verfügbar ist AWS CLI, indem Sie den `describe-pending-maintenance-actions` folgenden Befehl aufrufen:

```
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"
        }
    ]
}
```

Rufen Sie den `apply-pending-maintenance-action` folgenden Befehl auf, um die Betriebssystemupdates anzuwenden:

```
aws neptune apply-pending-maintenance-action \
    --apply-action system-update \
    --resource-identifier (ARN of your DB instance) \
    --opt-in-type immediate
```

Schutz von Daten in Ihrer Amazon Neptune Neptune-Datenbank

Das AWS Modell der gilt für den Datenschutz in Amazon Neptune. Wie in diesem Modell beschrieben, AWS ist verantwortlich für den Schutz der globalen Infrastruktur, auf der alle Systeme laufen. AWS Cloud Sie sind dafür verantwortlich, die Kontrolle über Ihre in dieser Infrastruktur gehosteten Inhalte zu behalten. Sie sind auch für die Sicherheitskonfiguration und die Verwaltungsaufgaben für die von Ihnen verwendeten AWS-Services verantwortlich. Weitere Informationen zum Datenschutz finden Sie unter Häufig gestellte Fragen zum Datenschutz . Informationen zum Datenschutz in Europa finden Sie im General Data Protection Regulation (GDPR) Center.

Aus Datenschutzgründen empfehlen wir, dass Sie Ihre AWS-Konto Anmeldeinformationen schützen und einzelne Benutzer mit AWS IAM Identity Center oder AWS Identity and Access Management

(IAM) einrichten. So erhält jeder Benutzer nur die Berechtigungen, die zum Durchführen seiner Aufgaben erforderlich sind. Außerdem empfehlen wir, die Daten mit folgenden Methoden schützen:

- Verwenden Sie für jedes Konto die Multi-Faktor-Authentifizierung (MFA).
- Wird verwendet SSL/TLS , um mit AWS Ressourcen zu kommunizieren. Wir benötigen TLS 1.2 und empfehlen TLS 1.3.
- Richten Sie die API und die Protokollierung von Benutzeraktivitäten mit ein AWS CloudTrail. Informationen zur Verwendung von CloudTrail Pfaden zur Erfassung von AWS Aktivitäten finden Sie unter Arbeiten mit CloudTrail Pfaden im AWS CloudTrail Benutzerhandbuch.
- Verwenden Sie AWS Verschlüsselungslösungen zusammen mit allen darin enthaltenen Standardsicherheitskontrollen AWS-Services.
- Verwenden Sie erweiterte verwaltete Sicherheitsservices wie Amazon Macie, die dabei helfen, in Amazon S3 gespeicherte persönliche Daten zu erkennen und zu schützen.
- Wenn Sie für den Zugriff AWS über eine Befehlszeilenschnittstelle oder eine API FIPS 140-3-validierte kryptografische Module benötigen, verwenden Sie einen FIPS-Endpunkt. Weitere Informationen über verfügbare FIPS-Endpunkte finden Sie unter Federal Information Processing Standard (FIPS) 140-3.

Wir empfehlen dringend, in Freitextfeldern, z. B. im Feld Name, keine vertraulichen oder sensiblen Informationen wie die E-Mail-Adressen Ihrer Kunden einzugeben. Dies gilt auch, wenn Sie mit Neptune oder anderen AWS-Services über die Konsole AWS CLI, API oder AWS SDKs arbeiten. Alle Daten, die Sie in Tags oder Freitextfelder eingeben, die für Namen verwendet werden, können für Abrechnungs- oder Diagnoseprotokolle verwendet werden. Wenn Sie eine URL für einen externen Server bereitstellen, empfehlen wir dringend, keine Anmeldeinformationen zur Validierung Ihrer Anforderung an den betreffenden Server in die URL einzuschließen.

> **Important**
>
> TLS 1.3 wird nur für Neptune Engine-Version 1.3.2.0 und höher unterstützt.

Sie verwenden AWS veröffentlichte API-Aufrufe, um Neptune über das Netzwerk zu verwalten. Clients müssen Transport Layer Security (TLS) 1.2 oder höher mit Suiten für eine starke Verschlüsselung unterstützen, wie in Verbindungen zu Neptune-Datenbanken verschlüsseln beschrieben. Die meisten modernen Systeme wie Java 7 und höher unterstützen diese Modi.

In den folgenden Abschnitten wird beschrieben, wie Neptune-Daten geschützt werden.

Themen

- Schützen Sie Ihre Amazon Neptune Neptune-Datenbank mit Amazon VPC
- Verschlüsseln von Verbindungen zu Ihrer Amazon Neptune Neptune-Datenbank mit SSL/HTTPS
- Neptun-Ressourcen im Ruhezustand verschlüsseln

Schützen Sie Ihre Amazon Neptune Neptune-Datenbank mit Amazon VPC

Ein Amazon Neptune Neptune-DB-Cluster kann nur in einer Amazon Virtual Private Cloud (Amazon VPC) erstellt werden, und seine Endpunkte sind innerhalb dieser VPC zugänglich, normalerweise von einer Amazon Elastic Compute Cloud (Amazon EC2) -Instance aus, die in dieser VPC ausgeführt wird. Alternativ kann über einen öffentlichen Endpunkt darauf zugegriffen werden. Weitere Informationen zu öffentlichen Endpunkten finden Sie unter Öffentliche Endpunkte von Neptune.

Sie können Ihre Neptune-Daten schützen, indem Sie den Zugriff auf die VPC einschränken, in der sich Ihr Neptune-DB-Cluster befindet, wie in Verbindung zu einem Amazon Neptune Neptune-Cluster herstellen beschrieben.

Verschlüsseln von Verbindungen zu Ihrer Amazon Neptune Neptune-Datenbank mit SSL/HTTPS

Amazon Neptune erlaubt nur Secure Sockets Layer (SSL) -Verbindungen über HTTPS zu beliebigen Instances oder Cluster-Endpunkten.

Neptune benötigt mindestens TLS Version 1.2 und verwendet die folgenden starken Verschlüsselungssammlungen:

- `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`

Ab der Neptune-Engine-Version 1.3.2.0 unterstützt Neptune TLS Version 1.3 mit den folgenden Verschlüsselungssammlungen:

- TLS_AES_128_GCM_SHA256
- TLS_AES_256_GCM_SHA384

Auch wenn HTTP-Verbindungen in früheren Engine-Versionen zulässig waren, müssen alle DB-Cluster, die eine neue DB-Cluster-Parametergruppe verwenden, standardmäßig SSL verwenden. Um Ihre Daten zu schützen, unterstützen Neptune-Endpunkte in Engine-Version *1.0.4.0* und höher ausschließlich HTTPS-Anfragen. Weitere Informationen finden Sie unter Herstellen von Verbindungen mit einer Neptune-DB-Instance über den HTTP-REST-Endpunkt.

Neptune stellt automatisch SSL-Zertifikate für Ihre Neptune-DB-Instances bereit. Sie müssen keine Zertifikate anfordern. Die Zertifikate werden bereitgestellt, wenn Sie eine neue Instance erstellen.

Neptune weist den Instanzen in Ihrem Konto für jede Region ein einzelnes Wildcard-SSL-Zertifikat zu. AWS Das Zertifikat enthält Einträge für die Clusterendpunkte, die schreibgeschützten Clusterendpunkte und die Instance-Endpunkte.

Zertifikatdetails

Die folgenden Einträge sind im bereitgestellten Zertifikat enthalten:

- Cluster-Endpunkt – `*.cluster-`*`a1b2c3d4wxyz`*`.`*`region`*`.neptune.amazonaws.com`
- Read-only Endpunkt — `*.cluster-ro-`*`a1b2c3d4wxyz`*`.`*`region`*`.neptune.amazonaws.com`
- Instance-Endpunkte – `*.`*`a1b2c3d4wxyz`*`.`*`region`*`.neptune.amazonaws.com`

Nur die hier aufgelisteten Einträge werden unterstützt.

Proxyverbindungen

Die Zertifikate unterstützen nur die im vorherigen Abschnitt aufgeführten Hostnamen.

Wenn Sie einen Load Balancer oder einen Proxy-Server (z. B. HAProxy) verwenden, müssen Sie die SSL-Terminierung verwenden und ein eigenes SSL-Zertifikat auf dem Proxy-Server besitzen.

SSL-Passthrough funktioniert nicht, da die bereitgestellten SSL-Zertifikate nicht mit dem Hostnamen des Proxy-Servers übereinstimmen.

CA-Stammzertifikate

Die Zertifikate für Neptune-Instances werden normalerweise mittels des lokalen Vertrauensspeichers des Betriebssystems oder SDK (z. B. des Java-SDK) validiert.

Wenn Sie ein Stammzertifikat manuell bereitstellen müssen, können Sie das Amazon-CA-Stammzertifikat im PEM-Format aus dem Amazon Trust Services Policy Repository herunterladen.

Weitere Informationen

Weitere Informationen zum Herstellen von Verbindungen mit Neptune-Endpunkten über SSL siehe the section called “Installieren der Gremlin-Konsole” und the section called “HTTP REST”.

Neptun-Ressourcen im Ruhezustand verschlüsseln

Data-at-rest Verschlüsselung ist die Empfehlung. AWS Weitere Informationen finden Sie unter Data-at-Rest Data-in-Transit Verschlüsselung. Die Verschlüsselung wird in der AWS Konsole erzwungen, wenn Sie einen neuen Neptune DB-Cluster oder eine neue Neptune Global DB erstellen. Dies bietet eine zusätzliche Datenschutzebene. Es schützt Ihre Daten vor unbefugtem Zugriff auf den zugrunde liegenden Speicher und trägt dazu bei, die Compliance-Anforderungen für die Verschlüsselung von Daten im Ruhezustand zu erfüllen.

Um die Schlüssel zu verwalten, die zum Verschlüsseln und Entschlüsseln Ihrer Neptune-Ressourcen verwendet werden, verwenden Sie ().AWS Key Management ServiceAWS KMS AWS KMS kombiniert sichere, hochverfügbare Hardware und Software, um ein für die Cloud skaliertes Schlüsselverwaltungssystem bereitzustellen. Mithilfe AWS KMS können Sie Verschlüsselungsschlüssel erstellen und die Richtlinien definieren, die steuern, wie diese Schlüssel verwendet werden können. AWS KMS unterstützt AWS CloudTrail, sodass Sie die Verwendung von Schlüsseln überprüfen können, um sicherzustellen, dass die Schlüssel ordnungsgemäß verwendet werden.

Im Ruhezustand sind alle zugehörigen Protokolle, Backups und Snapshots für alle verschlüsselten Neptune-DB-Cluster verschlüsselt. Die Neptune-Verschlüsselung gilt nicht für Protokolle, die nach Amazon exportiert werden. CloudWatch

Verschlüsselung von Neptun-Ressourcen

Wenn Sie einen Neptune DB-Cluster oder eine Neptune Global DB erstellen, können Sie die AWS KMS Schlüssel-ID für Ihren Verschlüsselungsschlüssel angeben. Wenn Sie keine AWS KMS Schlüssel-ID angeben, verwendet Neptune Ihren standardmäßigen Amazon RDS-Verschlüsselungsschlüssel (`aws/rds`) in der Region. AWS KMS erstellt einen Standard-Verschlüsselungsschlüssel für jede Region in Ihrem AWS Konto. In einem Neptune Global-Cluster wird es so viele AWS KMS Schlüssel wie Regionen geben.

Nachdem Sie eine Neptune-Ressource erstellt haben, können Sie den Verschlüsselungsschlüssel für diese Ressource nicht ändern. Stellen Sie also sicher, dass Sie Ihre Anforderungen an den Verschlüsselungsschlüssel ermitteln, bevor Sie Ihre Neptune-Ressource erstellen. Wenn ein anderer

AWS KMS Schlüssel erforderlich ist, können Sie einen Snapshot des vorhandenen Neptune-DB-Clusters verwenden, um einen neuen mit einem anderen AWS KMS Schlüssel zu erstellen (siehe Wiederherstellen aus einem DB-Cluster-Snapshot).

Sie können den Amazon-Ressourcennamen (ARN) eines Schlüssels von einem anderen Konto verwenden, um eine Neptune-Ressource zu verschlüsseln. Wenn Sie eine Neptune-Ressource mit demselben AWS Konto erstellen, dem der AWS KMS Verschlüsselungsschlüssel gehört, kann die AWS KMS Schlüssel-ID, die Sie übergeben, der AWS KMS Schlüsselalias und nicht der ARN des Schlüssels sein.

> **Important**
>
> Wenn Neptune den Zugriff auf den Verschlüsselungsschlüssel für einen Neptune-DB-Cluster verliert — zum Beispiel, wenn Neptune den Zugriff auf einen Schlüssel entzogen hat —, wird der verschlüsselte Cluster in einen Terminalzustand versetzt und kann nur aus einem Backup wiederhergestellt werden. Wir empfehlen dringend, immer Backups für verschlüsselte Neptune-DB-Cluster zu aktivieren, um den Verlust verschlüsselter Daten in Ihren Datenbanken zu verhindern.

Wichtige Berechtigungen für die Aktivierung der Verschlüsselung

Der IAM-Benutzer oder die IAM-Rolle, die einen Neptune-DB-Cluster erstellt, muss mindestens die folgenden Berechtigungen für den KMS-Schlüssel haben:

- `"kms:Encrypt"`
- `"kms:Decrypt"`
- `"kms:GenerateDataKey"`
- `"kms:ReEncryptTo"`
- `"kms:GenerateDataKeyWithoutPlaintext"`
- `"kms:CreateGrant"`
- `"kms:ReEncryptFrom"`
- `"kms:DescribeKey"`

Hier ist ein Beispiel (für eine `us-east-1` Region) für eine Schlüsselrichtlinie, die die erforderlichen Berechtigungen beinhaltet:

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"
        }
      }
    }
  ]
}
```

- Die erste Anweisung in dieser Richtlinie ist optional. Sie gewährt Zugriff auf den Root-Prinzipal des Benutzers.
- Die zweite Anweisung bietet Zugriff auf alle erforderlichen AWS KMS APIs für diese Rolle, bis hin zum RDS Service Principal.
- Mit der dritten Anweisung wird die Sicherheit weiter verschärft, indem sie erzwingt, dass dieser Schlüssel von dieser Rolle für keinen anderen Dienst verwendet werden kann. AWS

Sie könnten die `createGrant`-Berechtigungen weiter einschränken, indem Sie Folgendes hinzufügen:

```
"Condition": {
  "Bool": {
    "kms:GrantIsForAWSResource": true
  }
}
```

Einschränkungen der Neptune-Verschlüsselung

Für Neptune Encryption gelten die folgenden Einschränkungen:

- Sie können einen unverschlüsselten Neptune-DB-Cluster nicht in einen verschlüsselten konvertieren. Sie können die Verschlüsselung für einen Neptune-DB-Cluster nur aktivieren, wenn er erstellt wird. Sie können jedoch einen unverschlüsselten Neptune-DB-Cluster-Snapshot auf einem verschlüsselten Neptune-DB-Cluster wiederherstellen. Geben Sie dazu bei der

Wiederherstellung aus dem unverschlüsselten Neptune-DB-Cluster-Snapshot einen KMS-Verschlüsselungsschlüssel an.

- Aus Kompatibilitätsgründen ist es weiterhin möglich, einen unverschlüsselten Neptune-DB-Cluster über die CLI und die SDKs zu erstellen. AWS Die Konsole ermöglicht nur die Erstellung verschlüsselter Neptune-DB-Cluster.
- Sie können verschlüsselte und unverschlüsselte Neptune-DB-Cluster nicht in derselben Neptune Global DB mischen. Entweder sind alle Cluster verschlüsselt oder alle Cluster sind unverschlüsselt. Dies wird in der Neptune Global DB-Konfiguration erzwungen.

Authentifizieren Sie Ihre Amazon Neptune Neptune-Datenbank mit AWS Identity and Access Management

AWS Identity and Access Management (IAM) hilft einem Administrator, den Zugriff auf AWS-Service Ressourcen sicher zu kontrollieren. AWS IAM-Administratoren steuern, wer für die Verwendung von Neptune-Ressourcen authentifiziert (angemeldet) und autorisiert (berechtigt) werden kann. IAM ist ein Programm AWS-Service , das Sie ohne zusätzliche Kosten nutzen können.

Sie können AWS Identity and Access Management (IAM) verwenden, um sich bei Ihrer Neptune-DB-Instance oder Ihrem DB-Cluster zu authentifizieren. Wenn die IAM-Datenbankauthentifizierung aktiviert ist, muss jede Anfrage mit Signature Version 4 signiert werden. AWS

AWS Signature Version 4 fügt Authentifizierungsinformationen zu AWS Anfragen hinzu.
Aus Sicherheitsgründen müssen alle Anfragen an Neptune-DB-Cluster mit aktivierter IAM-Authentifizierung mit einem Zugriffsschlüssel signiert werden. Dieser Schlüssel besteht aus einer Zugriffsschlüssel-ID und einem geheimen Zugriffsschlüssel. Die Authentifizierung wird extern über IAM-Richtlinien verwaltet.

Neptune authentifiziert sich bei der Verbindung und überprüft bei WebSockets Verbindungen regelmäßig die Berechtigungen, um sicherzustellen, dass der Benutzer weiterhin Zugriff hat.

> **Note**
>
> - Das Widerrufen, Löschen oder Rotieren von Anmeldeinformationen in Verbindung mit IAM-Benutzern wird nicht empfohlen, da dies keine bereits geöffneten Verbindungen beendet.

- Es gibt Beschränkungen für die Anzahl gleichzeitiger WebSocket Verbindungen pro Datenbankinstanz und für die Dauer, die eine Verbindung bestehen kann. Weitere Informationen finden Sie unter WebSockets Grenzwerte.

Die Nutzung von IAM ist von Ihrer Rolle abhängig

Die Art und Weise, wie Sie AWS Identity and Access Management (IAM) verwenden, hängt von der Arbeit ab, die Sie in Neptune ausführen.

Service-Benutzer – Wenn Sie den Neptune-Service zur Ausführung von Aufgaben verwenden, stellt Ihnen Ihr Administrator die nötigen Anmeldeinformationen und Berechtigungen für die Nutzung der Neptune-Datenebene bereit. Wenn Sie einen umfangreicheren Zugriff benötigen, kann Ihnen Wissen über die Zugriffsverwaltung helfen, die richtigen Berechtigungen bei Ihrem Administrator anzufordern.

Service-Administrator – Wenn Sie in Ihrem Unternehmen für Neptune-Ressourcen verantwortlich sind, haben Sie wahrscheinlich Zugriff auf Neptune-Verwaltungsaktionen über die Neptune-Verwaltungs-API. Möglicherweise ermitteln Sie auch, welche Neptune-Datenzugriffsaktionen und Ressourcen Service-Benutzer benötigen, um ihre Arbeit zu erledigen. Ein IAM-Administrator kann anschließend IAM-Richtlinien anwenden, um die Berechtigungen Ihrer Service-Benutzer zu ändern.

IAM-Administrator – Wenn Sie IAM-Administrator sind, müssen Sie IAM-Richtlinien für die Verwaltung und den Datenzugriff in Neptune verfassen. Beispiele für identitätsbasierte Neptune-Richtlinien, die Sie in verwenden können, finden Sie unter Verwenden verschiedener Arten von IAM-Richtlinien für die Steuerung des Zugriffs auf Neptune.

Authentifizieren mit Identitäten

Authentifizierung ist die Art und Weise, wie Sie sich AWS mit Ihren Identitätsdaten anmelden. Sie müssen sich als IAM-Benutzer authentifizieren oder eine IAM-Rolle annehmen. Root-Benutzer des AWS-Kontos

Sie können sich als föderierte Identität anmelden, indem Sie Anmeldeinformationen aus einer Identitätsquelle wie AWS IAM Identity Center (IAM Identity Center), Single Sign-On-Authentifizierung oder Anmeldeinformationen verwenden. Google/Facebook Weitere Informationen zum Anmelden finden Sie unter So melden Sie sich bei Ihrem AWS-Konto an im Benutzerhandbuch für AWS-Anmeldung .

AWS Bietet für den programmatischen Zugriff ein SDK und eine CLI zum kryptografischen Signieren von Anfragen. Weitere Informationen finden Sie unter AWS Signature Version 4 for API requests im IAM-Benutzerhandbuch.

AWS-Konto Root-Benutzer

Wenn Sie einen erstellen AWS-Konto, beginnen Sie mit einer Anmeldeidentität, dem sogenannten AWS-Konto Root-Benutzer, der vollständigen Zugriff auf alle AWS-Services Ressourcen hat. Wir raten ausdrücklich davon ab, den Root-Benutzer für Alltagsaufgaben zu verwenden. Eine Liste der Aufgaben, für die Sie sich als Root-Benutzer anmelden müssen, finden Sie unter Tasks that require root user credentials im IAM-Benutzerhandbuch.

IAM-Benutzer und -Gruppen

Ein IAM-Benutzer ist eine Identität mit bestimmten Berechtigungen für eine einzelne Person oder Anwendung. Wir empfehlen die Verwendung temporärer Anmeldeinformationen anstelle von IAM-Benutzern mit langfristigen Anmeldeinformationen. Weitere Informationen finden Sie im IAM-Benutzerhandbuch unter Erfordern, dass menschliche Benutzer für den Zugriff AWS mithilfe temporärer Anmeldeinformationen einen Verbund mit einem Identitätsanbieter verwenden müssen.

Eine IAM-Gruppe spezifiziert eine Sammlung von IAM-Benutzern und erleichtert die Verwaltung von Berechtigungen für große Gruppen von Benutzern. Weitere Informationen finden Sie unter Anwendungsfälle für IAM-Benutzer im IAM-Benutzerhandbuch.

IAM-Rollen

Eine IAM-Rolle ist eine Identität mit spezifischen Berechtigungen, die temporäre Anmeldeinformationen bereitstellt. Sie können eine Rolle übernehmen, indem Sie von einer Benutzer- zu einer IAM-Rolle (Konsole) wechseln AWS CLI oder einen AWS API-Vorgang aufrufen. Weitere Informationen finden Sie unter Methoden, um eine Rolle zu übernehmen im IAM-Benutzerhandbuch.

IAM-Rollen sind nützlich für den Verbundbenutzer-Zugriff, temporäre IAM-Benutzerberechtigungen, kontoübergreifenden Zugriff, serviceübergreifenden Zugriff und Anwendungen, die auf Amazon EC2 laufen. Weitere Informationen finden Sie unter Kontoübergreifender Ressourcenzugriff in IAM im IAM-Benutzerhandbuch.

Aktivierung der IAM-Datenbankauthentifizierung in Amazon Neptune

Standardmäßig ist die IAM-Datenbankauthentifizierung deaktiviert, wenn Sie einen Amazon–Neptune-DB-Cluster erstellen. Sie können die IAM-Datenbank-Authentifizierung mithilfe der AWS-Managementkonsole aktivieren (oder wieder deaktivieren).

Befolgen Sie zum Erstellen eines neuen Neptune-DB-Clusters mit IAM-Authentifizierung über die Konsole die Anweisungen für die Erstellung eines Neptune-DB-Clusters in Starten eines Neptune-DB-Clusters mit dem AWS-Managementkonsole.

Scrollen Sie während des Erstellungsvorgangs nach unten zu Zusätzliche Einstellungen und wählen Sie IAM-Authentifizierung aktivieren aus.

So aktivieren oder deaktivieren Sie die IAM-Authentifizierung für eine vorhandene DB-Instance oder einen vorhandenen DB-Cluster

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die Amazon Neptune Neptune-Konsole unter. https://console.aws.amazon.com/neptune/home
2. Klicken Sie im Navigationsbereich auf Cluster.
3. Wählen Sie den Neptune-DB-Cluster aus, den Sie ändern möchten, und wählen Sie dann Modify.
4. Scrollen Sie nach unten zu Zusätzliche Einstellungen. Wählen Sie für die IAM-DB-Authentifizierung entweder IAM-Authentifizierung aktivieren oder IAM-Authentifizierung ausschalten (zum Deaktivieren). Klicken Sie anschließend auf Weiter.
5. Wählen Sie unter Planung von Änderungen aus, wann die Änderung übernommen werden soll, und klicken Sie dann auf Senden.

> **Important**
>
> Wenn Sie die IAM-Datenbankauthentifizierung aktivieren oder deaktivieren, wird die DB-Engine neu gestartet. Dieser Neustart beendet alle bestehenden Verbindungen zum Cluster und führt zu einer kurzen Unterbrechung der Verfügbarkeit. Planen Sie, diese Änderung während eines Wartungsfensters oder zu einem Zeitpunkt vorzunehmen, zu dem die Auswirkungen einer Verbindungsunterbrechung minimal sind.

Herstellen einer Verbindung zu Ihrer Amazon Neptune Neptune-Datenbank mit AWS Identity and Access Management Authentifizierung

Für Amazon Neptune Neptune-Ressourcen, für die die IAM-DB-Authentifizierung aktiviert ist, müssen alle HTTP-Anfragen mit AWS Signature Version 4 signiert werden. Allgemeine Informationen zum Signieren von Anfragen mit AWS Signature Version 4 finden Sie unter Signieren AWS von API-Anfragen.

AWS Signature Version 4 ist der Prozess zum Hinzufügen von Authentifizierungsinformationen zu AWS Anfragen. Aus Sicherheitsgründen AWS müssen die meisten Anfragen mit einem Zugriffsschlüssel signiert werden, der aus einer Zugriffsschlüssel-ID und einem geheimen Zugriffsschlüssel besteht.

> **Note**
>
> Wenn Sie temporäre Anmeldeinformationen verwenden, laufen diese nach einem angegebenen Intervall ab, , einschließlich des Sitzungs-Tokens.
> Sie müssen Ihren Sitzungs-Token aktualisieren, wenn Sie neue Anmeldeinformationen anfragen. Weitere Informationen finden Sie unter Verwenden temporärer Sicherheitsanmeldedaten, um Zugriff auf AWS Ressourcen anzufordern.

> **Important**
>
> Für den Zugriff auf Neptune mit IAM-based Authentifizierung müssen Sie HTTP-Anfragen erstellen und die Anfragen selbst signieren.

So funktioniert Signature Version 4

1. Sie erstellen eine kanonische Anforderung.
2. Mit der kanonischen Anforderung und einigen anderen Informationen erstellen Sie eine zu signierende Zeichenfolge.
3. Sie verwenden Ihren AWS geheimen Zugriffsschlüssel, um einen Signaturschlüssel abzuleiten, und verwenden dann diesen Signaturschlüssel und die zu signierende Zeichenfolge, um eine Signatur zu erstellen.

4. Fügen Sie diese Signatur der HTTP-Anforderung in einem Header oder als Abfragezeichenfolgeparameter hinzu.

Wenn Neptune die Anforderung empfängt, werden die gleichen Schritte ausgeführt, die Sie zum Berechnen der Signatur durchgeführt haben. Neptune vergleicht dann die berechnete Signatur mit der Signatur, die Sie mit der Anforderung gesendet haben. Wenn die Signaturen übereinstimmen, wird die Anforderung verarbeitet. Wenn die Signaturen nicht übereinstimmen, wird die Anforderung abgelehnt.

Allgemeine Informationen zum Signieren von Anfragen mit AWS Signature Version 4 finden Sie unter Signaturprozess für Signature Version 4 in der. Allgemeine AWS-Referenz

Die folgenden Abschnitte enthalten Beispiele, die zeigen, wie signierte Anforderungen an die Gremlin- und SPARQL-Endpunkte einer Neptune-DB-Instance mit aktivierter IAM-Authentifizierung gesendet werden.

Themen

- Voraussetzungen für die Verbindung von Amazon Neptune Neptune-Datenbanken mithilfe der IAM-Authentifizierung
- Herstellen einer Verbindung zu Amazon Neptune Neptune-Datenbanken mithilfe der IAM-Authentifizierung über die Befehlszeile
- Herstellen einer Verbindung zu Amazon Neptune Neptune-Datenbanken mithilfe der IAM-Authentifizierung mit der Gremlin-Konsole
- Herstellen einer Verbindung zu Amazon Neptune Neptune-Datenbanken mithilfe von IAM mit Gremlin Java
- Herstellen einer Verbindung zu Amazon Neptune Neptune-Datenbanken mithilfe der IAM-Authentifizierung mit Java und SPARQL
- Herstellen einer Verbindung zu Amazon Neptune Neptune-Datenbanken mithilfe der IAM-Authentifizierung mit SPARQL und Node.js
- Herstellen einer Verbindung zu Amazon Neptune Neptune-Datenbanken mithilfe der IAM-Authentifizierung mit Python
- Herstellen einer Verbindung zu Amazon Neptune Neptune-Datenbanken mithilfe der IAM-Authentifizierung mit Gremlin Python
- Herstellen einer Verbindung zu Amazon Neptune Neptune-Datenbanken mithilfe der IAM-Authentifizierung mit Gremlin JavaScript

- Herstellen einer Verbindung zu Amazon Neptune Neptune-Datenbanken mithilfe der IAM-Authentifizierung mit Gremlin Go
- Herstellen einer Verbindung zu Amazon Neptune Neptune-Datenbanken mithilfe der IAM-Authentifizierung mit Gremlin .NET

Voraussetzungen für die Verbindung von Amazon Neptune Neptune-Datenbanken mithilfe der IAM-Authentifizierung

Im Folgenden finden Sie Anweisungen zum Installieren von Apache Maven und Java 8 auf einer Amazon-EC2-Instance. Dies ist für die Beispiele für die Authentifizierung mit Amazon Neptune Signature Version 4 erforderlich.

So installieren Sie Apache Maven und Java 8 auf Ihrer EC2-Instance

1. Stellen Sie eine Verbindung mit Ihrer Amazon–EC2-Instance über einen SSH-Client her.
2. Installieren Sie Apache Maven auf Ihrer EC2-Instance. Wenn Sie Amazon Linux 2023 (bevorzugt) verwenden, verwenden Sie:

   ```
   sudo dnf update -y
   sudo dnf install maven -y
   ```

   Wenn Sie Amazon Linux 2 verwenden, laden Sie die neueste Binärdatei herunter von 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. Die Gremlin-Bibliotheken erfordern Java 8. Geben Sie Folgendes ein, um Java 8 auf Ihrer EC2-Instance zu installieren.

   ```
   sudo yum install java-1.8.0-devel
   ```

4. Geben Sie Folgendes ein, um Java 8 als Standard-Laufzeit Ihrer EC2-Instance festzulegen.

```
sudo /usr/sbin/alternatives --config java
```

Wenn Sie dazu aufgefordert werden, geben Sie die Nummer für Java 8 ein.

5. Geben Sie Folgendes ein, um Java 8 als Standard-Compiler Ihrer EC2-Instance festzulegen.

```
sudo /usr/sbin/alternatives --config javac
```

Wenn Sie dazu aufgefordert werden, geben Sie die Nummer für Java 8 ein.

Herstellen einer Verbindung zu Amazon Neptune Neptune-Datenbanken mithilfe der IAM-Authentifizierung über die Befehlszeile

Die Verwendung eines Befehlszeilentools zum Senden von Abfragen an Ihren Neptune-DB-Cluster ist von Vorteil, wie in vielen Beispielen in dieser Dokumentation gezeigt wird. Das Tool curl ist eine hervorragende Option für die Kommunikation mit Neptune-Endpunkten, wenn die IAM-Authentifizierung nicht aktiviert ist.

Um Ihre Daten zu schützen, sollte jedoch die IAM-Authentifizierung aktiviert werden.

Wenn die IAM-Authentifizierung aktiviert ist, muss jede Anforderung mit Signature Version 4 (Sig4) signiert werden. Das externe Befehlszeilen-Tool awscurl verwendet dieselbe Syntax wie `curl` und kann Abfragen mithilfe der Sig4-Signierung signieren. Im folgenden Abschnitt zu Verwenden von `awscurl` wird die sichere Verwendung von `awscurl` mit temporären Anmeldeinformationen beschrieben.

Einrichten eines Befehlszeilen-Tools für die Verwendung von HTTPS

Neptune setzt voraus, dass alle Verbindungen HTTPS verwenden. Jedes Befehlszeilen-Tool wie `curl` oder `awscurl` benötigt Zugriff auf entsprechende Zertifikate, um HTTPS verwenden zu können. Solange `curl` oder `awscurl` die entsprechenden Zertifikate finden kann, werden HTTPS-Verbindungen wie HTTP-Verbindungen behandelt, ohne dass zusätzliche Parameter benötigt werden. Die Beispiele in dieser Dokumentation basieren auf diesem Szenario.

Informationen zum Abruf dieser Zertifikate und ihrer korrekten Formatierung als CA-Speicher, den `curl` verwenden kann, finden Sie unter SSL-Zertifikat-Verifizierung in der `curl`-Dokumentation.

Sie können den Speicherort des CA-Zertifikatspeichers anschließend mittels der `CURL_CA_BUNDLE`-Umgebungsvariable speichern. Unter Windows sucht `curl` automatisch in einer Datei mit dem Namen `curl-ca-bundle.crt` danach. Es sucht zunächst in demselben Verzeichnis wie `curl.exe` und anschließend an anderer Stelle im Pfad. Weitere Informationen finden Sie unter SSL Certificate Verification.

Verwenden von **`awscurl`** mit temporären Anmeldeinformationen, um eine sichere Verbindung zu einem DB-Cluster mit aktivierter IAM-Authentifizierung herzustellen

Das Tool awscurl verwendet dieselbe Syntax wie`curl`, benötigt jedoch auch zusätzliche Informationen:

- **`--access_key`** – Gültiger Zugriffsschlüssel. Wenn nicht mit diesem Parameter angegeben, muss er in der Umgebungsvariablen `AWS_ACCESS_KEY_ID` oder in einer Konfigurationsdatei angegeben werden.
- **`--secret_key`** – Gültiger Geheimschlüssel, der dem Zugriffsschlüssel entspricht Wenn nicht mit diesem Parameter angegeben, muss er in der Umgebungsvariablen `AWS_SECRET_ACCESS_KEY` oder in einer Konfigurationsdatei angegeben werden.
- **`--security_token`** – Gültiges Sitzungstoken. Wenn nicht mit diesem Parameter angegeben, muss er in der Umgebungsvariablen `AWS_SECURITY_TOKEN` oder in einer Konfigurationsdatei angegeben werden.

In der Vergangenheit war es üblich, persistente Anmeldeinformationen mit `awscurl` zu verwenden, z. B. IAM-Benutzeranmeldeinformationen oder sogar Root-Anmeldeinformationen. Dies wird jedoch nicht empfohlen. Generieren Sie stattdessen temporäre Anmeldeinformationen mit einer der AWS - Security-Token-Service (STS)-APIs oder einem ihrer AWS CLI Wrapper.

Sie sollten die Werte `AccessKeyId`, `SecretAccessKey` und `SessionToken`, die vom STS-Aufruf zurückgegeben werden, in entsprechenden Umgebungsvariablen in Ihrer Shell-Sitzung platzieren und nicht in einer Konfigurationsdatei. Wenn die Shell beendet wird, werden die Anmeldeinformationen automatisch verworfen, anders als bei einer Konfigurationsdatei. Sie sollten auch keine längere Dauer für die temporären Anmeldeinformationen anfordern als wahrscheinlich benötigt.

Das folgende Beispiel zeigt die Schritte, die Sie in einer Linux-Shell ausführen könnten, um mittels sts assume-role temporäre, für eine halbe Stunde gültige Anmeldeinformationen zu erhalten und sie dann in Umgebungsvariablen zu platzieren, in denen `awscurl` sie finden kann:

```
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
```

Sie könnten dann `awscurl` verwenden, um eine signierte Anforderung ähnlich dieser an Ihren DB-Cluster zu senden:

```
awscurl (your cluster endpoint):8182/status \
    --region us-east-1 \
    --service neptune-db
```

Herstellen einer Verbindung zu Amazon Neptune Neptune-Datenbanken mithilfe der IAM-Authentifizierung mit der Gremlin-Konsole

Um über die Gremlin-Konsole mit Signature Version 4-Authentifizierung eine Verbindung zu Amazon Neptune herzustellen, schließen Sie einen Sigv4-Signaturer `requestInterceptor()` an die durch den Befehl hergestellte Verbindung an. `:remote` Dazu müssen Sie das `Cluster` Objekt manuell konfigurieren und es dann an den Befehl übergeben. `:remote`

Beachten Sie, dass sich dies deutlich von der typischen Situation unterscheidet, in der für den Befehl `:remote` eine Konfigurationsdatei benötigt wird, um die Verbindung herzustellen. Der Ansatz mit einer Konfigurationsdatei funktioniert nicht, da `requestInterceptor()` programmgesteuert festgelegt werden muss und daher die Konfiguration nicht aus einer Datei laden kann.

> **Note**
>
> Die folgenden Beispiele verwenden`requestInterceptor()`, was in TinkerPop 3.6.6 eingeführt wurde. Wenn Sie eine frühere TinkerPop Version als 3.6.6 (aber 3.5.5 oder höher) verwenden, verwenden Sie `handshakeInterceptor()` statt `requestInterceptor()` in den folgenden Codebeispielen.

Die folgenden Voraussetzungen sind erforderlich:

- Sie müssen die nötigen IAM-Anmeldeinformationen zum Signieren der Anforderungen besitzen. Weitere Informationen finden Sie unter Verwenden der standardmäßigen Anbieterkette für Anmeldeinformationen im AWS SDK für Java Entwicklerhandbuch.
- Sie müssen eine Version der Gremlin-Konsole installieren, die mit der Version der Neptune-Engine kompatibel ist, die von Ihrem DB-Cluster verwendet wird.

Wenn Sie temporäre Anmeldeinformationen verwenden, laufen diese nach einem bestimmten Intervall ab, ebenso wie das Sitzungstoken. Sie müssen Ihr Sitzungstoken daher aktualisieren, wenn Sie neue Anmeldeinformationen anfordern. Weitere Informationen finden Sie im IAM-Benutzerhandbuch unter Verwenden temporärer Sicherheitsanmeldedaten, um Zugriff auf AWS Ressourcen anzufordern.

Hilfe beim Herstellen einer Verbindung mit SSL/TLS finden Sie unterSSL/TLS Konfiguration.

Connect die Gremlin-Konsole mit Sig4-Signatur

1. Starten Sie die Gremlin-Konsole:

   ```
   $ bin/gremlin.sh
   ```

2. Installieren Sie bei der Eingabeaufforderung `gremlin>` die `amazon-neptune-sigv4-signer`-Bibliothek. (Sie müssen dies nur einmal für die Konsole ausführen):

   ```
   :install com.amazonaws amazon-neptune-sigv4-signer 2.4.0
   ```

   Wenn Sie bei diesem Schritt auf Probleme stoßen, kann es hilfreich sein, die TinkerPop Dokumentation zur Grape-Konfiguration zu lesen.

   > **Note**
   >
   > Wenn Sie einen HTTP-Proxy verwenden, können bei diesem Schritt Fehler auftreten, durch die der `:install`-Befehl nicht abgeschlossen wird. Führen Sie zur Problembehebung die folgenden Befehle aus, um der Konsole den Proxy mitzuteilen:
   >
   > ```
   > System.setProperty("https.proxyHost", "(the proxy IP address)")
   > ```

```
System.setProperty("https.proxyPort", "(the proxy port)")
```

3. Importieren Sie die Klasse, die für die Verarbeitung der Signierung erforderlich ist, in `requestInterceptor()`:

```
:import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider
:import com.amazonaws.neptune.auth.NeptuneNettyHttpSigV4Signer
```

4. Wenn Sie temporäre Anmeldeinformationen verwenden, müssen Sie auch Ihr Sitzungstoken wie folgt angeben:

```
System.setProperty("aws.sessionToken","(your session token)")
```

5. Wenn Sie Ihre Kontoanmeldeinformationen nicht anderweitig eingerichtet haben, können Sie sie wie folgt zuweisen:

```
System.setProperty("aws.accessKeyId","(your access key)")
System.setProperty("aws.secretKey","(your secret key)")
```

6. Konstruieren Sie das `Cluster`-Objekt manuell, um eine Verbindung zu Neptune herzustellen:

```
cluster = Cluster.build("(host name)")  \
                 .enableSsl(true) \
                 .requestInterceptor { r ->  \
                   def sigV4Signer = new NeptuneNettyHttpSigV4Signer("(Amazon
 region)", \
                                      DefaultCredentialsProvider.create()); \
                   sigV4Signer.signRequest(r); \
                   return r; } \
                 .create()
```

   Hilfe bei der Suche nach dem Hostnamen einer Neptune-DB-Instance finden Sie unter Verbinden mit Amazo-Neptune-Endpunkten.

7. Stellen Sie die `:remote`-Verbindung mithilfe des Variablennamens des `Cluster`-Objekts im vorherigen Schritt her:

```
:remote connect tinkerpop.server cluster
```

8. Geben Sie den folgenden Befehl ein, um zum Remote-Modus zu wechseln. Dies sendet alle Gremlin-Abfragen an die Remote-Verbindung.

```
:remote console
```

Herstellen einer Verbindung zu Amazon Neptune Neptune-Datenbanken mithilfe von IAM mit Gremlin Java

Hier ist ein Beispiel dafür, wie Sie mithilfe der Gremlin-Java-API mit Sig4-Signatur eine Verbindung zu Neptune herstellen können (es werden allgemeine Kenntnisse über die Verwendung von Maven vorausgesetzt). In diesem Beispiel wird die Amazon Neptune SigV4 Signer-Bibliothek verwendet, um das Signieren von Anfragen zu unterstützen. Definieren Sie zunächst die Abhängigkeiten als Teil der Datei `pom.xml`:

> **Note**
>
> Die folgenden Beispiele verwenden`requestInterceptor()`, was in 3.6.6 eingeführt wurde. TinkerPop Wenn Sie eine frühere TinkerPop Version als 3.6.6 (aber 3.5.5 oder höher) verwenden, verwenden Sie `handshakeInterceptor()` statt `requestInterceptor()` in den folgenden Codebeispielen.

```
<dependency>
  <groupId>com.amazonaws</groupId>
  <artifactId>amazon-neptune-sigv4-signer</artifactId>
  <version>3.1.0</version>
</dependency>
```

Der Amazon Neptune SigV4 Signer unterstützt beide Versionen 1.x und 2.x des Java SDK. AWS In den folgenden Beispielen wird 2.x verwendet, wobei es sich um eine Instanz handelt. `DefaultCredentialsProvider software.amazon.awssdk.auth.credentials.AwsCredentialsProvider` Wenn Sie von 1.x auf 2.x aktualisieren, lesen Sie die Änderungen des Credentials-Providers in der Dokumentation zum 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);
}
```

Kontenübergreifende IAM-Authentifizierung

Amazon Neptune unterstützt die kontoübergreifende IAM-Authentifizierung mithilfe von Rollenübernahme, manchmal auch als Rollenverkettung bezeichnet. So gewähren Sie Zugriff auf einen Neptune-Cluster von einer Anwendung aus, die in einem anderen AWS Konto gehostet wird:

- Erstellen Sie einen neuen IAM-Benutzer oder eine neue IAM-Rolle im AWS Anwendungskonto mit einer Vertrauensrichtlinie, die es dem Benutzer oder der Rolle ermöglicht, eine andere IAM-Rolle anzunehmen. Weisen Sie diese Rolle dem Computer zu, der die Anwendung hostet (EC2-Instance, Lambda-Funktion, ECS-Task usw.).

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "assumeRolePolicy",
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Resource": "arn:aws:iam::111122223333:role/role-name"
    }
  ]
}
```

- Erstellen Sie eine neue IAM-Rolle im AWS Neptune-Datenbankkonto, die den Zugriff auf die Neptune-Datenbank und die Rollenübernahme vom Anwendungskonto IAM ermöglicht. user/role Verwenden Sie eine Vertrauensrichtlinie von:

  JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "(ARN of application account IAM user or role)"
                ]
            },
            "Action": "sts:AssumeRole",
            "Condition": {}
        }
    ]
}
```

- Verwenden Sie das folgende Codebeispiel als Anleitung zur Verwendung dieser beiden Rollen, um der Anwendung den Zugriff auf Neptune zu ermöglichen. In diesem Beispiel wird die Rolle des Anwendungskontos DefaultCredentialProviderChainbeim Erstellen des übernommen. `STSclient` Das `STSclient` wird dann über das

verwendetSTSAssumeRoleSessionCredentialsProvider, um die im AWS Neptune-Datenbankkonto gehostete Rolle zu übernehmen.

```
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();
 }
```

Herstellen einer Verbindung zu Amazon Neptune Neptune-Datenbanken mithilfe der IAM-Authentifizierung mit Java und SPARQL

Dieser Abschnitt zeigt, wie Sie eine Verbindung mit Neptune über RDF4J oder Apache Jena mit Signature-Version-4-Authentifizierung herstellen.

Voraussetzungen

- Java 8 oder höher.
- Apache Maven 3.3 oder höher.

  Weitere Informationen zum Installieren dieser Voraussetzungen auf einer EC2-Instance mit Amazon Linux finden Sie unter Voraussetzungen für die Verbindung von Amazon Neptune Neptune-Datenbanken mithilfe der IAM-Authentifizierung.
- IAM-Anmeldeinformationen zum Signieren der Anfragen. Weitere Informationen finden Sie unter Verwenden der Standard-Anbieterkette für Anmeldeinformationen im AWS SDK für Java - Entwicklerhandbuch.

> **Note**
>
> Wenn Sie temporäre Anmeldeinformationen verwenden, laufen diese nach einem angegebenen Intervall ab, , einschließlich des Sitzungs-Tokens.
> Sie müssen Ihren Sitzungs-Token aktualisieren, wenn Sie neue Anmeldeinformationen anfragen. Weitere Informationen finden Sie im IAM-Benutzerhandbuch unter Verwenden temporärer Sicherheitsanmeldedaten zur Anforderung des Zugriffs auf AWS Ressourcen.

- Wählen Sie für die Variable `SERVICE_REGION` eine der folgenden Einstellungen aus, um die Region Ihrer Neptune-DB-Instance anzugeben:
  - USA Ost (Nord-Virginia): `us-east-1`
  - USA Ost (Ohio): `us-east-2`
  - USA West (Nordkalifornien): `us-west-1`
  - USA West (Oregon): `us-west-2`
  - Kanada (Zentral): `ca-central-1`
  - Kanada West (Calgary): `ca-west-1`
  - Südamerika (São Paulo): `sa-east-1`
  - Europa (Stockholm): `eu-north-1`
  - Europa (Spanien): `eu-south-2`
  - Europa (Irland): `eu-west-1`
  - Europa (London): `eu-west-2`
  - Europa (Paris): `eu-west-3`
  - Europa (Frankfurt): `eu-central-1`
  - Naher Osten (Bahrain): `me-south-1`
  - Naher Osten (VAE): `me-central-1`
  - Israel (Tel Aviv): `il-central-1`
  - Afrika (Kapstadt): `af-south-1`
  - Asien-Pazifik (Hongkong): `ap-east-1`
  - Asien-Pazifik (Tokio): `ap-northeast-1`
  - Asien-Pazifik (Seoul): `ap-northeast-2`
  - Asien-Pazifik (Osaka): `ap-northeast-3`

- Asien-Pazifik (Singapur): `ap-southeast-1`
- Asien-Pazifik (Sydney): `ap-southeast-2`
- Asien-Pazifik (Jakarta): `ap-southeast-3`
- Asien-Pazifik (Melbourne): `ap-southeast-4`
- Asien-Pazifik (Malaysia): `ap-southeast-5`
- Asien-Pazifik (Mumbai): `ap-south-1`
- Asien-Pazifik (Hyderabad): `ap-south-2`
- China (Peking): `cn-north-1`
- China (Ningxia): `cn-northwest-1`
- AWS GovCloud (US-West): `us-gov-west-1`
- AWS GovCloud (US-East): `us-gov-east-1`

Herstellen von Verbindungen mit RDF4J oder Apache Jena mittels Signature-Version-4-Signierung

1. Klonen Sie das Beispiel-Repository von GitHub.

   ```
   git clone https://github.com/aws/amazon-neptune-sparql-java-sigv4.git
   ```

2. Wechseln Sie in das geklonte Verzeichnis.

   ```
   cd amazon-neptune-sparql-java-sigv4
   ```

3. Überprüfen Sie den Branch mit dem neuesten Tag auf die neueste Version des Projekts.

   ```
   git checkout $(git describe --tags `git rev-list --tags --max-count=1`)
   ```

4. Geben Sie einen der folgenden Befehle ein, um den Beispielcode zu kompilieren und auszuführen.

   *`your-neptune-endpoint`*Ersetzen Sie durch den Hostnamen oder die IP-Adresse Ihrer Neptune-DB-Instance. Der Standard-Port ist 8182.

   > **Note**
   >
   > Informationen zum Ermitteln des Hostnamens Ihrer Neptune-DB-Instance finden Sie im Abschnitt Verbinden mit Amazo-Neptune-Endpunkten.

Eclipse RDF4J

Geben Sie Folgendes ein, um das RDF4J-Beispiel auszuführen.

```
mvn compile exec:java \
    -Dexec.mainClass="com.amazonaws.neptune.client.rdf4j.NeptuneRdf4JSigV4Example"
 \
    -Dexec.args="https://your-neptune-endpoint:portsparql region-name"
```

Apache Jena

Geben Sie Folgendes ein, um das Apache Jena-Beispiel auszuführen.

```
mvn compile exec:java \
    -Dexec.mainClass="com.amazonaws.neptune.client.jena.NeptuneJenaSigV4Example" \
    -Dexec.args="https://your-neptune-endpoint:port"
```

5. Informationen zum Anzeigen des Quellcodes für dieses Beispiel finden Sie in den Beispielen im `src/main/java/com/amazonaws/neptune/client/`-Verzeichnis.

Fügen Sie dem Abschnitt `<dependencies>` Ihrer `pom.xml` das `amazon-neptune-sigv4-signer` Maven-Paket hinzu, um den SigV4-Signaturtreiber in Ihrer eigenen Java-Anwendung zu verwenden. Es wird empfohlen, die Beispiele als Ausgangspunkt zu verwenden.

Herstellen einer Verbindung zu Amazon Neptune Neptune-Datenbanken mithilfe der IAM-Authentifizierung mit SPARQL und Node.js

Abfragen mithilfe der Signature V4-Signatur und der AWS SDK für Javascript V3

Hier ist ein Beispiel dafür, wie Sie mithilfe der Authentifizierung Node.js mit Signature Version 4 und dem AWS SDK für Javascript V3 eine Verbindung zu Neptune SPARQL herstellen können:

```
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);
    }
  );
}
```

Abfragen mit Signature V4-Signatur und AWS SDK für Javascript V2

Hier ist ein Beispiel dafür, wie Sie mithilfe der Authentifizierung Node.js mit Signature Version 4 und dem AWS SDK für Javascript V2 eine Verbindung zu Neptune SPARQL herstellen können:

```
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);
    });
}
```

Herstellen einer Verbindung zu Amazon Neptune Neptune-Datenbanken mithilfe der IAM-Authentifizierung mit Python

Der `boto3` `neptunedata` Client bietet die einfachste Möglichkeit, von Python aus eine Verbindung zu einer IAM-enabled Neptune-Datenbank herzustellen. Der Client verarbeitet das Signieren mit Signature Version 4 automatisch, sodass Sie Anfragen nicht selbst signieren müssen.

Voraussetzungen

- Python 3.x

- Die `boto3` Bibliothek: `pip install boto3`
- AWS Anmeldeinformationen, die mit einer beliebigen Standardmethode konfiguriert wurden (Umgebungsvariablen, AWS Konfigurationsdatei, Instanzprofil oder Lambda-Ausführungsrolle)
- Eine IAM-Richtlinie, die `neptune-db:*` Aktionen auf Ihrem Neptune-Cluster ermöglicht

Verbindung mit Standardanmeldedaten herstellen

Dieser Ansatz funktioniert, wenn Anmeldeinformationen automatisch aufgelöst werden `boto3` können, z. B. auf Amazon Elastic Compute Cloud-Instances mit Instanzprofilen, in AWS Lambda Funktionen oder mit konfigurierten AWS Profilen.

Ersetzen Sie die Endpunkt-URL durch Ihren Neptune-Cluster-Endpunkt.

Gremlin und 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

Der `boto3` `neptunedata` Client unterstützt SPARQL derzeit nicht. Um SPARQL-Abfragen an eine IAM-enabled Neptune-Datenbank zu senden, signieren Sie Anfragen manuell, indem Sie Signature Version 4 mit den Dienstprogrammen zur `botocore` Anforderungssignierung und der Bibliothek () verwenden. `requests` `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**
>
> In diesem Beispiel `botocore.session.Session` werden AWS Anmeldeinformationen automatisch aus Umgebungsvariablen, AWS Konfigurationsdateien, Instanzprofilen oder Lambda-Ausführungsrollen aufgelöst. Sie müssen die Anmeldeinformationen nicht explizit festlegen.

Verwenden temporärer Anmeldeinformationen

Verwenden Sie für kontoübergreifenden Zugriff, zeitlich begrenzte Sitzungen oder als bewährte Sicherheitsmethode zur Vermeidung langlebiger Anmeldeinformationen, temporäre Anmeldeinformationen abzurufen und eine AWS STS zu erstellen. `boto3.Session`

Gremlin und 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**
>
> Temporäre Anmeldeinformationen laufen nach einem bestimmten Intervall ab (die Standardeinstellung ist 1 Stunde) und werden vom Client nicht automatisch aktualisiert. Verwenden Sie für Anwendungen mit langer Laufzeit stattdessen profilbasierte Anmeldeinformationen oder Instanzprofile.

SPARQL

Da der `boto3` `neptunedata` Client SPARQL derzeit nicht unterstützt, müssen Sie Anfragen manuell mit den temporären Anmeldeinformationen signieren.

```
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())
```

Verwenden mit AWS Lambda

In Lambda stellt die Ausführungsrolle Anmeldeinformationen automatisch bereit. `boto3` Das Beispiel für Standardanmeldedaten funktioniert ohne Änderung.

Ein vollständiges Lambda-Beispiel mit Verbindungsmanagement und Wiederholungslogik finden Sie unter. the section called “Beispiele für Lambda-Funktionen”

Herstellen einer Verbindung zu Amazon Neptune Neptune-Datenbanken mithilfe der IAM-Authentifizierung mit Gremlin Python

-Übersicht

Dieses Handbuch zeigt, wie Sie mithilfe des Gremlin Python-Treibers, mit Signature Version 4-Authentifizierung und dem AWS SDK für Python (Boto3) eine Verbindung zu einer Amazon Neptune Neptune-Datenbank mit aktivierter IAM-Authentifizierung herstellen.

Erstellen Sie eine grundlegende Verbindung

Verwenden Sie das folgende Codebeispiel als Anleitung zum Herstellen einer Basisverbindung mit der IAM-Authentifizierung mithilfe des Gremlin Python-Treibers.

```
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()
```

Herstellen einer Verbindung zu Amazon Neptune Neptune-Datenbanken mithilfe der IAM-Authentifizierung mit Gremlin JavaScript

-Übersicht

In diesem Handbuch wird gezeigt, wie Sie mithilfe des JavaScript Gremlin-Treibers, mit Signature Version 4-Authentifizierung und dem SDK für Javascript v3 eine Verbindung zu einer Amazon Neptune Neptune-Datenbank mit aktivierter IAM-Authentifizierung herstellen. AWS

Voraussetzungen

- Ein Amazon Neptune Neptune-Cluster mit aktivierter IAM-Authentifizierung.
- Knoten 13 oder höher (siehe Minimalversionen für Gremlin JavaScript und AWS SDK für Javascript v3).
- AWS konfigurierte Anmeldeinformationen (über Umgebungsvariablen, Datei mit gemeinsamen Anmeldeinformationen oder IAM-Rolle).

Erstellen Sie eine Basisverbindung

Verwenden Sie das folgende Codebeispiel als Anleitung zum Herstellen einer Basisverbindung mit IAM-Authentifizierung mithilfe des JavaScript Gremlin-Treibers.

```
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();
```

Herstellen einer Verbindung zu Amazon Neptune Neptune-Datenbanken mithilfe der IAM-Authentifizierung mit Gremlin Go

-Übersicht

Dieses Handbuch zeigt, wie Sie mithilfe des Gremlin Go-Treibers, mit Signature Version 4-Authentifizierung und dem SDK für GO v2 eine Verbindung zu einer Amazon Neptune Neptune-Datenbank mit aktivierter IAM-Authentifizierung herstellen. AWS

Voraussetzungen

- Ein Amazon Neptune Neptune-Cluster mit aktivierter IAM-Authentifizierung.
- Go 1.22 oder höher (siehe die unterstützten Mindestversionen für Gremlin Go und AWS SDK for Go v2).

- AWS konfigurierte Anmeldeinformationen (über Umgebungsvariablen, Datei mit gemeinsamen Anmeldeinformationen oder IAM-Rolle)

Erstellen Sie eine Basisverbindung

Verwenden Sie das folgende Codebeispiel als Anleitung zum Herstellen einer Basisverbindung mit IAM-Authentifizierung mithilfe des Gremlin Go-Treibers.

```
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)
}
```

Dynamische Aktualisierung der Gremlin Go-Anmeldeinformationen

Gremlin Go ermöglicht DynamicAuth die Injektion eines Funktionszeigers zum Abrufen von Anmeldeinformationen und Generieren des Headers, wodurch verhindert wird, dass der Header bei lang andauernden Verbindungen abläuft.

```
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)
}
```

Herstellen einer Verbindung zu Amazon Neptune Neptune-Datenbanken mithilfe der IAM-Authentifizierung mit Gremlin .NET

-Übersicht

Dieses Handbuch zeigt, wie Sie mithilfe des Gremlin.NET-Treibers, mit Signature Version 4-Authentifizierung und dem SDK for .NET v3 eine Verbindung zu einer Amazon Neptune Neptune-Datenbank mit aktivierter IAM-Authentifizierung herstellen. AWS

Erstellen Sie eine Basisverbindung

Um eine Verbindung mit Gremlin .NET herzustellen, verwenden Sie die `SigV4RequestSigner` Quelldatei aus der benutzerdefinierten Bibliothek. https://github.com/aws/amazon-neptune-gremlin-dotnet-sigv4 Ein Beispiel für die Einrichtung eines Projekts befindet sich in und wird https://github.com/aws/amazon-neptune-gremlin-dotnet-sigv4/tree/main/exampleunten wiedergegeben.

Projektdatei:

```
<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>
```

Beispielprogramm:

```
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);
            }
        }
```

```
    }
}
```

Verwaltung des Zugriffs auf Amazon Neptune Neptune-Datenbanken mithilfe von IAM-Richtlinien

IAM-Richtlinien sind JSON-Objekte, die Berechtigungen für die Verwendung von Aktionen und Ressourcen definieren.

Sie kontrollieren den Zugriff, AWS indem Sie Richtlinien erstellen und diese an Identitäten oder Ressourcen anhängen. AWS Eine Richtlinie definiert Berechtigungen, wenn sie mit einer Identität oder Ressource verknüpft sind. AWS bewertet diese Richtlinien, wenn ein Principal eine Anfrage stellt. Die meisten Richtlinien werden AWS als JSON-Dokumente gespeichert. Weitere Informationen zu JSON-Richtliniendokumenten finden Sie unter Übersicht über JSON-Richtlinien im IAM-Benutzerhandbuch.

Mit Hilfe von Richtlinien legen Administratoren fest, wer Zugriff auf was hat, indem sie definieren, welches Prinzipal welche Aktionen auf welchen Ressourcenund unter welchen Bedingungendurchführen darf.

Standardmäßig haben Benutzer, Gruppen und Rollen keine Berechtigungen. Ein IAM-Administrator erstellt IAM-Richtlinien und fügt sie zu Rollen hinzu, die die Benutzer dann übernehmen können. IAM-Richtlinien definieren Berechtigungen unabhängig von der Methode, die zur Ausführung der Operation verwendet wird.

Identity-Based Richtlinien

Identity-based Richtlinien sind Richtliniendokumente für JSON-Berechtigungen, die Sie an eine Identität (Benutzer, Gruppe oder Rolle) anhängen. Diese Richtlinien steuern, welche Aktionen Identitäten für welche Ressourcen und unter welchen Bedingungen ausführen können. Informationen zum Erstellen identitätsbasierter Richtlinien finden Sie unter Definieren benutzerdefinierter IAM-Berechtigungen mit vom Kunden verwalteten Richtlinien im IAM-Benutzerhandbuch.

Identity-based Richtlinien können Inline-Richtlinien (direkt in eine einzelne Identität eingebettet) oder verwaltete Richtlinien (eigenständige Richtlinien, die mehreren Identitäten zugeordnet sind) sein. Informationen dazu, wie Sie zwischen verwalteten und Inline-Richtlinien wählen, finden Sie unter Choose between managed policies and inline policies im IAM-Benutzerhandbuch.

Verwenden von Service Control Policies (SCP) mit AWS Organisationen

Service Control Policies (SCPs) sind JSON-Richtlinien, die die maximalen Berechtigungen für eine Organisation oder Organisationseinheit (OU) in festlegen. AWS Organizations AWS Organizations ist ein Dienst zur Gruppierung und zentralen Verwaltung mehrerer AWS Konten, die Ihrem Unternehmen gehören. Wenn Sie innerhalb einer Organisation alle Features aktivieren, können Sie Service-Kontrollrichtlinien (SCPs) auf alle oder einzelne Ihrer Konten anwenden. Das SCP schränkt die Berechtigungen für Entitäten in Mitgliedskonten ein, einschließlich der einzelnen AWS Root-Benutzer. Weitere Informationen zu Organizations und SCPs finden Sie unter So funktionieren SCPs im AWS Organizations Benutzerhandbuch.

Kunden, die Amazon Neptune in einem AWS Konto innerhalb eines AWS Unternehmens einsetzen, können mithilfe von SCPs kontrollieren, welche Konten Neptune verwenden können. Um den Zugriff auf Neptune innerhalb eines Mitgliedskontos zu gewährleisten, musst du Folgendes tun:

- Erlaubt den Zugriff auf `rds:*` und `neptune-db:*` für Neptune-Datenbankoperationen. Weitere Informationen finden Sie unter Warum sind Amazon RDS-Berechtigungen und Ressourcen für die Verwendung von Neptune Database erforderlich? für Einzelheiten darüber, warum Amazon RDS-Berechtigungen für die Neptune-Datenbank erforderlich sind.
- Erlauben Sie den Zugriff `neptune-graph:*` auf Neptune Analytics-Operationen.

Für die Verwendung der Amazon-Neptune-Konsole erforderliche Berechtigungen

Damit Benutzer mit der Amazon-Neptune-Konsole arbeiten können, müssen sie einen Mindestsatz von Berechtigungen besitzen. Diese Berechtigungen ermöglichen Benutzern, die Neptune-Ressourcen für ihr AWS -Konto zu beschreiben und weitere verwandte Informationen bereitzustellen, einschließlich Informationen in den Bereichen Amazon-EC2-Sicherheit und -Netzwerk.

Wenn Sie eine IAM-Richtlinie erstellen, die strenger ist als die mindestens erforderlichen Berechtigungen, funktioniert die Konsole nicht wie vorgesehen für Benutzer mit dieser IAM-Richtlinie. Um sicherzustellen, dass diese Benutzer die Neptune-Konsole weiter verwenden können, müssen Sie ihnen die verwaltete Richtlinie `NeptuneReadOnlyAccess` anfügen, wie in Verwenden AWS verwaltete Richtlinien für den Zugriff auf Amazon Neptune Neptune-Datenbanken beschrieben.

Sie müssen Benutzern, die nur die Amazon Neptune-API AWS CLI oder die Amazon Neptune Neptune-API aufrufen, keine Mindestberechtigungen für die Konsole gewähren.

Anfügen einer IAM-Richtlinie an einen IAM-Benutzer

Um eine verwaltete oder benutzerdefinierte Richtlinie anzuwenden, fügen Sie diese an einen IAM-Benutzer an. Eine praktische Anleitung zu diesem Thema finden Sie unter Praktische Anleitung: Erstellen und Anfügen Ihrer ersten vom Kunden verwalteten Richtlinie im IAM-Benutzerhandbuch.

Wenn Sie die praktischen Anleitung durchgehen, können Sie eine der in diesem Abschnitt aufgeführten Beispielrichtlinien als Ausgangsbasis verwenden und auf Ihre Bedürfnisse anpassen. Am Ende des Tutorials verfügen Sie über einen IAM-Benutzer mit einer angefügten Richtlinie, der die Aktion `neptune-db:*` verwenden kann.

> **Important**
>
> - Die Anwendung von Änderungen einer IAM-Richtlinie auf die angegebenen Neptune-Ressourcen kann bis zu 10 Minuten dauern.
> - IAM-Richtlinien, die auf einen Neptune-DB-Cluster angewendet werden, werden auf alle Instances in diesem Cluster angewendet.

Verwenden verschiedener Arten von IAM-Richtlinien für die Steuerung des Zugriffs auf Neptune

Um Zugriff auf Neptune-Verwaltungsaktionen oder auf Daten in einem Neptune-DB-Cluster zu gewähren, fügen Sie einem IAM-Benutzer oder einer IAM-Rolle Richtlinien an. Informationen zum Anfügen einer IAM-Richtlinie an einen Benutzer finden Sie unter Anfügen einer IAM-Richtlinie an einen IAM-Benutzer. Informationen zum Anfügen einer Richtlinie an eine Rolle finden Sie unter Hinzufügen und Entfernen von IAM-Richtlinien im IAM-Benutzerhandbuch.

Für den allgemeinen Zugriff auf Neptune können Sie eine der von Neptune verwalteten Richtlinien verwenden. Um den Zugriff stärker einzuschränken, können Sie mithilfe der von Neptune unterstützten administrativen Aktionen und Ressourcen eine eigene benutzerdefinierte Richtlinie erstellen.

In einer benutzerdefinierten IAM-Richtlinie können Sie zwei verschiedene Arten von Richtlinienanweisungen verwenden, die verschiedene Zugriffsmodi auf einen Neptune-DB-Cluster steuern:

- Administrative Richtlinienanweisungen – Administrative Richtlinienanweisungen ermöglichen den Zugriff auf die Neptune-Verwaltungs-APIs, die Sie zum Erstellen, Konfigurieren und Verwalten von DB-Clustern und ihrer Instances verwenden.

  Da Neptune Funktionalität mit Amazon RDS teilt, verwenden administrative Aktionen, Ressourcen und Bedingungsschlüssel in Neptune-Richtlinien standardmäßig das Präfix `rds:`.

- Data-access Richtlinienerklärungen — Data-access Richtlinienerklärungen verwenden Datenzugriffsaktionen, Ressourcen und Bedingungsschlüssel, um den Zugriff auf die Daten zu steuern, die ein DB-Cluster enthält.

  Neptune-Datenzugriffsaktionen, -Ressourcen und -Bedingungsschlüssel verwenden das Präfix `neptune-db:`.

Verwenden von IAM-Bedingungskontextschlüsseln in Amazon Neptune

Sie können Bedingungen in einer IAM-Richtlinienanweisung angeben, die den Zugriff auf Neptune steuert. Die Richtlinienanweisung ist nur wirksam, wenn die Bedingungen erfüllt werden.

Vielleicht möchten Sie, dass eine Richtlinienanweisung erst nach einem bestimmten Datum wirksam wird oder dass der Zugriff nur zulässig ist, wenn ein bestimmter Wert in der Anforderung enthalten ist.

Zum Ausdruck von Bedingungen verwenden Sie im `Condition`-Element einer Richtlinienanweisung vordefinierte Bedingungsschlüssel zusammen mit IAM-Bedingungs-Richtlinienoperatoren wie „gleich“ oder „kleiner als“.

Wenn Sie mehrere `Condition`-Elemente in einer Anweisung oder mehrere Schlüssel in einem einzelnen `Condition`-Element angeben, wertet AWS diese mittels einer logischen `AND`-Operation aus. Wenn Sie mehrere Werte für einen einzelnen Bedingungsschlüssel angeben, AWS wertet die Bedingung mithilfe einer logischen `OR` Operation aus. Alle Bedingungen müssen erfüllt sein, bevor die Berechtigungen für die Anweisung erteilt werden.

Sie können auch Platzhaltervariablen verwenden, wenn Sie Bedingungen angeben. Beispielsweise können Sie einem IAM-Benutzer die Berechtigung für den Zugriff auf eine Ressource nur dann gewähren, wenn sie mit dessen IAM-Benutzernamen gekennzeichnet ist. Weitere Informationen finden Sie unter IAM-Richtlinienelemente: Variablen und Tags im IAM-Benutzerhandbuch.

Der Datentyp eines Bedingungsschlüssels legt fest, welche Bedingungsoperatoren Sie für den Vergleich der Werte in der Anforderung mit den Werten in der Richtlinienanweisung verwenden

können. Wenn Sie einen Bedingungsoperator verwenden, der nicht mit diesem Datentyp kompatibel ist, wird niemals eine Übereinstimmung ermittelt und die Richtlinienanweisung wird nie angewendet.

Neptune unterstützt andere Sätze von Bedingungsschlüsseln für administrative Richtlinienanweisungen als für Datenzugriffs-Richtlinienanweisungen:

- Bedingungsschlüssel für administrative Richtlinienanweisungen
- Bedingungsschlüssel für Datenzugriff-Richtlinienanweisungen

Unterstützung für IAM-Richtlinien- und Zugriffssteuerungs-Features in Amazon Neptune

Die folgende Tabelle zeigt die IAM-Features, die Neptune für administrative Richtlinienanweisungen und Datenzugriff-Richtlinienanweisungen unterstützt:

IAM-Features, die Sie mit Neptune verwenden können

IAM-Feature	Administrativ	Data-access
Identity-based Richtlinien	Ja	Ja
Resource-based Richtlinien	Nein	Nein
Richtlinienaktionen	Ja	Ja
Richtlinienressourcen	Ja	Ja
Globale Bedingungsschlüssel	Ja	(eine Teilmenge)
Tag-based Zustandsschlüssel	Ja	Nein
Zugriffssteuerungslisten (ACLs)	Nein	Nein
Service-Kontrollrichtlinien (SCPs)	Ja	Ja
Serviceverknüpfte Rollen	Ja	Nein

IAM-Richtlinien – Einschränkungen

Die Anwendung von Änderungen einer IAM-Richtlinie auf die angegebenen Neptune-Ressourcen kann bis zu 10 Minuten dauern.

IAM-Richtlinien, die auf einen Neptune-DB-Cluster angewendet werden, werden auf alle Instances in diesem Cluster angewendet.

Neptune unterstützt derzeit keine kontenübergreifende Zugriffskontrolle auf Datenebene. Cross-account Die Zugriffskontrolle wird nur beim Massenladen und bei Verwendung von Rollenverkettung unterstützt. Weitere Informationen finden Sie im Tutorial zum Massenladen.

Verwenden AWS verwaltete Richtlinien für den Zugriff auf Amazon Neptune Neptune-Datenbanken

AWS adressiert viele gängige Anwendungsfälle durch die Bereitstellung eigenständiger IAM-Richtlinien, die von erstellt und verwaltet werden. AWS Die verwalteten Richtlinien erteilen die erforderlichen Berechtigungen für viele häufige Anwendungsfälle, sodass Sie nicht mühsam ermitteln müssen, welche Berechtigungen erforderlich sind. Weitere Informationen finden Sie unter AWS - verwaltete Richtlinien im IAM-Benutzerhandbuch.

Die folgenden AWS verwalteten Richtlinien, die Sie Benutzern in Ihrem Konto zuordnen können, gelten für die Verwendung von Amazon Neptune Neptune-Verwaltungs-APIs:

- NeptuneReadOnlyAccess— Gewährt schreibgeschützte Verwaltungsaktionen (wie `rds:Describe*` und`rds:ListTagsForResource`) und schreibgeschützte Datenzugriffsaktionen (`neptune-db:Read*`, `neptune-db:Get*`, und`neptune-db:List*`) für alle Neptun-Ressourcen. Verwenden Sie diese Richtlinie für Benutzer, die Clusterkonfigurationen anzeigen und Daten abfragen müssen, ohne Änderungen vorzunehmen.
- NeptuneFullAccess— Erteilt alle administrativen Aktionen (`rds:*`auf Neptun-Ressourcen) und alle Datenzugriffsaktionen (). `neptune-db:*` Diese Richtlinie eignet sich für Administratoren, die Neptune-Cluster über das AWS CLI oder SDK verwalten, aber keinen Zugriff benötigen AWS-Managementkonsole .
- NeptuneConsoleFullAccess— Erteilt alle administrativen Aktionen für Neptune-Ressourcen sowie zusätzliche Berechtigungen für Amazon EC2 (VPC), IAM und Neptune Analytics, die für Workflows benötigt werden. AWS-Managementkonsole Diese Richtlinie beinhaltet keine Datenzugriffsaktionen (). `neptune-db:*` Verwenden Sie diese Richtlinie für Benutzer, die Neptune über die AWS Management Console verwalten.

- NeptuneGraphReadOnlyAccess— Diese Richtlinie gilt für Neptune Analytics. Einzelheiten finden Sie NeptuneGraphReadOnlyAccessin Neptune Analytics.
- AWSServiceRoleForNeptuneGraphPolicy— Diese Richtlinie gilt für Neptune Analytics. Einzelheiten finden Sie AWSServiceRoleForNeptuneGraphPolicyin Neptune Analytics.

Neptune-IAM-Rollen und -Richtlinien gewähren einen bestimmten Zugriff auf Amazon–RDS-Ressourcen, da Neptune für bestimmte Verwaltungs-Features operative Technologien mit Amazon RDS teilt. Dazu gehören administrative API-Berechtigungen, weswegen administrative Neptune-Aktionen das Präfix `rds:` haben.

Benutzerdefinierte Richtlinien erstellen

Wenn die AWS verwalteten Richtlinien für Ihren Anwendungsfall zu umfassend sind, können Sie benutzerdefinierte IAM-Richtlinien erstellen, die nur die spezifischen Berechtigungen gewähren, die Sie benötigen. Neptune unterstützt zwei Kategorien von benutzerdefinierten Richtlinien:

- Verwaltungsrichtlinien — Steuern Sie den Zugriff auf Neptune-Verwaltungsvorgänge wie das Erstellen, Ändern und Löschen von Clustern und Instanzen. Diese Aktionen verwenden das `rds:` Präfix. Beispiele finden Sie unter the section called “Beispiele für Verwaltungsrichtlinien”.
- Data-access Richtlinien — Steuern Sie den Zugriff auf die Daten in Ihrer Neptune-Graphdatenbank, einschließlich Lese-, Schreib- und Löschoperationen. Diese Aktionen verwenden das `neptune-db:` Präfix. Beispiele finden Sie unter the section called “Data-access Beispiele für Richtlinien”.

Durch die Kombination von Richtlinien für Verwaltung und Datenzugriff können Sie detaillierte Berechtigungen gewähren, die auf jeden Benutzer oder jede Rolle in Ihrer Organisation zugeschnitten sind.

Validieren von IAM-Richtlinien

Wenn Sie benutzerdefinierte IAM-Richtlinien erstellen oder bearbeiten, empfehlen wir Ihnen, diese zu überprüfen, bevor Sie sie auf Benutzer, Gruppen oder Rollen anwenden.

IAM Access Analyzer-Richtlinienvalidierung — IAM Access Analyzer bietet Richtlinienprüfungen, mit denen Ihre IAM-Richtlinien anhand der Grammatik und der Best Practices der IAM-Richtlinien überprüft werden. AWS Es identifiziert Fehler, Sicherheitswarnungen und Vorschläge, um Ihnen bei der Erstellung von Richtlinien zu helfen, die funktionsfähig sind und den bewährten Sicherheitsmethoden entsprechen. Weitere Informationen finden Sie unter Überprüfen von Richtlinien mit IAM Access Analyzer im IAM-Benutzerhandbuch.

IAM Policy Simulator — Mit dem IAM Policy Simulator können Sie die Auswirkungen von IAM-Richtlinien testen, bevor Sie sie in die Produktion übernehmen. Sie können API-Aufrufe an AWS Dienste simulieren, um zu überprüfen, ob Ihre Richtlinien den erwarteten Zugriff gewähren oder verweigern. Weitere Informationen finden Sie unter Testen von IAM-Richtlinien mit dem IAM-Richtliniensimulator im IAM-Benutzerhandbuch.

Updates zu Neptune AWS Verwaltete Richtlinien

Die folgende Tabelle listet die Aktualisierungen der von Neptune verwalteten Richtlinien ab dem Zeitpunkt auf, an dem Neptune mit der Verfolgung dieser Änderungen begonnen hat:

Richtlinie	Description	Date
AWS verwaltete Richtlinien für Amazon Neptune — Aktualisierung vorhandener Richtlinien	Die `NeptuneReadOnlyAccess` und die `NeptuneFullAccess` verwalteten Richtlinien enthalten jetzt `Sid` (Kontoausweis-ID) als Kennung in der Richtlinienerklärung.	2024-01-22
NeptuneGraphReadOnlyAccess(veröffentlicht)	Diese Version bietet schreibgeschützten Zugriff auf Neptune-Analytics-Graphen und -Ressourcen.	29.11.2023-
AWSServiceRoleForNeptuneGraphPolicy(veröffentlicht)	Veröffentlicht, um Neptune Analytics-Graphen Zugriff auf die Veröffentlichung von Betriebs- und Nutzungsmetriken und Protokollen CloudWatch zu ermöglichen. Weitere Informationen finden Sie unter Using service-linked roles (SLRs) in Neptune Analytics.	29.11.2023-

Richtlinie	Description	Date
NeptuneConsoleFullAccess(Berechtigungen hinzugefügt)	Die hinzugefügten Berechtigungen bieten den gesamten benötigten Zugriff für die Interaktion mit Neptune-Analytics-Graphen.	2023- 11/29
NeptuneFullAccess(Berechtigungen hinzugefügt)	Es wurden Datenzugriffsberechtigungen und Berechtigungen für neue globale Datenbank-APIs hinzugefügt.	28.07.2022
NeptuneConsoleFullAccess(Berechtigungen hinzugefügt)	Hinzufügung von Berechtigungen für neue globale Datenbank-APIs.	2021-07-21
Beginn der Verfolgung von Änderungen durch Neptune	Neptune begann, Änderungen an seinen AWS verwalteten Richtlinien zu verfolgen.	21.07.2022

Gewährung **`NeptuneReadOnlyAccess`**an Amazon Neptune Neptune-Datenbanken mithilfe von AWS Verwaltete Richtlinie

Die unten stehende NeptuneReadOnlyAccessverwaltete Richtlinie gewährt Lesezugriff auf alle Neptune-Aktionen und -Ressourcen sowohl für administrative als auch für Datenzugriffszwecke.

> **Note**
>
> Diese Richtlinie wurde am 21.07.2022 aktualisiert und umfasst nun schreibgeschützte Datenzugriffsberechtigungen und administrative Berechtigungen sowie Berechtigungen für globale Datenbankaktionen.

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": [
                "*"
            ]
        }
```

```
    ]
}
```

Gewährung **NeptuneFullAccess**an Amazon Neptune Neptune-Datenbanken mithilfe von AWS Verwaltete Richtlinie

Die unten stehende NeptuneFullAccessverwaltete Richtlinie gewährt vollen Zugriff auf alle Neptune-Aktionen und -Ressourcen, sowohl für administrative als auch für Datenzugriffszwecke. Sie wird empfohlen, wenn Sie vollen Zugriff über das AWS CLI oder über ein SDK benötigen, aber nicht über das. AWS-Managementkonsole

> **Note**
>
> Diese Richtlinie wurde am 21.07.2022 aktualisiert und umfasst vollständige Datenzugriffs- und administrative Berechtigungen sowie Berechtigungen für globale Datenbankaktionen.

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": [
                "*"
            ]
        }
    ]
}
```

Gewährung **NeptuneConsoleFullAccess**mit AWS Verwaltete Richtlinie

Die unten stehende NeptuneConsoleFullAccessverwaltete Richtlinie gewährt vollen Zugriff auf alle Neptun-Aktionen und -Ressourcen für administrative Zwecke, jedoch nicht für Datenzugriffszwecke. Dies schließt zusätzliche Berechtigungen ein, um den Zugriff auf Neptune über die Konsole zu vereinfachen, einschließlich eingeschränkter IAM- und Amazon-EC2 (VPC)-Berechtigungen.

> **Note**
>
> Diese Richtlinie wurde am 29.11.2023 aktualisiert und umfasst nun auch die Berechtigungen, die für die Interaktion mit Neptune-Analytics-Graphen erforderlich sind.
> Sie wurde am 21.07.2022 aktualisiert und umfasst nun auch Berechtigungen für globale Datenbankaktionen.

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"
        }
      }
    }
  ]
}
```

Anpassen des Zugriffs auf Amazon Neptune Neptune-Ressourcen mithilfe von IAM-Bedingungskontextschlüsseln

Sie können in den IAM-Richtlinien, die den Zugriff auf Neptune-Verwaltungsaktionen und -Ressourcen steuern, Bedingungen angeben. Die Richtlinienanweisung ist nur wirksam, wenn die Bedingungen erfüllt werden.

Vielleicht möchten Sie, dass eine Richtlinienanweisung erst nach einem bestimmten Datum wirksam wird oder dass der Zugriff nur zulässig ist, wenn ein bestimmter Wert in der API-Anforderung enthalten ist.

Zum Ausdruck von Bedingungen verwenden Sie im `Condition`-Element einer Richtlinienanweisung vordefinierte Bedingungsschlüssel zusammen mit IAM-Bedingungs-Richtlinienoperatoren wie „gleich“ oder „kleiner als“.

Wenn Sie mehrere `Condition`-Elemente in einer Anweisung oder mehrere Schlüssel in einem einzelnen `Condition`-Element angeben, wertet AWS diese mittels einer logischen `AND`-Operation aus. Wenn Sie mehrere Werte für einen einzelnen Bedingungsschlüssel angeben, AWS wertet die Bedingung mithilfe einer logischen Operation aus. `OR` Alle Bedingungen müssen erfüllt sein, bevor die Berechtigungen für die Anweisung erteilt werden.

Sie können auch Platzhaltervariablen verwenden, wenn Sie Bedingungen angeben. Beispielsweise können Sie einem IAM-Benutzer die Berechtigung für den Zugriff auf eine Ressource nur dann gewähren, wenn sie mit dessen IAM-Benutzernamen gekennzeichnet ist. Weitere Informationen finden Sie unter IAM-Richtlinienelemente: Variablen und Tags im IAM-Benutzerhandbuch.

Der Datentyp eines Bedingungsschlüssels legt fest, welche Bedingungsoperatoren Sie für den Vergleich der Werte in der Anforderung mit den Werten in der Richtlinienanweisung verwenden können. Wenn Sie einen Bedingungsoperator verwenden, der nicht mit diesem Datentyp kompatibel ist, wird niemals eine Übereinstimmung ermittelt und die Richtlinienanweisung wird nie angewendet.

IAM-Bedingungsschlüssel für administrative Neptune-Richtlinienanweisungen

- Globale Bedingungsschlüssel — Sie können die meisten AWS globalen Bedingungsschlüssel in den administrativen Richtlinienerklärungen von Neptune verwenden.
- Service-specific Bedingungsschlüssel — Dies sind Schlüssel, die für bestimmte AWS Dienste definiert sind. Die Bedingungsschlüssel, die Neptune für administrative Richtlinienanweisungen unterstützt, werden in IAM-Bedingungsschlüssel für die Verwaltung von Amazon Neptune aufgelistet.

IAM-Bedingungsschlüssel für Neptune-Datenzugriff-Richtlinienanweisungen

- Globale Bedingungsschlüssel – Die Teilmenge dieser Schlüssel, die Neptune in Datenzugriff-Richtlinienanweisungen unterstützt, werden in AWS Globale Bedingungskontextschlüssel, die von Neptune in Datenzugriffspolitikanweisungen unterstützt werden aufgelistet.
- Service-specific Bedingungsschlüssel, die Neptune für Datenzugriffspolitikanweisungen definiert, sind unter aufgeführt. Bedingungsschlüssel

Erstellen von benutzerdefinierten IAM-Richtlinienerklärungen zur Verwaltung von Amazon Neptune

Mit administrativen Richtlinienanweisungen können Sie steuern, welche Aktionen ein IAM-Benutzer ausführen kann, um eine Neptune-Datenbank zu verwalten.

Eine administrative Neptune-Richtlinienanweisung gewährt Zugriff auf eine oder mehrere administrative Aktionen und administrative Ressourcen, die von Neptune unterstützt werden. Sie können Bedingungsschlüssel auch verwenden, um die administrativen Berechtigungen spezifischer zu gestalten.

> **Note**
>
> Da Neptune Funktionalität mit Amazon RDS teilt, verwenden administrative Aktionen, Ressourcen und servicespezifische Bedingungsschlüssel in administrativen Richtlinienanweisungen standardmäßig das Präfix `rds:`.

Themen

- IAM-Aktionen zur Verwaltung von Amazon Neptune

- IAM-Ressourcentypen für die Verwaltung von Amazon Neptune
- IAM-Bedingungsschlüssel für die Verwaltung von Amazon Neptune
- Erstellung von IAM-Verwaltungsrichtlinienerklärungen für Amazon Neptune

IAM-Aktionen zur Verwaltung von Amazon Neptune

Sie können die unten aufgelisteten administrativen Aktionen im `Action`-Element einer IAM-Richtlinienanweisung verwenden, um den Zugriff auf die Neptune-Verwaltungs-APIs zu kontrollieren. Wenn Sie eine Aktion in einer Richtlinie verwenden, erlauben oder verweigern Sie in der Regel den Zugriff auf die API-Operation oder den CLI-Befehl mit demselben Namen. Dabei kann es mitunter vorkommen, dass eine einzige Aktion den Zugriff auf mehr als eine Operation steuert. Alternativ erfordern einige Vorgänge mehrere verschiedene Aktionen.

Das Feld `Resource type` in der folgenden Liste gibt an, ob die einzelnen Aktionen jeweils Berechtigungen auf Ressourcenebene unterstützen. Wenn es keinen Wert in diesem Feld gibt, müssen Sie alle Ressourcen ("*") im `Resource`-Element Ihrer Richtlinienanweisung angeben. Wenn die Spalte einen Ressourcentyp enthält, können Sie einen Ressourcen-ARN dieses Typs in einer Anweisung mit dieser Aktion angeben. Die administrativen Neptune-Ressourcentypen sind auf dieser Seite aufgelistet.

Erforderliche Ressourcen werden in der folgenden Liste mit einem Sternchen (*) gekennzeichnet. Wenn Sie einen Berechtigungs-ARN auf Ressourcenebene in einer Anweisung mit dieser Aktion angeben, muss er diesen Typ haben. Einige Aktionen unterstützen mehrere Ressourcentypen. Wenn ein Ressourcentyp optional ist (d. h. nicht mit einem Sternchen gekennzeichnet ist), müssen Sie ihn nicht angeben.

Weitere Informationen zu den hier aufgelisteten Feldern finden Sie unter Aktionstabelle im IAM-Benutzerhandbuch.

rds: AddRoleToDBCluster

`AddRoleToDBCluster` verknüpft eine IAM-Rolle mit einem Neptune-DB-Cluster.

Zugriffsebene: `Write`.

Abhängige Aktionen: `iam:PassRole`.

Ressourcentyp: cluster (erforderlich).

rds: AddSourceIdentifierToSubscription

`AddSourceIdentifierToSubscription` fügt eine Quell-ID zu einem vorhandenen Neptune-Ereignisbenachrichtigungs-Abonnement hinzu.

Zugriffsebene: `Write`.

Ressourcentyp: es (erforderlich).

rds: AddTagsToResource

`AddTagsToResource` verknüpft eine IAM-Rolle mit einem Neptune-DB-Cluster.

Zugriffsebene: `Write`.

Ressourcentypen:

- db
- es
- pg
- cluster-snapshot
- subgrp

Bedingungsschlüssel:

- als:RequestTag/*tag-key*
- war: TagKeys

rds: ApplyPendingMaintenanceAction

`ApplyPendingMaintenanceAction` wendet eine ausstehende Wartungsaktion auf eine Ressource an.

Zugriffsebene: `Write`.

Ressourcentyp: db (erforderlich).

rds: CopyDBClusterParameterGroup

`Kopieren DBCluster ParameterGroup` kopiert die angegebene DB-Cluster-Parametergruppe.

Zugriffsebene: `Write`.

Ressourcentyp: cluster-pg (erforderlich).

rds: CopyDBClusterSnapshot

`DBClusterSnapshot kopieren` kopiert einen Snapshot eines DB-Clusters.

Zugriffsebene: `Write`.

Ressourcentyp: cluster-snapshot (erforderlich).

rds: CopyDBParameterGroup

`DBParameterGruppe kopieren` kopiert die angegebene DB-Parametergruppe.

Zugriffsebene: `Write`.

Ressourcentyp: pg (erforderlich).

rds:CreateDBCluster

`Erstellen DBCluster` erstellt einen neuen Neptune-DB-Cluster.

Zugriffsebene: `Tagging`.

Abhängige Aktionen: `iam:PassRole`.

Ressourcentypen:

- cluster (erforderlich).
- cluster-pg (erforderlich).
- subgrp (erforderlich).

Bedingungsschlüssel:

- als:RequestTag/*tag-key*
- war: TagKeys
- neptun-Rds_ DatabaseEngine

rds: CreateDBClusterParameterGroup

`Erstellen DBCluster ParameterGroup` erstellt eine neue DB-Cluster-Parametergruppe.

Zugriffsebene: `Tagging`.

Ressourcentyp: cluster-pg (erforderlich).

Bedingungsschlüssel:

- als:RequestTag/*tag-key*
- war: TagKeys

rds: CreateDBClusterSnapshot

`DBClusterSnapshot erstellen` erstellt einen Snapshot eines DB-Clusters.

Zugriffsebene: `Tagging`.

Ressourcentypen:

- cluster (erforderlich).
- cluster-snapshot (erforderlich).

Bedingungsschlüssel:

- als:RequestTag/*tag-key*
- war: TagKeys

rds:CreateDBInstance

`Erstellen DBInstance` erstellt eine neue DB-Instance.

Zugriffsebene: `Tagging`.

Abhängige Aktionen: `iam:PassRole`.

Ressourcentypen:

- db (erforderlich).
- pg (erforderlich).

- subgrp (erforderlich).

Bedingungsschlüssel:

- aws:RequestTag/*tag-key*
- war: TagKeys

rds: CreateDBParameterGroup

`DBParameterGruppe erstellen` erstellt eine neue DB-Parametergruppe.

Zugriffsebene: `Tagging`.

Ressourcentyp: pg (erforderlich).

Bedingungsschlüssel:

- als:RequestTag/*tag-key*
- war: TagKeys

rds: CreateDBSubnetGroup

`DBSubnetGruppe erstellen` erstellt eine neue DB-Subnetzgruppe.

Zugriffsebene: `Tagging`.

Ressourcentyp: subgrp (erforderlich).

Bedingungsschlüssel:

- als:RequestTag/*tag-key*
- war: TagKeys

rds: CreateEventSubscription

`CreateEventSubscription` erstellt ein Neptune-Ereignisbenachrichtigungs-Abonnement.

Zugriffsebene: `Tagging`.

Ressourcentyp: es (erforderlich).

Bedingungsschlüssel:

- als:RequestTag/*tag-key*
- war: TagKeys

rds:DeleteDBCluster

`Löschen DBCluster` löscht einen vorhandenen Neptune-DB-Cluster.

Zugriffsebene: `Write`.

Ressourcentypen:

- cluster (erforderlich).
- cluster-snapshot (erforderlich).

rds: DeleteDBClusterParameterGroup

`Löschen DBCluster ParameterGroup` löscht eine angegebene DB-Cluster-Parametergruppe.

Zugriffsebene: `Write`.

Ressourcentyp: cluster-pg (erforderlich).

rds: DeleteDBClusterSnapshot

`DBClusterSchnappschuss löschen` löscht einen DB-Cluster-Snapshot.

Zugriffsebene: `Write`.

Ressourcentyp: cluster-snapshot (erforderlich).

rds:DeleteDBInstance

`Löschen DBInstance` löscht eine angegebene DB-Instance.

Zugriffsebene: `Write`.

Ressourcentyp: db (erforderlich).

rds: DeleteDBParameterGroup

`DBParameterGruppe löschen`löscht eine angegebene DBParameterGroup.

Zugriffsebene: `Write`.

Ressourcentyp: pg (erforderlich).

rds: DeleteDBSubnetGroup

`DBSubnetGruppe löschen` löscht eine DB-Subnetzgruppe.

Zugriffsebene: `Write`.

Ressourcentyp: subgrp (erforderlich).

rds: DeleteEventSubscription

`DeleteEventSubscription` löscht ein Ereignisbenachrichtigungs-Abonnement.

Zugriffsebene: `Write`.

Ressourcentyp: es (erforderlich).

rds: DescribeDBClusterParameterGroups

`Beschreiben DBCluster ParameterGroups`gibt eine Liste von DBClusterParameterGroup Beschreibungen zurück.

Zugriffsebene: `List`.

Ressourcentyp: cluster-pg (erforderlich).

rds: DescribeDBClusterParameters

`Beschreiben Sie die DBCluster Parameter` gibt die detaillierte Parameterliste für eine bestimmte DB-Cluster-Parametergruppe zurück.

Zugriffsebene: `List`.

Ressourcentyp: cluster-pg (erforderlich).

rds: DescribeDBClusterSnapshotAttributes

`Beschreiben DBCluster SnapshotAttributes` gibt eine Liste der Namen und Werte von DB-Cluster-Snapshot-Attributen für einen manuellen DB-Cluster-Snapshot zurück.

Zugriffsebene: `List`.

Ressourcentyp: cluster-snapshot (erforderlich).

rds: DescribeDBClusterSnapshots

`Beschreiben Sie Schnappschüsse DBCluster` gibt Informationen zu DB-Cluster-Snapshots zurück.

Zugriffsebene: `Read`.

rds:DescribeDBClusters

`Beschreiben DBClusters` gibt Informationen zu einem bereitgestellten Neptune-DB-Cluster zurück.

Zugriffsebene: `List`.

Ressourcentyp: cluster (erforderlich).

rds: DescribeDBEngineVersions

`DBEngineVersionen beschreiben` gibt eine Liste der verfügbaren DB-Engines zurück.

Zugriffsebene: `List`.

Ressourcentyp: pg (erforderlich).

rds:DescribeDBInstances

`Beschreiben DBInstances` gibt Informationen zu DB-Instances zurück.

Zugriffsebene: `List`.

Ressourcentyp: es (erforderlich).

rds: DescribeDBParameterGroups

`Beschreiben Sie DBParameter Gruppen`gibt eine Liste von DBParameterGroup Beschreibungen zurück.

Zugriffsebene: `List`.

Ressourcentyp: pg (erforderlich).

rds:DescribeDBParameters

`Beschreiben DBParameters` gibt die detaillierte Parameterliste für eine bestimmte DB-Parametergruppe zurück.

Zugriffsebene: `List`.

Ressourcentyp: pg (erforderlich).

rds: DescribeDBSubnetGroups

`Beschreiben Sie DBSubnet Gruppen`gibt eine Liste von DBSubnetGroup Beschreibungen zurück.

Zugriffsebene: `List`.

Ressourcentyp: subgrp (erforderlich).

rds: DescribeEventCategories

`DescribeEventCategories` gibt eine Liste der Kategorien für alle Ereignisquelltypen oder (wenn angegeben) für einen angegebenen Quelltyp zurück.

Zugriffsebene: `List`.

rds: DescribeEventSubscriptions

`DescribeEventSubscriptions` listet alle Abonnementbeschreibungen für ein Kundenkonto auf.

Zugriffsebene: `List`.

Ressourcentyp: es (erforderlich).

rds: DescribeEvents

`DescribeEvents` gibt Ereignisse für DB-Instances, DB-Sicherheitsgruppen und DB-Parametergruppen in den vergangenen 14 Tagen zurück.

Zugriffsebene: `List`.

Ressourcentyp: es (erforderlich).

rds: DescribeOrderableDBInstanceOptions

`DescribeOrderableDBInstanceOptionen` gibt eine Liste der bestellbaren DB-Instance-Optionen für die angegebene Engine zurück.

Zugriffsebene: `List`.

rds: DescribePendingMaintenanceActions

`DescribePendingMaintenanceActions` gibt eine Liste von Ressourcen (z. B. DB-Instances) zurück, für die mindestens eine Wartungsaktion aussteht.

Zugriffsebene: `List`.

Ressourcentyp: db (erforderlich).

rds: DescribeValidDBInstanceModifications

`DescribeValidDBInstanceÄnderungen` listet verfügbare Änderungen auf, die Sie für Ihre DB-Instance ausführen können

Zugriffsebene: `List`.

Ressourcentyp: db (erforderlich).

rds:FailoverDBCluster

`Failover DBCluster` erzwingt einen Failover für einen DB-Cluster.

Zugriffsebene: `Write`.

Ressourcentyp: cluster (erforderlich).

rds: ListTagsForResource

`ListTagsForResource` listet alle Tags für eine Neptune-Ressource auf.

Zugriffsebene: `Read`.

Ressourcentypen:

- cluster-snapshot
- db
- es

- pg
- subgrp

rds:ModifyDBCluster

`Modifizieren DBCluster`

Ändert eine Einstellung für einen Neptune-DB-Cluster.

Zugriffsebene: `Write`.

Abhängige Aktionen: `iam:PassRole`.

Ressourcentypen:

- cluster (erforderlich).
- cluster-pg (erforderlich).

rds: ModifyDBClusterParameterGroup

`Modifizieren DBCluster ParameterGroup` ändert die Parameter einer DB-Cluster-Parametergruppe.

Zugriffsebene: `Write`.

Ressourcentyp: cluster-pg (erforderlich).

rds: ModifyDBClusterSnapshotAttribute

`Modifizieren DBCluster SnapshotAttribute` fügt ein Attribut und Werte zu einem manuellen DB-Cluster-Snapshot hinzu oder entfernt ein Attribut und Werte aus diesem.

Zugriffsebene: `Write`.

Ressourcentyp: cluster-snapshot (erforderlich).

rds:ModifyDBInstance

`Modifizieren DBInstance` ändert Einstellungen für eine DB-Instance.

Zugriffsebene: `Write`.

Abhängige Aktionen: `iam:PassRole`.

Ressourcentypen:

- db (erforderlich).
- pg (erforderlich).

rds: ModifyDBParameterGroup

`DBParameterGruppe ändern` ändert die Parameter einer DB-Parametergruppe.

Zugriffsebene: `Write`.

Ressourcentyp: pg (erforderlich).

rds: ModifyDBSubnetGroup

`DBSubnetGruppe ändern` ändert eine vorhandene DB-Subnetzgruppe.

Zugriffsebene: `Write`.

Ressourcentyp: subgrp (erforderlich).

rds: ModifyEventSubscription

`ModifyEventSubscription` ändert ein vorhandenes Neptune-Ereignisbenachrichtigungs-Abonnement.

Zugriffsebene: `Write`.

Ressourcentyp: es (erforderlich).

rds:RebootDBInstance

`Starten Sie neu DBInstance` startet den Datenbank-Engine-Service für die Instance neu.

Zugriffsebene: `Write`.

Ressourcentyp: db (erforderlich).

rds: RemoveRoleFromDBCluster

`RemoveRoleFromDBCluster`trennt eine AWS Identity and Access Management (IAM) -Rolle von einem Amazon Neptune Neptune-DB-Cluster.

Zugriffsebene: `Write`.

Abhängige Aktionen: `iam:PassRole`.

Ressourcentyp: cluster (erforderlich).

rds: RemoveSourceIdentifierFromSubscription

`RemoveSourceIdentifierFromSubscription` entfernt eine Quell-ID aus einem vorhandenen Neptune-Ereignisbenachrichtigungs-Abonnement.

Zugriffsebene: `Write`.

Ressourcentyp: es (erforderlich).

rds: RemoveTagsFromResource

`RemoveTagsFromResource` entfernt Metadaten-Tags aus einer Neptune-Ressource.

Zugriffsebene: `Tagging`.

Ressourcentypen:

- cluster-snapshot
- db
- es
- pg
- subgrp

Bedingungsschlüssel:

- als:RequestTag/*tag-key*
- war: TagKeys

rds: ResetDBClusterParameterGroup

`Zurücksetzen DBCluster ParameterGroup` ändert die Parameter einer DB-Cluster-Parametergruppe auf den Standardwert zurück.

Zugriffsebene: `Write`.

Ressourcentyp: cluster-pg (erforderlich).

rds: ResetDBParameterGroup

DBParameterGruppe zurücksetzenändert die Parameter einer DB-Parametergruppe auf den engine/system Standardwert.

Zugriffsebene: `Write`.

Ressourcentyp: pg (erforderlich).

rds: RestoreDBClusterFromSnapshot

Wiederherstellen DBCluster FromSnapshot erstellt einen neuen DB-Cluster aus einem DB-Cluster-Snapshot

Zugriffsebene: `Write`.

Abhängige Aktionen: `iam:PassRole`.

Ressourcentypen:

- cluster (erforderlich).
- cluster-snapshot (erforderlich).

Bedingungsschlüssel:

- als:RequestTag/*tag-key*
- war: TagKeys

rds: RestoreDBClusterToPointInTime

Wiederherstellen DBCluster ToPointInTime stellt einen DB-Cluster zu einem beliebigen Zeitpunkt her.

Zugriffsebene: `Write`.

Abhängige Aktionen: `iam:PassRole`.

Ressourcentypen:

- cluster (erforderlich).
- subgrp (erforderlich).

Bedingungsschlüssel:

- als:RequestTag/*tag-key*
- war: TagKeys

rds:StartDBCluster

`Starten DBCluster` startet den angegebenen DB-Cluster.

Zugriffsebene: `Write`.

Ressourcentyp: cluster (erforderlich).

rds:StopDBCluster

`Stopp DBCluster` stoppt den angegebenen DB-Cluster.

Zugriffsebene: `Write`.

Ressourcentyp: cluster (erforderlich).

IAM-Ressourcentypen für die Verwaltung von Amazon Neptune

Neptune unterstützt die Ressourcentypen in der folgenden Tabelle für die Verwendung im `Resource`-Element der administrativen IAM-Richtlinienanweisungen. Weitere Informationen zum `Resource`-Element finden Sie unter IAM-JSON-Richtlinienelemente: Ressource.

Die Liste der Neptune-Verwaltungsaktionen identifiziert die Ressourcentypen, die für die einzelnen Aktionen angegeben werden können. Ein Ressourcentyp bestimmt auch, welche Bedingungsschlüssel Sie in einer Richtlinie einschließen können, wie in der letzten Spalte der Tabelle angegeben.

Die Spalte `ARN` in der folgenden Tabelle gibt das Format für den Amazon-Ressourcennamen (ARN) an, das Sie verwenden müssen, um Ressourcen dieses Typs zu referenzieren. Die Bestandteile mit vorangestelltem `$` müssen durch die tatsächlichen Werte für das jeweilige Szenario ersetzt werden. Beispiel: Wenn Sie `$user-name` in einem ARN sehen, müssen Sie diese Zeichenfolge durch den tatsächlichen Namen eines IAM-Benutzers oder eine Richtlinienvariable ersetzen, die den

Namen eines IAM-Benutzers enthält. Weitere Informationen zu ARNs finden Sie unter IAM-ARNs und Arbeiten mit administrativen ARNs in Amazon Neptune.

Die Spalte `Condition Keys` Bedingungsschlüssel gibt Bedingungskontextschlüssel an, die Sie nur dann in eine IAM-Richtlinienanweisung einfügen können, wenn sowohl diese Ressource als auch eine kompatible unterstützende Aktion in der Anweisung enthalten sind.

Ressourcentypen	ARN	Bedingungsschlüssel
`cluster` (ein DB-Cluster)	`arn:`*`partition`*` :rds:`*`region`*`:`*`account-id`*` :cluster: `*`instance-name`*	aws:/ResourceTag*tag-key* rds: Cluster-Tag/ *tag-key*
`cluster-pg` (eine DB-Cluster-Parametergruppe)	`arn:`*`partition`*` :rds:`*`region`*`:`*`account-id`*` :cluster-pg: `*`neptune-DBClusterParameterGroupName`*	war:/ResourceTag*tag-key*
`cluster-snapshot` (ein DB-Cluster-Snapshot)	`arn:`*`partition`*` :rds:`*`region`*`:`*`account-id`*` :cluster-snapshot: `*`neptune-DBClusterSnapshotName`*	aws:ResourceTag/*tag-key* rds: cluster-snapshot-tag/ *tag-key*
db (eine DB-Instance)	`arn:`*`partition`*` :rds:`*`region`*`:`*`account-id`*` :db:`*`neptune-DbInstanceName`*	war:/ResourceTag*tag-key* rds: DatabaseClass rds: DatabaseEngine rds: db-tag/ *tag-key*

Ressourcentypen	ARN	Bedingungsschlüssel
es (ein Ereignisabonnement)	arn:*partition* :rds:*region*:*account-id* :es:*neptune-CustSubscriptionId*	war:/ResourceTag*tag-key* rds:es-tag/ *tag-key*
pg (eine DB-Parametergruppe)	arn:*partition* :rds:*region*:*account-id* :pg:*neptune-ParameterGroupName*	war:/ResourceTag*tag-key* rds: pg-tag/ *tag-key*
subgrp (eine DB-Subnetzgruppe)	arn:*partition* :rds:*region*:*account-id* :subgrp:*neptune-DBSubnetGroupName* }	war:/ResourceTag*tag-key* rds: subgrp-tag/ *tag-key*

IAM-Bedingungsschlüssel für die Verwaltung von Amazon Neptune

Mithilfe von Bedingungsschlüsseln können Sie Bedingungen in einer IAM-Richtlinienanweisung angeben, damit die Anweisung nur dann wirksam wird, wenn die Bedingungen erfüllt werden. Die Bedingungsschlüssel, die Sie in administrativen Richtlinienanweisungen in Neptune verwenden können, gehören den folgenden Kategorien an:

- Globale Bedingungsschlüssel — Diese sind für die allgemeine Verwendung mit Diensten definiert. AWS AWS Die meisten können in den administrativen Richtlinienanweisungen in Neptune verwendet werden.
- Administrative Ressourceneigenschaft-Bedingungsschlüssel – Diese Schlüssel (wie unten aufgelistet) basieren auf Eigenschaften von Verwaltungsressourcen.
- Tag-based Schlüssel für Zugriffsbedingungen — Diese unten aufgeführten Schlüssel basieren auf AWS Tags, die mit administrativen Ressourcen verknüpft sind.

Administrative Ressourceneigenschaft-Bedingungssschlüssel in Neptune

Bedingungsschlüssel	Beschreibung	Typ
`rds:DatabaseClass`	Filtert den Zugriff nach Typ der DB-Instance-Klasse	Zeichenfolge
`rds:DatabaseEngine`	Filtert den Zugriff nach dem Datenbank-Engine. Mögliche Werte finden Sie im Engine-Parameter in der CreateDBInstance-API	Zeichenfolge
`rds:DatabaseName`	Filtert den Zugriff nach benutzerdefiniertem Name der Datenbank auf der DB-Instance	Zeichenfolge
`rds:EndpointType`	Filtert den Zugriff nach dem Typ des Endpunkts. Einer der folgenden Typen: READER, WRITER, CUSTOM.	Zeichenfolge
`rds:Vpc`	Filtert den Zugriff nach dem Wert, der angibt, ob die DB-Instance in einer Amazon Virtual Private Cloud (Amazon VPC) ausgeführt wird. Geben Sie `true` an, um anzuzeigen, dass die DB-Instance in einer Amazon-VPC ausgeführt wird.	Boolesch

Administrative Tag-basierte Bedingungsschlüssel

Amazon Neptune unterstützt die Angabe von Bedingungen in einer IAM-Richtlinie mit benutzerdefinierten Tags, um den Zugriff auf Neptune über die Management-API-Referenz zu steuern.

Wenn Sie beispielsweise Ihren DB-Instances ein Tag mit dem Namen `environment` und Werten wie `beta`, `staging` und `production` hinzufügen, können Sie anschließend eine Richtlinie erstellen, die den Zugriff auf die Instances anhand des Werts dieses Tags einschränkt.

> **Important**
>
> Wenn Sie den Zugriff auf Ihre Neptune-Ressourcen mit Tags verwalten, muss der Zugriff auf die Tags geschützt werden. Sie können den Zugriff auf Tags einschränken, indem

Sie Richtlinien für die Aktionen `AddTagsToResource` und `RemoveTagsFromResource` erstellen.
Beispielsweise könnten Sie die folgende Richtlinie verwenden, um das Hinzufügen oder Entfernen von Tags für alle Ressourcen abzulehnen. Anschließend könnten Sie Richtlinien erstellen, um bestimmten Benutzern das Hinzufügen oder Entfernen von Tags zu ermöglichen.

JSON

```
{ "Version":"2012-10-17",
  "Statement":[
    { "Sid": "DenyTagUpdates",
      "Effect": "Deny",
      "Action": [
        "rds:AddTagsToResource",
        "rds:RemoveTagsFromResource"
      ],
      "Resource":"*"
    }
  ]
}
```

Die folgenden Tag-basierten Bedingungsschlüssel funktionieren nur mit administrativen Ressourcen in administrativen Richtlinienanweisungen.

Tag-based Schlüssel für administrative Bedingungen

Bedingungsschlüssel	Beschreibung	Typ
aws:RequestTag/${TagKey}	Filtert den Zugriff basierend auf dem Vorhandensein von Tag-Schlüssel-Wert-Paaren in der Anforderung.	Zeichenfolge
aws:ResourceTag/${TagKey}	Filtert den Zugriff basierend auf den Tag-Schlüssel-Wert-Paaren, die der Ressource angefügt sind.	Zeichenfolge

Bedingungsschlüssel	Beschreibung	Typ
aws:TagKeys	Filtert den Zugriff basierend auf dem Vorhandensein von Tag-Schlüsseln in der Anforderung.	Zeichenfolge
rds:cluster-pg-tag/${TagKey}	Filtert den Zugriff nach dem Tag, das einer DB-Cluster-Parametergruppe angefügt ist.	Zeichenfolge
rds:cluster-snapshot-tag/${TagKey}	Filtert den Zugriff nach dem Tag, das einem DB-Cluster-Snapshot angefügt ist.	Zeichenfolge
rds:cluster-tag/${TagKey}	Filtert den Zugriff nach dem Tag, das einem DB-Cluster angefügt ist.	Zeichenfolge
rds:db-tag/${TagKey}	Filtert den Zugriff nach dem Tag, das einer DB-Instance angefügt ist.	Zeichenfolge
rds:es-tag/${TagKey}	Filtert den Zugriff nach dem Tag, das einem Ereignisabonnement angefügt ist.	Zeichenfolge
rds:pg-tag/${TagKey}	Filtert den Zugriff nach dem Tag, das einer DB-Parametergruppe angefügt ist.	Zeichenfolge
rds:req-tag/${TagKey}	Filtert den Zugriff nach dem Satz von Tag-Schlüsseln und -Werten, die zum Taggen einer Ressource verwendet werden können.	Zeichenfolge
rds:secgrp-tag/${TagKey}	Filtert den Zugriff nach dem Tag, das einer DB-Sicherheitsgruppe angefügt ist.	Zeichenfolge
rds:snapshot-tag/${TagKey}	Filtert den Zugriff nach dem Tag, das einem DB-Snapshot angefügt ist.	Zeichenfolge

Bedingungsschlüssel	Beschreibung	Typ
`rds:subgrp-tag/${TagKey}`	Filtert den Zugriff nach dem Tag, das einer DB-Subnetzgruppe angefügt ist.	Zeichenfolge

Erstellung von IAM-Verwaltungsrichtlinienerklärungen für Amazon Neptune

Beispiele für allgemeine administrative Richtlinien

Die folgenden Beispiele zeigen die Erstellung administrativer Neptune-Richtlinien, die Berechtigungen für verschiedene Verwaltungsaktionen in einem DB-Cluster gewähren.

Richtlinie, die das Löschen einer angegebenen DB-Instance durch IAM-Benutzer verhindert

Dies ist ein Beispiel für eine Richtlinie, die das Löschen einer angegebenen Neptune-DB-Instance durch IAM-Benutzer verhindert:

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"
    }
  ]
}
```

Richtlinie, die Berechtigungen zum Erstellen neuer DB-Instances gewährt

Dies ist ein Beispiel für eine Richtlinie, die IAM-Benutzern die Erstellung von DB-Instances in einem angegebenen Neptune-DB-Cluster erlaubt:

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AllowCreateInstance",
      "Effect": "Allow",
      "Action": "rds:CreateDBInstance",
      "Resource": "arn:aws:rds:us-west-2:123456789012:cluster:my-cluster"
    }
  ]
}
```

Richtlinie, die Berechtigungen zum Erstellen neuer DB-Instances gewährt, die eine bestimmte DB-Parametergruppe verwenden

Dies ist ein Beispiel für eine Richtlinie, die IAM-Benutzern die Erstellung von DB-Instances in einem angegebenen Neptune-DB-Cluster (hier`us-west-2`) mittels einer einzelnen angegebenen DB-Parametergruppe erlaubt.

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"
      ]
    }
  ]
}
```

Richtlinie, die Berechtigungen zum Beschreiben einer Ressource gewährt

Dies ist ein Beispiel für eine Richtlinie, die IAM-Benutzern die Beschreibung jeder Neptune-Ressource erlaubt.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowDescribe",
            "Effect": "Allow",
            "Action": "rds:Describe*",
            "Resource": "*"
        }
    ]
}
```

Tag-based Beispiele für Verwaltungsrichtlinien

Die folgenden Beispiele zeigen die Erstellung administrativer Neptune-Richtlinien mit Tags, um die Berechtigungen für verschiedene Verwaltungsaktionen in einem DB-Cluster zu filtern.

Beispiel 1: Gewähren von Berechtigungen für Aktionen für eine Ressource mit einem benutzerdefinierten Tag, das mehrere Werte annehmen kann

Die folgende Richtlinie erlaubt die Verwendung der `ModifyDBInstance`-, `CreateDBInstance`- oder `DeleteDBInstance`-API für jede DB-Instance, deren `env`-Tag auf `dev` oder `test` festgelegt ist:

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"
        }
      }
    }
  ]
}
```

Beispiel 2: Einschränken des Satzes von Tag-Schlüsseln und -Werten, mit denen ein Tag für eine Ressource erstellt werden kann

Diese Richtlinie verwendet einen `Condition`-Schlüssel, um das Hinzufügen eines Tags mit dem Schlüssel `env` und dem Wert `test`, `qa` oder `dev` zu einer Ressource zu erlauben:

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"
        }
      }
    }
  ]
}
```

Beispiel 3: Erlauben Sie vollen Zugriff auf Neptune-Ressourcen basierend auf aws: ResourceTag

Die folgende Richtlinie ist dem ersten Beispiel oben ähnlich, verwendet jedoch stattdessen `aws:ResourceTag`:

JSON

```
{ "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AllowFullAccessToDev",
      "Effect": "Allow",
      "Action": [
        "rds:*"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/env": "dev",
          "rds:DatabaseEngine": "neptune"
        }
      }
    }
  ]
}
```

Erstellen benutzerdefinierter IAM-Richtlinienerklärungen für den Zugriff auf Daten in Amazon Neptune

Neptune-Datenzugriff-Richtlinienanweisungen verwenden Datenzugriffsaktionen, Ressourcen und Bedingungsschlüssel, denen jeweils das Präfix `neptune-db:` vorangestellt ist.

Themen

- Verwenden von Abfrageaktionen in Neptune-Datenzugriff-Richtlinienanweisungen
- IAM-Aktionen für den Datenzugriff in Amazon Neptune
- IAM-Ressourcentypen für den Zugriff auf Daten in Amazon Neptune
- IAM-Bedingungsschlüssel für den Zugriff auf Daten in Amazon Neptune
- Erstellen von IAM-Datenzugriffsrichtlinien in Amazon Neptune

Verwenden von Abfrageaktionen in Neptune-Datenzugriff-Richtlinienanweisungen

Es gibt drei Neptune-Abfrageaktionen, die in Datenzugriff-Richtlinienanweisungen verwendet werden können, `ReadDataViaQuery`, `WriteDataViaQuery` und `DeleteDataViaQuery`. Eine bestimmte Abfrage benötigt möglicherweise Berechtigungen, um mehr als eine dieser Aktionen auszuführen, und es möglicherweise nicht immer offensichtlich, welche Kombination dieser Aktionen zulässig sein muss, um eine Abfrage auszuführen.

Neptune ermittelt vor der Ausführung einer Abfrage die Berechtigungen, die für die Ausführung der einzelnen Schritte der Abfrage erforderlich sind, und kombiniert diese zu einem vollständigen Satz der Berechtigungen, die für die Abfrage erforderlich sind. Beachten Sie, dass dieser vollständige Satz von Berechtigungen alle Aktionen enthält, die die Abfrage möglicherweise ausführt. Dabei handelt es sich nicht notwendigerweise um den Satz von Aktionen, den die Abfrage tatsächlich für Ihre Daten ausführt.

Das bedeutet, dass Sie Berechtigungen für alle Aktionen bereitstellen müssen, die die Abfrage möglicherweise ausführt, um die Ausführung einer bestimmten Abfrage zu ermöglichen, unabhängig davon, ob die Abfrage diese Aktionen tatsächlich ausführt oder nicht.

Dies sind einige Beispiele für Gremlin-Abfragen, die dies im Detail zeigen:

- ```
  g.V().count()
  ```
  ```

`g.V()` und `count()` benötigen lediglich Lesezugriff, sodass die Abfrage insgesamt lediglich einen `ReadDataViaQuery`-Zugriff benötigt.

- ```
  g.addV()
  ```

  `addV()` muss überprüfen, ob ein Eckpunkt mit einer bestimmten ID vorhanden ist oder nicht, bevor ein neuer Eckpunkt eingefügt wird. Das bedeutet, dass die Abfrage einen `ReadDataViaQuery`- und einen `WriteDataViaQuery`-Zugriff benötigt.

- ```
  g.V('1').as('a').out('created').addE('createdBy').to('a')
  ```

  `g.V('1').as('a')` und `out('created')` benötigen lediglich Lesezugriff. `addE().from('a')` benötigt jedoch sowohl Lese- als auch Schreibzugriff, da `addE()` die Eckpunkte `to` und `from` lesen muss und prüfen muss, ob bereits eine Kante mit derselben ID vorhanden ist, bevor eine neue Kante hinzugefügt wird. Die Abfrage benötigt daher insgesamt einen `ReadDataViaQuery`- und einen `WriteDataViaQuery`-Zugriff.

- ```
  g.V().drop()
  ```

  `g.V()` benötigt lediglich Lesezugriff. `drop()` benötigt einen Lese- und Löschzugriff, da Eckpunkte oder Kanten vor dem Löschen gelesen werden müssen. Daher benötigt die Abfrage insgesamt einen `ReadDataViaQuery`- und einen `DeleteDataViaQuery`-Zugriff.

- ```
  g.V('1').property(single, 'key1', 'value1')
  ```

  `g.V('1')` benötigt lediglich Lesezugriff. `property(single, 'key1', 'value1')` benötigt jedoch einen Lese-, Schreib- und Löschzugriff. Hier werden im Schritt `property()` Schlüssel und Wert eingefügt, falls noch nicht im Eckpunkt vorhanden. Falls jedoch bereits vorhanden, wird der vorhandene Eigenschaftswert gelöscht und es wird ein neuer Wert eingefügt. Daher benötigt die Abfrage insgesamt einen `ReadDataViaQuery`-, `WriteDataViaQuery`- und `DeleteDataViaQuery`-Zugriff.

  Jede Abfrage, die den Schritt `property()` enthält, benötigt `ReadDataViaQuery`-, `WriteDataViaQuery`- und `DeleteDataViaQuery`-Berechtigungen.

Dies sind einige openCypher-Beispiele:

- ```
  MATCH (n)
  RETURN n
  ```

  Diese Abfrage liest alle Knoten in der Datenbank und gibt sie zurück. Hierfür wird lediglich ein `ReadDataViaQuery`-Zugriff benötigt.

- ```
  MATCH (n:Person)
  SET n.dept = 'AWS'
  ```

  Diese Abfrage benötigt einen `ReadDataViaQuery`-, `WriteDataViaQuery`- und `DeleteDataViaQuery`-Zugriff. Sie liest alle Knoten mit der Bezeichnung 'Person' und fügt ihnen entweder eine neue Eigenschaft mit dem Schlüssel `dept` und dem Wert `AWS` hinzu oder (wenn die Eigenschaft `dept` bereits vorhanden ist) löscht den alten Wert und fügt stattdessen `AWS` ein. Wenn der festzulegende Wert `null` ist, löscht `SET` die Eigenschaft vollständig.

  Da die `SET`-Klausel in einigen Fällen möglicherweise einen vorhandenen Wert löschen muss, benötigt sie stets `DeleteDataViaQuery`-, `ReadDataViaQuery`- und `WriteDataViaQuery`-Berechtigungen.

- ```
  MATCH (n:Person)
  DETACH DELETE n
  ```

  Diese Abfrage benötigt `ReadDataViaQuery`- und `DeleteDataViaQuery`-Berechtigungen. Sie sucht alle Knoten mit der Bezeichnung `Person` und löscht sie zusammen mit den Kanten, die mit diesen Knoten verbunden sind, und allen zugehörigen Bezeichnungen und Eigenschaften.

- ```
  MERGE (n:Person {name: 'John'})-[:knows]->(:Person {name: 'Peter'})
  RETURN n
  ```

  Diese Abfrage benötigt `ReadDataViaQuery`- und `WriteDataViaQuery`-Berechtigungen.
  Die `MERGE`-Klausel stimmt entweder mit einem angegebenen Muster überein oder erstellt es.
  Da ein Schreibvorgang erfolgen kann, wenn das Muster nicht übereinstimmt, sind Schreib- und Leseberechtigungen erforderlich.

IAM-Aktionen für den Datenzugriff in Amazon Neptune

Beachten Sie, dass Neptun-Datenzugriffsaktionen das Präfix `neptune-db:` besitzen, während administrative Aktionen in Neptune das Präfix `rds:` besitzen.

Der Amazon-Ressourcenname (ARN) für eine Datenressource in IAM ist nicht mit dem ARN identisch, der dem Cluster bei der Erstellung zugeordnet wird. Sie müssen den ARN wie in Angeben von Datenressourcen beschrieben konstruieren. Diese Datenressourcen-ARNs können Platzhalter verwenden, um mehrere Ressourcen einzuschließen.

Data-access Richtlinienerklärungen können auch den Bedingungsschlüssel neptune-db:QueryLanguage enthalten, um den Zugriff anhand der Abfragesprache einzuschränken.

Ab Release: 1.2.0.0 (21.07.2022) unterstützt Neptune die Einschränkung von Berechtigungen auf eine oder mehrere bestimmte Neptune-Aktionen. Dies ermöglicht eine detailliertere Zugriffssteuerung als bisher möglich.

Important

- Die Anwendung von Änderungen einer IAM-Richtlinie auf die angegebenen Neptune-Ressourcen kann bis zu 10 Minuten dauern.
- IAM-Richtlinien, die auf einen Neptune-DB-Cluster angewendet werden, werden auf alle Instances in diesem Cluster angewendet.

Query-based Aktionen für den Datenzugriff

Note

Es ist nicht immer offensichtlich, welche Berechtigungen für die Ausführung einer bestimmten Abfrage benötigt werden, da Abfragen abhängig von den verarbeiteten Daten mehr als eine Aktion ausführen können. Weitere Informationen finden Sie unter Verwenden von Abfrageaktionen.

`Neptun-DB: ReadDataViaQuery`

`ReadDataViaQuery` ermöglicht Benutzern das Lesen von Daten aus der Neptune-Datenbank mittels Abfragen.

Aktionsgruppen: Schreibgeschützt, Lesen und Schreiben.

Aktionskontextschlüssel: `neptune-db:QueryLanguage`.

Erforderliche Ressourcen: Datenbank.

Neptun-db: WriteDataViaQuery

`WriteDataViaQuery` ermöglicht Benutzern das Schreiben von Daten zur Neptune-Datenbank mittels Abfragen.

Aktionsgruppen: Lesen und Schreiben.

Aktionskontextschlüssel: `neptune-db:QueryLanguage`.

Erforderliche Ressourcen: Datenbank.

Neptun-db: DeleteDataViaQuery

`DeleteDataViaQuery` ermöglicht Benutzern das Löschen von Daten aus der Neptune-Datenbank mittels Abfragen.

Aktionsgruppen: Lesen und Schreiben.

Aktionskontextschlüssel: `neptune-db:QueryLanguage`.

Erforderliche Ressourcen: Datenbank.

Neptun-db: GetQueryStatus

`GetQueryStatus` ermöglicht Benutzern die Prüfung des Status aller aktiven Abfragen.

Aktionsgruppen: Schreibgeschützt, Lesen und Schreiben.

Aktionskontextschlüssel: `neptune-db:QueryLanguage`.

Erforderliche Ressourcen: Datenbank.

Neptun-db: GetStreamRecords

`GetStreamRecords` ermöglicht Benutzern das Abrufen von Stream-Datensätzen aus Neptune.

Aktionsgruppen: Lesen und Schreiben.

Aktionskontextschlüssel: `neptune-db:QueryLanguage`.

Erforderliche Ressourcen: Datenbank.

Neptun-db: CancelQuery

`CancelQuery` ermöglicht Benutzern das Abbrechen von Abfragen.

Aktionsgruppen: Lesen und Schreiben.

Erforderliche Ressourcen: Datenbank.

Allgemeine Datenzugriffsaktionen

Neptun-db: GetEngineStatus

`GetEngineStatus` ermöglicht Benutzern die Prüfung des Status der Neptune-Engine.

Aktionsgruppen: Schreibgeschützt, Lesen und Schreiben.

Erforderliche Ressourcen: Datenbank.

Neptun-db: GetStatisticsStatus

`GetStatisticsStatus` ermöglicht Benutzern die Prüfung des Status der Statistiken, die für die Datenbank gesammelt werden.

Aktionsgruppen: Schreibgeschützt, Lesen und Schreiben.

Erforderliche Ressourcen: Datenbank.

Neptun-db: GetGraphSummary

`GetGraphSummary` Die Diagrammübersichts-API ermöglicht Benutzern das Abrufen einer schreibgeschützten Übersicht über ein Diagramm.

Aktionsgruppen: Schreibgeschützt, Lesen und Schreiben.

Erforderliche Ressourcen: Datenbank.

Neptun-db: ManageStatistics

`ManageStatistics` ermöglicht Benutzern die Verwaltung der Erfassung von Statistiken für die Datenbank.

Aktionsgruppen: Lesen und Schreiben.

Erforderliche Ressourcen: Datenbank.

Neptun-db: DeleteStatistics

`DeleteStatistics` ermöglicht Benutzern das Löschen aller Statistiken in der Datenbank.

Aktionsgruppen: Lesen und Schreiben.

Erforderliche Ressourcen: Datenbank.

Neptun-db: ResetDatabase

`ResetDatabase` ermöglicht Benutzern das Abrufen des für eine Zurücksetzung benötigten Tokens und das Zurücksetzen der Neptune-Datenbank.

Aktionsgruppen: Lesen und Schreiben.

Erforderliche Ressourcen: Datenbank.

Bulk-loader Aktionen für den Datenzugriff

Neptun-DB: StartLoaderJob

`StartLoaderJob` ermöglicht Benutzern das Starten eines Massen-Loader-Auftrags.

Aktionsgruppen: Lesen und Schreiben.

Erforderliche Ressourcen: Datenbank.

Neptun-db: GetLoaderJobStatus

`GetLoaderJobStatus` ermöglicht Benutzern das Prüfen des Status eines Massen-Loader-Auftrags.

Aktionsgruppen: Schreibgeschützt, Lesen und Schreiben.

Erforderliche Ressourcen: Datenbank.

Neptun-db: ListLoaderJobs

`ListLoaderJobs` ermöglicht Benutzern das Auflisten aller Massen-Loader-Aufträge.

Aktionsgruppen: Nur Auflistung, Schreibgeschützt, Lesen und Schreiben.

Erforderliche Ressourcen: Datenbank.

Neptun-db: CancelLoaderJob

`CancelLoaderJob` ermöglicht Benutzern das Abbrechen von Loader-Aufträgen.

Aktionsgruppen: Lesen und Schreiben.

Erforderliche Ressourcen: Datenbank.

Machine-learning Aktionen für den Datenzugriff

Neptun-DB: StartMLDataProcessingJob

`StartMLDataProcessingJob` ermöglicht Benutzern das Starten von Neptune-ML-Datenverarbeitungsaufträgen.

Aktionsgruppen: Lesen und Schreiben.

Erforderliche Ressourcen: Datenbank.

Neptun-db: StartMLModelTrainingJob

`StartMLModelTrainingJob` ermöglicht Benutzern das Starten von ML-Modelltrainingsaufträgen.

Aktionsgruppen: Lesen und Schreiben.

Erforderliche Ressourcen: Datenbank.

Neptun-db: StartMLModelTransformJob

`StartMLModelTransformJob` ermöglicht Benutzern das Starten von ML-Modelltransformierungsaufträgen.

Aktionsgruppen: Lesen und Schreiben.

Erforderliche Ressourcen: Datenbank.

neptune-db: ML-Endpunkt erstellen

`CreateMLEndpoint` ermöglicht Benutzer das Erstellen von Neptune-ML-Endpunkten.

Aktionsgruppen: Lesen und Schreiben.

Erforderliche Ressourcen: Datenbank.

neptune-db: GetMLDataProcessingJobStatus

`GetMLDataProcessingJobStatus` ermöglicht Benutzern das Prüfen von Neptune-ML-Datenverarbeitungsaufträgen.

Aktionsgruppen: Schreibgeschützt, Lesen und Schreiben.

Erforderliche Ressourcen: Datenbank.

Neptun-db: GetMLModelTrainingJobStatus

`GetMLModelTrainingJobStatus` ermöglicht Benutzern das Prüfen des Status von Neptune-ML-Modelltrainingsaufträgen.

Aktionsgruppen: Schreibgeschützt, Lesen und Schreiben.

Erforderliche Ressourcen: Datenbank.

Neptun-db: GetMLModelTransformJobStatus

`GetMLModelTransformJobStatus` ermöglicht Benutzern das Prüfen des Status von Neptune-ML-Modelltransformierungsaufträgen.

Aktionsgruppen: Schreibgeschützt, Lesen und Schreiben.

Erforderliche Ressourcen: Datenbank.

Neptun-db: GetMLEndpointStatus

`GetMLEndpointStatus` ermöglicht Benutzern das Prüfen des Status von Neptune-ML-Endpunkten.

Aktionsgruppen: Schreibgeschützt, Lesen und Schreiben.

Erforderliche Ressourcen: Datenbank.

Neptun-db: ListMLDataProcessingJobs

`ListMLDataProcessingJobs` ermöglicht Benutzern das Auflisten aller Neptune-ML-Datenverarbeitungsaufträge.

Aktionsgruppen: Nur Auflistung, Schreibgeschützt, Lesen und Schreiben.

Erforderliche Ressourcen: Datenbank.

Neptun-db: ListMLModelTrainingJobs

`ListMLModelTrainingJobs` ermöglicht Benutzern das Auflisten aller Neptune-ML-Modelltrainingsaufträge.

Aktionsgruppen: Nur Auflistung, Schreibgeschützt, Lesen und Schreiben.

Erforderliche Ressourcen: Datenbank.

Neptun-db: ListMLModelTransformJobs

`ListMLModelTransformJobs` ermöglicht Benutzern das Auflisten aller Neptune-ML-Modelltransformierungsaufträge.

Aktionsgruppen: Nur Auflistung, Schreibgeschützt, Lesen und Schreiben.

Erforderliche Ressourcen: Datenbank.

neptune-db: Listet ML-Endpunkte auf

`ListMLEndpoints` ermöglicht Benutzern das Auflisten aller Neptune-ML-Endpunkte.

Aktionsgruppen: Nur Auflistung, Schreibgeschützt, Lesen und Schreiben.

Erforderliche Ressourcen: Datenbank.

neptun-DB: CancelMLDataProcessingJob

`CancelMLDataProcessingJob` ermöglicht Benutzern das Abbrechen von Neptune-ML-Datenverarbeitungsaufträgen.

Aktionsgruppen: Lesen und Schreiben.

Erforderliche Ressourcen: Datenbank.

Neptun-db: CancelMLModelTrainingJob

`CancelMLModelTrainingJob` ermöglicht Benutzern das Abbrechen von Neptune-ML-Modelltrainingsaufträgen.

Aktionsgruppen: Lesen und Schreiben.

Erforderliche Ressourcen: Datenbank.

Neptun-db: CancelMLModelTransformJob

`CancelMLModelTransformJob` ermöglicht Benutzern das Abbrechen von Neptune-ML-Modelltransformierungsaufträgen.

Aktionsgruppen: Lesen und Schreiben.

Erforderliche Ressourcen: Datenbank.

`neptune-db: Löschen Sie den ML-Endpunkt`

`DeleteMLEndpoint` ermöglicht Benutzer das Löschen von Neptune-ML-Endpunkten.

Aktionsgruppen: Lesen und Schreiben.

Erforderliche Ressourcen: Datenbank.

IAM-Ressourcentypen für den Zugriff auf Daten in Amazon Neptune

Datenressourcen besitzen wie Datenaktionen das Präfix `neptune-db:`.

In einer Neptune-Datenzugriffsrichtlinie geben Sie den DB-Cluster an, auf den Sie Zugriff gewähren, im ARN im folgenden Format an:

```
arn:aws:neptune-db:region:account-id:cluster-resource-id/*
```

Dieser Ressourcen-ARN enthält die folgenden Teile:

- *`region`*ist die AWS Region für den Amazon Neptune DB-Cluster.
- *`account-id`* ist die Nummer des AWS -Kontos für den DB-Cluster.
- *`cluster-resource-id`* ist eine Ressourcen-ID für den DB-Cluster.

> **Important**
>
> Die `cluster-resource-id` unterscheidet sich von der Cluster-ID. Um eine Cluster-Ressourcen-ID im Neptune zu finden AWS-Managementkonsole, suchen Sie im Abschnitt Konfiguration nach dem betreffenden DB-Cluster.

IAM-Bedingungsschlüssel für den Zugriff auf Daten in Amazon Neptune

Mithilfe von Bedingungsschlüsseln können Sie Bedingungen in einer IAM-Richtlinienanweisung angeben, damit die Anweisung nur dann wirksam wird, wenn die Bedingungen erfüllt werden.

Die Bedingungsschlüssel, die Sie in Datenzugriff-Richtlinienanweisungen in Neptune verwenden können, gehören den folgenden Kategorien an:

- Globale Bedingungsschlüssel — Die Teilmenge der AWS globalen Bedingungsschlüssel, die Neptune in Richtlinienerklärungen für den Datenzugriff unterstützt, ist unten aufgeführt.

- Service-specific Bedingungsschlüssel — Dies sind Schlüssel, die von Neptune speziell für die Verwendung in Datenzugriffspolitikanweisungen definiert wurden. Gegenwärtig gibt es nur einen, neptune-db:, der Zugriff nur gewährtQueryLanguage, wenn eine bestimmte Abfragesprache verwendet wird.

AWS Globale Bedingungskontextschlüssel, die von Neptune in Datenzugriffspolitikanweisungen unterstützt werden

Die folgende Tabelle listet die Teilmenge der globalen AWS -Bedingungskontextschlüssel auf, die Amazon Neptune für die Verwendung in Datenzugriff-Richtlinienanweisungen unterstützt:

Globale Bedingungsschlüssel, die Sie in Datenzugriff-Richtlinienanweisungen verwenden können

Bedingungsschlüssel	Beschreibung	Typ
aws:CurrentTime	Filtert den Zugriff nach aktuellen Datum und aktueller Uhrzeit der Anforderung.	String
aws:EpochTime	Filtert den Zugriff nach Datum und Uhrzeit der Anforderung ausgedrückt als UNIX-Epochenwert.	Numeric
aws:PrincipalAccount	Filtert den Zugriff nach dem Konto, zu dem der anfordernde Prinzipal gehört.	String
aws:PrincipalArn	Filtert den Zugriff nach dem ARN des Prinzipals, der die Anforderung gesendet hat.	String
aws:PrincipalIsAWSService	Erlaubt den Zugriff nur, wenn der Anruf direkt von einem AWS Dienstprinzipal getätigt wird.	Boolean
aws:PrincipalOrgID	Filtert den Zugriff nach der ID der Organisation in AWS Organizations, zu denen der anfragende Principal gehört.	String
aws:PrincipalOrgPaths	Filtert den Zugriff nach dem AWS Organisationspfad für den Prinzipal, der die Anfrage stellt.	String

Bedingungsschlüssel	Beschreibung	Typ
aws:PrincipalTag	Filtert den Zugriff nach einem Tag, das dem anfordernden Prinzipal angefügt ist.	String
aws:PrincipalType	Filtert den Zugriff nach dem Typ des Prinzipals, der die Anforderung sendet.	String
aws:RequestedRegion	Filtert den Zugriff nach der AWS Region, die in der Anfrage aufgerufen wurde.	String
aws:SecureTransport	Gewährt den Zugriff nur, wenn die Anforderung über SSL gesendet wurde.	Boolean
aws:SourceIp	Filtert den Zugriff nach der IP-Adresse des Anforderers.	String
aws:TokenIssueTime	Filtert den Zugriff nach Datum und Uhrzeit der Ausgabe temporärer Sicherheitsanmeldeinformationen.	String
aws:UserAgent	Filtert den Zugriff nach der Client-Anwendung des Anforderers.	String
aws:userid	Filtert den Zugriff nach der Prinzipal–ID des Anforderers.	String
aws:ViaAWSService	Ermöglicht den Zugriff nur, wenn ein AWS Dienst die Anfrage in Ihrem Namen gestellt hat.	Boolean

Servicespezifische Neptune-Bedingungsschlüssel

Neptune unterstützt den folgenden servicespezifischen Bedingungsschlüssel für IAM-Richtlinien:

Servicespezifische Neptune-Bedingungsschlüssel

Bedingungsschlüssel	Beschreibung	Typ
`neptune-db:QueryLanguage`	Filtert den Datenzugriff nach der verwendeten Abfragesprache. Gültige Werte sind: `Gremlin`, `OpenCypher` und `Sparql`. Unterstützte Aktionen sind `ReadDataViaQuery`, `WriteDataViaQuery`, `DeleteDataViaQuery`, `GetQueryStatus` und `CancelQuery`.	`String`

Erstellen von IAM-Datenzugriffsrichtlinien in Amazon Neptune

Die folgenden Beispiele zeigen die Erstellung benutzerdefinierter IAM-Richtlinien, die eine differenzierte Zugriffssteuerung für APIs und Aktionen auf Datenebene verwenden, wie in Neptune-Engine-Version 1.2.0.0 eingeführt.

Beispiel für eine Richtlinie, die den uneingeschränkten Zugriff auf die Daten in einem Neptune-DB-Cluster ermöglicht

Das folgende Beispiel für eine Richtlinie ermöglicht IAM-Benutzern die Herstellung von Verbindungen mit Neptune-DB-Clustern mittels IAM-Datenbank-Authentifizierung und verwendet das Zeichen „*“, um alle verfügbaren Aktionen abzugleichen.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "neptune-db:*",
      "Resource": "arn:aws:neptune-db:us-east-1:123456789012:cluster-ABCD1234EFGH5678IJKL90MNOP/*"
    }
```

```
  ]
}
```

Das vorhergehende Beispiel enthält einen Ressourcen-ARN in einem Format, das für die Neptune-IAM-Authentifizierung spezifisch ist. Informationen zum Konstruieren des ARN finden Sie unter Angeben von Datenressourcen. Beachten Sie, dass der ARN für eine IAM-Autorisierung `Resource` nicht mit dem ARN identisch ist, der dem Cluster bei Erstellung zugewiesen wurde.

Beispiel für eine Richtlinie, die einen schreibgeschützten Zugriff auf ein Neptune-DB-Cluster gewährt

Die folgende Richtlinie gewährt die Berechtigung für einen schreibgeschützten Vollzugriff auf Daten in einem Neptune-DB-Cluster:

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/*"
    }
  ]
}
```

Beispiel für eine Richtlinie, die jede Art von Zugriff auf ein Neptune-DB-Cluster ablehnt

Die IAM-Standardaktion ist die Ablehnung des Zugriffs auf einen DB-Cluster, wenn der Effekt `Allow` nicht gewährt wird. Die folgende Richtlinie verweigert jedoch jeglichen Zugriff auf einen DB-Cluster für ein bestimmtes AWS Konto und eine bestimmte Region, was dann Vorrang vor allen Auswirkungen hat. `Allow`

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": "neptune-db:*",
      "Resource": "arn:aws:neptune-db:us-east-1:123456789012:cluster-
ABCD1234EFGH5678IJKL90MNOP/*"
    }
  ]
}
```

Beispiel für eine Richtlinie, die Lesezugriff über Abfragen gewährt

Die folgende Richtlinie gewährt nur die Berechtigung zum Lesen aus einem Neptune-DB-Cluster über eine Abfrage:

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "neptune-db:ReadDataViaQuery",
      "Resource": "arn:aws:neptune-db:us-east-1:123456789012:cluster-
ABCD1234EFGH5678IJKL90MNOP/*"
    }
  ]
}
```

Beispiel für eine Richtlinie, die nur Gremlin-Abfragen zulässt

Die folgende Richtlinie verwendet den Bedingungsschlüssel `neptune-db:QueryLanguage`, um die Berechtigung zu gewähren, Neptune nur mit der Gremlin-Abfragesprache abzufragen:

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"
        }
      }
    }
  ]
}
```

Beispiel für eine Richtlinie, die jeden Zugriff außer dem Zugriff auf die Neptune-ML-Modellverwaltung gewährt

Die folgende Richtlinie gewährt Vollzugriff auf alle Neptune-Diagrammoperationen außer Neptune-ML-Modellverwaltungs-Features:

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/*"
    }
  ]
}
```

Beispiel für eine Richtlinie, die Zugriff auf die Neptune-ML-Modellverwaltung gewährt

Diese Richtlinie gewährt Zugriff auf die Neptune-ML-Modellverwaltungs-Features:

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/*"
    }
  ]
}
```

Beispiel für eine Richtlinie, die vollständigen Abfragezugriff gewährt

Die folgende Richtlinie gewährt Vollzugriff auf Neptune-Diagramm-Abfrageoperationen, jedoch nicht auf Features wie Fast Reset, Streams, Massen-Loader, Neptune-ML-Modellverwaltung usw.:

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/*"
    }
  ]
}
```

Beispiel für eine Richtlinie, die nur für Gremlin-Abfragen Vollzugriff gewährt

Die folgende Richtlinie gewährt Vollzugriff auf Neptune-Diagramm-Abfrageoperationen über die Gremlin-Abfragesprache, jedoch nicht auf Abfragen über andere Sprachen oder auf Features wie Fast Reset, Streams, Massen-Loader, Neptune-ML-Modellverwaltung usw.:

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"
        }
      }
    }
  ]
}
```

Beispiel für eine Richtlinie, die Vollzugriff gewährt, ausgenommen Fast Reset

Die folgende Richtlinie gewährt Vollzugriff auf einen Neptune-DB-Cluster, ausgenommen bei Verwendung von Fast Reset:

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/*"
    }
  ]
}
```

Verwenden von serviceverknüpften Rollen für Amazon Neptune

Amazon Neptune verwendet AWS Identity and Access Management (IAM) serviceverknüpfte Rollen. Eine serviceverknüpfte Rolle ist eine einzigartige Art von IAM-Rolle, die direkt mit Neptune verknüpft ist. Service-linked Rollen sind von Neptune vordefiniert und beinhalten alle Berechtigungen, die der Dienst benötigt, um andere AWS Dienste in Ihrem Namen aufzurufen.

> **Important**
>
> Für bestimmte Verwaltungs-Features verwendet Amazon Neptune operative Technologien, die mit Amazon RDS geteilt werden. Dies umfasst die serviceverknüpfte Rolle und API-Verwaltungsberechtigungen.

Eine serviceverknüpfte Rolle vereinfacht das Einrichten von Neptune, da Sie die erforderlichen Berechtigungen nicht manuell hinzufügen müssen. Neptune definiert die Berechtigungen seiner serviceverknüpften Rollen. Wenn nicht anders definiert, kann Neptune nur seine Rollen übernehmen. Die definierten Berechtigungen umfassen die Vertrauens- und Berechtigungsrichtlinie. Diese Berechtigungsrichtlinie kann keinen anderen IAM-Entitäten zugewiesen werden.

Sie können die Rollen nur nach dem Löschen der zugehörigen Ressourcen löschen. Dies schützt Ihre Neptune-Ressourcen, da Sie nicht versehentlich die Berechtigung für den Zugriff auf die Ressourcen entfernen können.

Informationen zu anderen Diensten, die dienstbezogene Rollen unterstützen, finden Sie unter AWS Dienste, die mit IAM funktionieren. Suchen Sie dort in der Spalte Rolle nach den Diensten, für die Ja steht. Service-Linked Wählen Sie über einen Link Ja aus, um die Dokumentation zu einer serviceverknüpften Rolle für diesen Service anzuzeigen.

Service-Linked Rollenberechtigungen für Neptune

Neptune verwendet die `AWSServiceRoleForRDS` serviceverknüpfte Rolle, damit Neptune und Amazon RDS AWS Dienste im Namen Ihrer Datenbank-Instances aufrufen können. Die serviceverknüpfte Rolle `AWSServiceRoleForRDS` vertraut dem Service `rds.amazonaws.com`, sodass dieser die Rolle annehmen kann.

Die Rollenberechtigungsrichtlinie erlaubt Neptune die Durchführung der folgenden Aktionen für die angegebenen Ressourcen:

- Aktionen auf `ec2`:
  - `AssignPrivateIpAddresses`
  - `AuthorizeSecurityGroupIngress`
  - `CreateNetworkInterface`
  - `CreateSecurityGroup`
  - `DeleteNetworkInterface`
  - `DeleteSecurityGroup`
  - `DescribeAvailabilityZones`
  - `DescribeInternetGateways`
  - `DescribeSecurityGroups`
  - `DescribeSubnets`
  - `DescribeVpcAttribute`
  - `DescribeVpcs`
  - `ModifyNetworkInterfaceAttribute`
  - `RevokeSecurityGroupIngress`
  - `UnassignPrivateIpAddresses`

- Aktionen auf `sns`:
  - `ListTopic`
  - `Publish`
- Aktionen auf `cloudwatch`:
  - `PutMetricData`
  - `GetMetricData`
  - `CreateLogStream`
  - `PullLogEvents`
  - `DescribeLogStreams`
  - `CreateLogGroup`

> **Note**
>
> Sie müssen Berechtigungen konfigurieren, damit eine juristische Stelle von IAM (z. B. Benutzer, Gruppe oder Rolle) eine serviceverknüpfte Rolle erstellen, bearbeiten oder löschen kann. Möglicherweise erhalten Sie die folgende Fehlermeldung:
> Unable to create the resource. Überprüfen Sie, ob Sie die Berechtigung haben, eine serviceverknüpfte Rolle zu erstellen. Andernfalls warten Sie und versuchen Sie es später noch einmal.
> Wenn Sie diesen Fehler erhalten, überprüfen Sie, ob die folgenden Berechtigungen aktiviert sind:
>
> ```
> {
>     "Action": "iam:CreateServiceLinkedRole",
>     "Effect": "Allow",
>     "Resource": "arn:aws:iam::*:role/aws-service-role/rds.amazonaws.com/
> AWSServiceRoleForRDS",
>     "Condition": {
>         "StringLike": {
>             "iam:AWSServiceName":"rds.amazonaws.com"
>         }
>     }
> }
> ```

Weitere Informationen finden Sie unter Service-Linked Rollenberechtigungen im IAM-Benutzerhandbuch.

Eine Service-Linked Rolle für Neptune erstellen

Sie müssen eine serviceverknüpfte Rolle nicht manuell erstellen. Wenn Sie eine Instance oder einen Cluster erstellen, erstellt Neptune die serviceverknüpfte Rolle für Sie.

Important

Weitere Informationen finden Sie unter In meinem IAM-Konto wird eine neue Rolle angezeigt im IAM-Benutzerhandbuch.

Wenn Sie diese serviceverknüpfte Rolle löschen und dann erneut erstellen müssen, können Sie die Rolle in Ihrem Konto mit demselben Verfahren neu anlegen. Neptune erstellt die serviceverknüpfte Rolle erneut für Sie, wenn Sie eine Instance oder einen Cluster erstellen.

Eine Service-Linked Rolle für Neptune bearbeiten

Neptune erlaubt Ihnen nicht, die serviceverknüpfte Rolle `AWSServiceRoleForRDS` zu bearbeiten. Da möglicherweise verschiedene Entitäten auf die Rolle verweisen, kann der Rollenname nach dem Erstellen einer serviceverknüpften Rolle nicht mehr geändert werden. Sie können jedoch die Beschreibung der Rolle mit IAM bearbeiten. Weitere Informationen finden Sie unter Bearbeiten einer Service-Linked Rolle im IAM-Benutzerhandbuch.

Eine Service-Linked Rolle für Neptune löschen

Wenn Sie ein Feature oder einen Dienst, die bzw. der eine serviceverknüpften Rolle erfordert, nicht mehr benötigen, sollten Sie diese Rolle löschen. Auf diese Weise haben Sie keine ungenutzte juristische Stelle, die nicht aktiv überwacht oder verwaltet wird. Sie müssen jedoch alle Ihre Instances und Cluster löschen, bevor Sie die verknüpfte serviceverknüpfte Rolle löschen können.

Eine Service-Linked Rolle vor dem Löschen bereinigen

Bevor Sie mit IAM eine serviceverknüpfte Rolle löschen können, müssen Sie sich zunächst vergewissern, dass die Rolle über keine aktiven Sitzungen verfügt, und alle Ressourcen entfernen, die von der Rolle verwendet werden.

So überprüfen Sie in der IAM-Konsole, ob die serviceverknüpfte Rolle über eine aktive Sitzung verfügt

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie im Navigationsbereich der IAM Console Roles (Rollen) aus. Wählen Sie dann den Namen (nicht das Kontrollkästchen) der Rolle `AWSServiceRoleForRDS` aus.
3. Wählen Sie auf der Seite Summary für die ausgewählte Rolle die Registerkarte Access Advisor.
4. Überprüfen Sie auf der Registerkarte Access Advisor die jüngsten Aktivitäten für die serviceverknüpfte Rolle.

> **Note**
>
> Wenn Sie sich nicht sicher sind, ob Neptune die Rolle `AWSServiceRoleForRDS` verwendet, können Sie versuchen, die Rolle zu löschen. Wenn der Service die Rolle verwendet, schlägt die Löschung fehl und Sie können die -Regionen anzeigen, in denen die Rolle verwendet wird. Wenn die Rolle verwendet wird, müssen Sie warten, bis die Sitzung beendet wird, bevor Sie die Rolle löschen können. Die Sitzung für eine serviceverknüpfte Rolle können Sie nicht widerrufen.

Wenn Sie die Rolle `AWSServiceRoleForRDS` entfernen möchten, müssen Sie zunächst alle Ihre Instances und Cluster löschen.

Löschen Ihrer kompletten Instances

Verwenden Sie eine dieser Verfahren, um Ihrer kompletten Instance zu löschen.

So löschen Sie eine Instance (Konsole)

1. Öffnen Sie die Amazon RDS-Konsole unter https://console.aws.amazon.com/rds/.
2. Wählen Sie im Navigationsbereich Instances aus.
3. Wählen Sie in der Liste Instances die Instance aus, die Sie löschen möchten.
4. Wählen Sie Instance-Aktionen und dann Löschen aus.
5. Wenn Sie die Aufforderung Create final Snapshot? (Finalen Snapshot erstellen), wählen Sie Yes (Ja) oder No (Nein) aus.

6. Wenn Sie im vorherigen Schritt Yes (Ja) gewählt haben, geben Sie unter Final snapshot name (Endgültiger Snapshot-Name) den Namen Ihres endgültigen DB-Snapshots ein.
7. Wählen Sie Löschen aus.

Um eine Instanz zu löschen (AWS CLI)

Siehe `delete-db-instance` in der AWS CLI -Befehlsreferenz.

So löschen Sie eine Instance (API)

Siehe `DeleteDBInstance`.

Löschen Ihrer kompletten Cluster

Verwenden Sie eines der folgenden Verfahren, um einen einzelnen Cluster zu löschen. Wiederholen Sie anschließend das Verfahren für jeden Ihrer Cluster.

So löschen Sie einen Cluster (Konsole)

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die Amazon Neptune Neptune-Konsole unter. https://console.aws.amazon.com/neptune/home
2. Wählen Sie in der Liste Clusters den Cluster aus, den Sie löschen möchten.
3. Wählen Sie Cluster Actions und dann Delete aus.
4. Wählen Sie Löschen aus.

So löschen Sie einen Cluster (CLI)

Siehe `delete-db-cluster` in der AWS CLI -Befehlsreferenz.

So löschen Sie einen Cluster (API)

Siehe `DeleteDBCluster`

Sie können die IAM-Konsole, die IAM-CLI oder die IAM-API verwenden, um die serviceverknüpfte Rolle `AWSServiceRoleForRDS` zu löschen. Weitere Informationen finden Sie unter Löschen einer Service-Linked Rolle im IAM-Benutzerhandbuch.

Temporäre Anmeldeinformationen verwenden, um eine Verbindung zu Amazon Neptune herzustellen

Amazon Neptune unterstützt die IAM-Authentifizierung mit temporären Anmeldeinformationen.

Sie können eine angenommene Rolle zur Authentifizierung mithilfe einer IAM-Authentifizierungsrichtlinie verwenden, etwa eine der Beispielrichtlinien in den vorherigen Abschnitten.

Wenn Sie temporäre Anmeldeinformationen verwenden, müssen Sie `AWS_SESSION_TOKEN` zusätzlich zu `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY` und `SERVICE_REGION` angeben.

> **Note**
>
> Die temporären Anmeldeinformationen laufen nach einem angegebenen Intervall ab, , einschließlich des Sitzungs-Tokens.
> Sie müssen Ihren Sitzungs-Token aktualisieren, wenn Sie neue Anmeldeinformationen anfragen. Weitere Informationen finden Sie unter Verwenden temporärer Sicherheitsanmeldedaten, um Zugriff auf Ressourcen anzufordern. AWS

In den folgenden Abschnitten wird beschrieben, wie Sie den Zugriff zulassen und temporäre Anmeldeinformationen abrufen.

So führen Sie die Authentifizierung mithilfe temporärer Anmeldeinformationen durch

1. Erstellen Sie eine IAM-Rolle mit Berechtigung für den Zugriff auf einen Neptune-Cluster. Weitere Informationen zum Erstellen dieser Rolle finden Sie unter the section called “Arten von IAM-Richtlinien”.
2. Fügen Sie eine Vertrauensstellung der Rolle hinzu, die den Zugriff auf die Anmeldeinformationen ermöglicht.

   Rufen Sie die temporären Anmeldeinformationen ab, einschließlich `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY` und `AWS_SESSION_TOKEN`.
3. Stellen Sie eine Verbindung mit dem Neptune-Cluster her und signieren Sie die Anforderungen mithilfe der temporären Anmeldeinformationen. Für weitere Informationen zu Verbindungen und zum Signieren von Anfragen finden Sie unter the section called “Verbindung mit IAM-Authentifizierung herstellen”.

Es gibt verschiedene Methoden zum Abrufen der temporären Anmeldeinformationen, die von der jeweiligen Umgebung abhängig sind.

Themen

- Abrufen temporärer Anmeldeinformationen mit dem AWS CLI
- Einrichten AWS Lambda für Neptune IAM-Authentifizierung
- Einrichten von Amazon EC2 für die Neptune-IAM-Authentifizierung

Abrufen temporärer Anmeldeinformationen mit dem AWS CLI

Um Anmeldeinformationen mithilfe von AWS Command Line Interface (AWS CLI) abzurufen, müssen Sie zunächst eine Vertrauensstellung hinzufügen, die dem AWS Benutzer, der den AWS CLI Befehl ausführt, die Berechtigung erteilt, die Rolle zu übernehmen.

Fügen Sie der Neptune-IAM-Authentifizierungsrolle die folgende Vertrauensstellung hinzu. Wenn Sie keine Neptune-IAM-Authentifizierungsrolle haben, siehe the section called “Arten von IAM-Richtlinien”.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:user/test"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Informationen zum Hinzufügen der Vertrauensstellung zur Rolle finden Sie unter Bearbeiten der Vertrauensstellung für eine vorhandene Rolle im AWS Directory Service Administrationshandbuch.

Wenn die Neptune-Richtlinie noch nicht an eine Rolle angefügt ist, erstellen Sie eine neue Rolle. Fügen Sie die Neptune-IAM-Authentifizierungsrichtlinie an und dann die Vertrauensrichtlinie hinzu. Informationen zum Erstellen einer neuen Rolle finden Sie unter Erstellen einer neuen Rolle.

> **Note**
>
> In den folgenden Abschnitten wird davon ausgegangen, dass Sie das AWS CLI installiert haben.

Um das auszuführen AWS CLI manuell

1. Geben Sie den folgenden Befehl ein, um die Anmeldeinformationen mithilfe der AWS CLI anzufordern. Ersetzen Sie den Rollen-ARN, den Namen der Sitzung und das Profil durch Ihre eigenen Werte.

```
aws sts assume-role  --role-arn  arn:aws:iam::123456789012:role/NeptuneIAMAuthRole
 --role-session-name test --profile testprofile
```

2. Das folgende Beispiel zeigt die Ausgabe des Befehls. Der Abschnitt `Credentials` enthält die Werte, die Sie benötigen.

> **Note**
>
> Notieren Sie den `Expiration`-Wert, da Sie ihn später zum Anfordern neuer Anmeldeinformationen benötigen.

```
{
    "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. Stellen Sie die Umgebungsvariablen anhand der zurückgegebenen Anmeldeinformationen ein.

```
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. Stellen Sie die Verbindung mit einem der folgenden Verfahren her.

- the section called “Von der Gremlin-Konsole aus eine Verbindung herstellen”
- the section called “Verbindung mit Gremlin Java herstellen”
- the section called “Verbindung mit SPARQL Java herstellen”
- the section called “Herstellen einer Verbindung mit Python”

So verwenden Sie ein Skript zum Abrufen der Anmeldeinformationen

1. Führen Sie den folgenden Befehl aus, um den jq-Befehl zu installieren. Das Skript verwendet diesen Befehl, um die Ausgabe des AWS CLI Befehls zu analysieren.

```
sudo yum -y install jq
```

2. Erstellen Sie in einem Texteditor eine Datei mit der Bezeichnung `credentials.sh`, und fügen Sie den folgenden Text hinzu. Ersetzen Sie die Service-Region, den Rollen-ARN, den Namen der Sitzung und das Profil durch Ihre eigenen Werte.

```
#!/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. Stellen Sie die Verbindung mit einem der folgenden Verfahren her.

   - the section called “Von der Gremlin-Konsole aus eine Verbindung herstellen”
   - the section called “Verbindung mit Gremlin Java herstellen”
   - the section called “Verbindung mit SPARQL Java herstellen”

- the section called “Herstellen einer Verbindung mit Python”

Einrichten AWS Lambda für Neptune IAM-Authentifizierung

AWS Lambda schließt bei jeder Ausführung der Lambda-Funktion automatisch Anmeldeinformationen ein.

Zunächst müssen Sie eine Vertrauensstellung hinzufügen, die dem Lambda-Service die Berechtigung zur Übernahme der Rolle gewährt.

Fügen Sie der Neptune-IAM-Authentifizierungsrolle die folgende Vertrauensstellung hinzu. Wenn Sie keine Neptune-IAM-Authentifizierungsrolle haben, siehe the section called “Arten von IAM-Richtlinien”.

JSON

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

Informationen zum Hinzufügen der Vertrauensstellung zur Rolle finden Sie unter Bearbeiten der Vertrauensstellung für eine vorhandene Rolle im AWS Directory Service Administrationshandbuch.

Wenn die Neptune-Richtlinie noch nicht an eine Rolle angefügt ist, erstellen Sie eine neue Rolle. Fügen Sie die Neptune-IAM-Authentifizierungsrichtlinie an und dann die Vertrauensrichtlinie hinzu. Informationen zum Erstellen einer neuen Rolle finden Sie unter Erstellen einer neuen Rolle im AWS Directory Service Administrationshandbuch.

Zugriff auf Neptune über Lambda

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die AWS Lambda Konsole unter https://console.aws.amazon.com/lambda/.
2. Erstellen Sie eine neue Lambda-Funktion für die Python-Version 3.6.
3. Weisen Sie die `AWSLambdaVPCAccessExecutionRole`-Rolle der Lambda-Funktion zu. Dies ist für den Zugriff auf Neptune-Ressourcen erforderlich, die nur in einer VPC existieren.
4. Weisen Sie der Lambda-Funktion die Neptune-Authentifizierungs-IAM-Rolle zu.

   Weitere Informationen finden Sie unter AWS -Lambda-Berechtigungen im AWS Lambda - Entwicklerhandbuch.
5. Kopieren Sie das IAM-Authentifizierungs-Python-Beispiel in den Lambda-Funktionscode.

   Weitere Informationen zum Beispiel und zum Beispielcode finden Sie unter the section called “Herstellen einer Verbindung mit Python”.

Einrichten von Amazon EC2 für die Neptune-IAM-Authentifizierung

Mit Amazon EC2 können Sie Instance-Profile verwenden, um automatisch Anmeldeinformationen bereitzustellen. Weitere Informationen finden Sie unter Verwenden von Instance-Profilen im IAM-Benutzerhandbuch.

Zunächst müssen Sie eine Vertrauensstellung hinzufügen, die dem Amazon-EC2-Service die Berechtigung zur Übernahme der Rolle gewährt.

Fügen Sie der Neptune-IAM-Authentifizierungsrolle die folgende Vertrauensstellung hinzu. Wenn Sie keine Neptune-IAM-Authentifizierungsrolle haben, siehe the section called “Arten von IAM-Richtlinien”.

JSON

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

```
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Informationen zum Hinzufügen der Vertrauensstellung zur Rolle finden Sie unter Bearbeiten der Vertrauensstellung für eine vorhandene Rolle im AWS Directory Service Administrationshandbuch.

Wenn die Neptune-Richtlinie noch nicht an eine Rolle angefügt ist, erstellen Sie eine neue Rolle. Fügen Sie die Neptune-IAM-Authentifizierungsrichtlinie an und dann die Vertrauensrichtlinie hinzu. Informationen zum Erstellen einer neuen Rolle finden Sie unter Erstellen einer neuen Rolle im AWS Directory Service Administrationshandbuch.

So verwenden Sie ein Skript zum Abrufen der Anmeldeinformationen

1. Führen Sie den folgenden Befehl aus, um den jq-Befehl zu installieren. Das Skript verwendet diesen Befehl zum Analysieren der Ausgabe des curl-Befehl.

   ```
   sudo yum -y install jq
   ```

2. Erstellen Sie in einem Texteditor eine Datei mit der Bezeichnung `credentials.sh`, und fügen Sie den folgenden Text hinzu. Ersetzen Sie die Service-Region durch Ihren eigenen Wert.

   ```
   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. Führen Sie das Skript in der bash-Shell mit dem `source`-Befehl aus:

```
source credentials.sh
```

   Noch besser ist es, die Befehle in diesem Skript zur `.bashrc`-Datei auf Ihrer EC2-Instance hinzuzufügen, so dass sie automatisch aufgerufen werden, wenn Sie sich anmelden, um temporäre Anmeldeinformationen für die Gremlin-Konsole verfügbar zu machen.

4. Stellen Sie die Verbindung mit einem der folgenden Verfahren her.

   - the section called “Von der Gremlin-Konsole aus eine Verbindung herstellen”
   - the section called “Verbindung mit Gremlin Java herstellen”
   - the section called “Verbindung mit SPARQL Java herstellen”
   - the section called “Herstellen einer Verbindung mit Python”

Protokollierung und Überwachung von Nutzung und Leistung in Amazon Neptune

Amazon Neptune unterstützt verschiedene Methoden für die Überwachung von Leistung und Nutzung:

- Cluster-Status – Prüfen des Zustands der Graphdatenbank-Engine eines Neptune-Clusters. Weitere Informationen finden Sie unter the section called “Instance-Status”.
- Amazon CloudWatch — Neptune sendet automatisch Messwerte an Alarme CloudWatch und unterstützt CloudWatch diese auch. Weitere Informationen finden Sie unter the section called “Benutzen CloudWatch”.
- Audit-Protokolldateien – Sie können Datenbankprotokolldateien über die Neptune-Konsole anzeigen, herunterladen oder beobachten. Weitere Informationen finden Sie unter the section called “Prüfprotokolle mit Neptune”.

- Protokolle in Amazon CloudWatch Logs veröffentlichen — Sie können einen Neptune-DB-Cluster so konfigurieren, dass Audit-Protokolldaten in einer Protokollgruppe in Amazon CloudWatch Logs veröffentlicht werden. Mit CloudWatch Logs können Sie die Protokolldaten in Echtzeit analysieren, Alarme erstellen und Metriken anzeigen und CloudWatch Logs verwenden, um Ihre Protokolldatensätze auf einem äußerst dauerhaften Speicher zu speichern. CloudWatch Weitere Informationen finden Sie unter Neptun-Logs CloudWatch .
- AWS CloudTrail— Neptune unterstützt die API-Protokollierung mithilfe von. CloudTrail Weitere Informationen finden Sie unter the section called “Neptune-API-Aufrufe protokollieren mit AWS CloudTrail”.
- Tagging – Sie können Tags verwenden, um Ihren Neptune-Ressourcen Metadaten hinzuzufügen und die Nutzung mit Tags nachzuverfolgen. Weitere Informationen finden Sie unter the section called “Neptun-Ressourcen taggen”.

Amazon Neptune- und Schnittstellen-VPC-Endpunkte (AWS PrivateLink)

Sie können AWS PrivateLinkes verwenden, um eine private Verbindung zwischen Ihrer VPC und den Amazon Neptune API-Endpunkten herzustellen. Sie können auf Amazon Neptune API-Operationen zugreifen, als ob sie sich in Ihrer VPC befinden würden, ohne ein Internet-Gateway, ein NAT-Gerät, eine VPN-Verbindung oder eine Verbindung verwenden zu müssen. Der Verkehr zwischen Ihrer VPC und Amazon Neptune verbleibt im Amazon-Netzwerk.

Amazon Neptune teilt die API-Infrastruktur mit Amazon Relational Database Service. Um VPC-Schnittstellen-Endpunkte für Amazon Neptune einzurichten, folgen Sie den Anweisungen in Amazon RDS-API und Schnittstellen-VPC-Endpunkten (AWS PrivateLink) im Amazon Aurora Aurora-Benutzerhandbuch.

Überlegungen zur Einhaltung von Vorschriften für Amazon Neptune

Informationen darüber, ob ein in den Geltungsbereich bestimmter Compliance-Programme AWS-Service fällt, finden Sie unter AWS-Services Umfang nach Compliance-Programm unter Umfang nach Compliance-ProgrammAWS-Services . Wählen Sie aus, an dem Sie interessiert sind. Allgemeine Informationen finden Sie unter AWS Compliance-Programme AWS .

Sie können Prüfberichte von Drittanbietern unter herunterladen AWS Artifact. Weitere Informationen finden Sie unter Berichte herunterladen unter .

Ihre Verantwortung für die Einhaltung der Vorschriften bei der Nutzung AWS-Services hängt von der Vertraulichkeit Ihrer Daten, den Compliance-Zielen Ihres Unternehmens und den geltenden Gesetzen und Vorschriften ab. Weitere Informationen zu Ihrer Verantwortung für die Einhaltung der Vorschriften bei der Nutzung AWS-Services finden Sie in der AWS Sicherheitsdokumentation.

Aufbau robuster und katastrophentoleranter Amazon Neptune Neptune-Bereitstellungen

Die AWS globale Infrastruktur basiert auf Regionen und Availability Zones. AWS AWS Regionen bieten mehrere physisch getrennte und isolierte Availability Zones, die über Netzwerke mit niedriger Latenz, hohem Durchsatz und hoher Redundanz miteinander verbunden sind. Mithilfe von Availability Zones können Sie Anwendungen und Datenbanken erstellen und ausführen, die automatisch Failover zwischen Availability Zones ausführen, ohne dass es zu Unterbrechungen kommt. Availability Zones sind besser hoch verfügbar, fehlertoleranter und skalierbarer als herkömmliche Infrastrukturen mit einem oder mehreren Rechenzentren.

Ein Amazon–Neptune-DB-Cluster kann nur in einer Amazon VPC erstellt werden, die mindestens zwei Subnetze in mindestens zwei Availability Zones besitzt. Mit der Verteilung Ihrer Cluster-Instances über mindestens zwei Availability Zones trägt Neptune dazu bei, dass im DB-Cluster auch bei einem unwahrscheinlichen Availability-Zone-Ausfall Instances verfügbar sind. Das Cluster-Volume für den Neptune-DB-Cluster umfasst stets drei Availability Zones, um dauerhaften Speicherplatz mit einem geringeren Datenverlustrisiko bereitzustellen.

Weitere Informationen zu AWS Regionen und Availability Zones finden Sie unter AWS Globale Infrastruktur.

Migrieren eines vorhandenen Graphen zu Amazon Neptune

Es gibt eine Reihe von Tools und Techniken, mit denen Sie vorhandene Graphdaten aus einem anderen Datenspeicher zu Amazon Neptune migrieren können.

Ein einfacher Migrations-Workflow umfasst die folgenden Schritte:

- Exportieren der Daten aus dem vorhandenen Speicher zu Amazon Simple Storage Service (Amazon S3).
- Bereinigen und Formatieren der Daten für den Import.
- Laden der Daten in einen Neptune-DB-Cluster mit Neptune-Massen-Loader.
- Konfigurieren Sie Ihre Gremlin- oder SPARQL-Anwendung so, dass sie den entsprechenden Endpunkt verwendet, den Neptune bereitstellt.

> **Note**
>
> Ihr Neptune-Cluster muss in einer VPC ausgeführt werden, auf die Ihre Anwendung zugreifen kann.

Je nachdem, wo die Daten gespeichert sind, gibt es Möglichkeiten, einige dieser Schritte zu vereinfachen und zu automatisieren:

Themen

- Migration von Neo4j zu Amazon Neptune
- Migrieren eines vorhandenen Graphen von einem Apache TinkerPop Gremlin-Server zu Amazon Neptune
- Migrieren eines vorhandenen Graphen von einem RDF-Triple-Store zu Amazon Neptune
- Verwenden von AWS Database Migration Service (AWS DMS) zur Migration von einer relationalen oder NoSQL-Datenbank zu Amazon Neptune
- Migration von Blazegraph zu Amazon Neptune

Migration von Neo4j zu Amazon Neptune

Neo4j und Amazon Neptune sind beide Graphdatenbanken, die für transaktionale Online-Graph-Workloads konzipiert wurden und das Labeled-Property-Graph-Datenmodell unterstützen. Aufgrund dieser Ähnlichkeiten ist Neptune eine beliebte Wahl für Kunden, die ihre aktuellen Neo4j-Anwendungen migrieren möchten. Bei diesen Migrationen handelt es sich jedoch nicht einfach um Lift-and-Shift-Lösungen, da es zwischen den beiden Datenbanken Unterschiede bei den Sprachen und der Feature-Unterstützung, den Betriebsmerkmalen, der Serverarchitektur und den Speicherkapazitäten gibt.

Auf dieser Seite wird der Migrationsprozess beschrieben und es werden Dinge aufgezeigt, die vor der Migration einer Neo4j-Graph-Anwendung zu Neptune zu beachten sind. Diese Überlegungen gelten generell für jede Neo4j-Graph-Anwendung, unabhängig davon, ob sie von einer Community-, Enterprise- oder Aura-Datenbank betrieben wird. Obwohl jede Lösung einzigartig ist und zusätzliche Verfahren erfordern kann, folgen alle Migrationen demselben allgemeinen Muster.

Jeder der in den folgenden Abschnitten beschriebenen Schritte beinhaltet Überlegungen und Empfehlungen zur Vereinfachung des Migrationsprozesses. Darüber hinaus gibt es Open-Source-Tools und Blog-Beiträge, in denen der Prozess beschrieben wird, sowie einen Abschnitt zur Feature-Kompatibilität mit empfohlenen Architekturoptionen.

Themen

- Allgemeine Informationen zur Migration von Neo4j zu Neptune
- Vorbereitung der Migration von Neo4j zu Neptune
- Bereitstellung der Infrastruktur bei der Migration von Neo4j zu Neptune
- Datenmigration von Neo4j zu Neptune
- Anwendungsmigration von Neo4j zu Neptune
- Neptune-Kompatibilität mit Neo4j
- Umschreiben von Cypher-Abfragen zur Ausführung in openCypher auf Neptune
- Ressourcen für die Migration von Neo4j zu Neptune

Allgemeine Informationen zur Migration von Neo4j zu Neptune

Mit der Unterstützung für die openCypher-Abfragesprache durch Neptune können Sie die meisten Neo4j-Workloads, die das Bolt-Protokoll oder HTTPS verwenden, zu Neptune verschieben.

OpenCypher ist jedoch eine Open-Source-Spezifikation, die die meisten, aber nicht alle Funktionen enthält, die von anderen Datenbanken wie Neo4j unterstützt werden.

Obwohl Neptune in vielerlei Hinsicht kompatibel ist, stellt es keinen direkten Ersatz für Neo4j dar. Neptune ist ein vollständig verwalteter Graphdatenbank-Service mit Enterprise-Features wie hohe Verfügbarkeit und hohe Datenbeständigkeit, der sich architektonisch von Neo4j unterscheidet. Neptune ist Instance-basiert und verfügt über eine einzige primäre Writer-Instance und bis zu 15 Lesereplikat-Instances, mit denen Sie die Lesekapazität horizontal skalieren können. Mit Neptune Serverless können Sie Ihre Datenverarbeitungskapazität je nach Abfragevolumen automatisch nach oben oder unten skalieren. Dies ist unabhängig vom Neptune-Speicher, der automatisch skaliert wird, wenn Sie Daten hinzufügen.

Neptune unterstützt die Open-Source-Standardspezifikation OpenCypher, Version 9. Wir bei glauben AWS, dass Open Source für alle gut ist, und wir setzen uns dafür ein, sowohl unseren Kunden den Wert von Open Source näherzubringen als auch Open-Source-Communities die operative Exzellenz AWS zu bieten.

Viele Anwendungen, die auf Neo4j laufen, verwenden jedoch auch proprietäre Features, die nicht als Open-Source-Funktionen verfügbar sind und die Neptune nicht unterstützt. Neptune unterstützt beispielsweise keine APOC-Prozeduren, einige Cypher-spezifische Klauseln und Funktionen sowie die Datentypen `Char`, `Date` oder `Duration`. Neptune wandelt die fehlenden Datentypen automatisch in unterstützte Datentypen um.

Neben OpenCypher unterstützt Neptune auch die Apache TinkerPop Gremlin-Abfragesprache für Eigenschaftsdiagramme (sowie SPARQL für RDF-Daten). Gremlin kann mit openCypher auf demselben Eigenschaftsgraphen interoperieren und in vielen Fällen können Sie Gremlin verwenden, um Funktionen bereitzustellen, die openCypher nicht bietet. Im Folgenden finden Sie einen kurzen Vergleich der beiden Sprachen:

	openCypher	Gremlin
Style	Deklarativ	Unerlässlich

Syntax	Mustervergleich
	`Match p=(a)-[:route]->(d)`
	`WHERE a.code='ANC'`
	`RETURN p`
		Transversal-basiert
		`g.V().has('code', 'ANC').`
		`out('route').path().`
		`by(elementMap())`

	openCypher	Gremlin
Benutzerfreundlichkeit	Von SQL-inspiriert, auch für Nicht-Programmierer lesbar	Steilere Lernkurve, ähnlich wie bei Programmiersprachen wie Java
Flexibilität	Niedrig	Hoch
Unterstützung für Abfragen	Zeichenfolgenbasierte Abfragen	Zeichenfolgenbasierte Abfragen oder Inline-Code, der von Client-Bibliotheken unterstützt wird
Clients	HTTPS und Bolt	HTTPS und Websockets

Im Allgemeinen ist es nicht erforderlich, Ihr Datenmodell zu ändern, um von Neo4j zu Neptune zu migrieren, da sowohl Neo4j als auch Neptune Labeled Property Graph (LPG)-Daten unterstützen. Neptune weist jedoch einige Architektur- und Datenmodellunterschiede auf, die Sie zur Leistungsoptimierung nutzen können. Beispiel:

- Neptune IDs werden als erstklassige Bürger behandelt.
- Neptune verwendet AWS Identity and Access Management (IAM)-Richtlinien, um den Zugriff auf Ihre Graphdaten auf flexible und detaillierte Weise zu sichern.
- Neptune bietet mehrere Möglichkeiten zur Verwendung von Jupyter-Notebooks, um Abfragen auszuführen und die Ergebnisse zu visualisieren. Neptune funktioniert auch mit Visualisierungs-Tools von Drittanbietern.
- >Neptune hat zwar keinen direkten Ersatz für die Neo4j Graph Data Science (GDS)-Bibliothek, unterstützt aber Graphanalysen mit einer Vielzahl von Lösungen. In mehreren Beispiel-Notebooks wird beispielsweise gezeigt, wie die Neptune-Integration mit dem AWS Pandas SDK in Python-Umgebungen genutzt werden kann, um Analysen von Grafikdaten durchzuführen.

Bitte wenden Sie sich an den AWS Support oder wenden Sie sich an Ihr AWS Account-Team, wenn Sie Fragen haben. Wir nutzen Ihr Feedback, um neue Features zu priorisieren, die Ihren Anforderungen entsprechen.

Vorbereitung der Migration von Neo4j zu Neptune

Die Migration von der Neo4j-Graphdatenbank zum Neptune-Graphdatenbankdienst kann auf zwei Arten angegangen werden: Re-Platforming oder Refactoring/Rearchitecting. Der Re-Platforming-Ansatz beinhaltet die Änderung des bestehenden Datenmodells und der Anwendungsarchitektur, um die Fähigkeiten von Neptune bestmöglich zu nutzen, während sich der Refactoring-Ansatz darauf konzentriert, äquivalente Komponenten in Neptune zu finden, um eine vergleichbare Implementierung zu erstellen. In der Praxis wird häufig eine Kombination dieser Strategien verwendet, da der Migrationsprozess ein Gleichgewicht zwischen der Neptune-Zielarchitektur und den Einschränkungen und Anforderungen der bestehenden Neo4j-Implementierung beinhaltet. Unabhängig vom Ansatz besteht der Schlüssel darin, ausgehend von den Anwendungsfällen der Anwendung das Datenmodell, die Abfragen und die Gesamtarchitektur zu entwerfen, die Ihren Anforderungen am besten entsprechen.

Konzepte für die Migration

Bei der Migration einer Neo4j-Anwendung zu Neptune empfehlen wir eine von zwei Strategien: entweder Re-Platforming oder Refactoring/Rearchitecting. Weitere Informationen zu Migrationsstrategien finden Sie unter 6 Strategien für die Migration von Anwendungen in die Cloud, Blog-Beitrag von Stephen Orban.

Der Re-Platforming-Ansatz, der manchmal auch genannt wird lift-tinker-and-shift, umfasst die folgenden Schritte:

- Identifizieren Sie die Anwendungsfälle, für die Ihre Anwendung gedacht ist.
- Modifizieren Sie das bestehende Graphdatenmodell und die Anwendungsarchitektur, um diese Workload-Anforderungen mithilfe der Funktionen von Neptune bestmöglich zu erfüllen.
- Ermitteln Sie, wie Daten, Abfragen und andere Teile der Quellanwendung zum Zielmodell und zur Zielarchitektur migriert werden sollen.

Dieser vom Ende zum Anfang vorgehende Ansatz ermöglicht Ihnen, Ihre Anwendung auf die Art von Neptune-Lösung zu migrieren, die Sie entwerfen würden, wenn es sich um ein völlig neues Projekt handeln würde.

Dagegen beinhaltet das Refactoring-Konzept:

- Identifizieren der Komponenten der bestehenden Implementierung, einschließlich Infrastruktur, Daten, Abfragen und Anwendungsfunktionen.

- Suchen nach Äquivalenten in Neptune, die verwendet werden können, um eine vergleichbare Implementierung zu erstellen.

Dieser vom Anfang zum Ende vorgehende Ansatz zielt darauf ab, eine Implementierung gegen eine andere auszutauschen.

In der Praxis werden Sie wahrscheinlich eine Mischung aus diesen beiden Konzepten wählen. Sie könnten mit einem Anwendungsfall beginnen, die Neptune-Zielarchitektur entwerfen, sich dann aber der bestehenden Neo4j-Implementierung zuwenden, um Einschränkungen und Invarianten zu identifizieren, die Sie beibehalten müssen. Möglicherweise müssen Sie die Integration mit anderen externen Systemen fortsetzen oder weiterhin spezielle APIs Angebote für Nutzer Ihrer Graph-Anwendung anbieten. Anhand dieser Informationen können Sie ermitteln, welche Daten bereits vorhanden sind, um sie in Ihr Zielmodell zu übertragen, und welche Daten aus anderen Quellen kommen müssen.

An anderen Stellen könnten Sie damit beginnen, einen bestimmten Teil Ihrer Neo4j-Implementierung als Informationsquelle zu der Aufgabe zu analysieren, für die Ihre Anwendung vorgesehen ist. Diese Art von „Archäologie“ in der bestehenden Anwendung kann dabei helfen, einen Anwendungsfall zu beschreiben, den Sie dann so ausrichten können, dass die Fähigkeiten von Neptune genutzt werden.

Unabhängig davon, ob Sie eine neue Anwendung mit Neptune erstellen oder eine bestehende Anwendung von Neo4j migrieren, empfehlen wir, ausgehend von Anwendungsfällen ein Datenmodell, eine Reihe von Abfragen und eine Anwendungsarchitektur zu entwerfen, die Ihren geschäftlichen Anforderungen entsprechen.

Architektonische Unterschiede zwischen Neptune und Neo4j

Wenn Kunden zum ersten Mal erwägen, eine Anwendung von Neo4j zu Neptune zu migrieren, ist es oft verlockend, einen like-to-like Vergleich anhand der Instanzgröße durchzuführen. Die Architekturen von Neo4j und Neptune weisen jedoch grundlegende Unterschiede auf. Neo4j basiert auf einem all-in-one Ansatz, bei dem das Laden von Daten, Daten-ETL, Anwendungsabfragen, Datenspeicherung und Verwaltungsvorgänge alle in demselben Satz von Rechenressourcen, wie z. B. EC2-Instances, ablaufen.

Neptune ist hingegen eine OLTP-fokussierte Graphdatenbank, in der die Architektur die Verantwortlichkeiten voneinander trennt und in der Ressourcen entkoppelt sind, so dass sie dynamisch und unabhängig skaliert werden können.

Ermitteln Sie bei der Migration von Neo4j zu Neptune die Anforderungen an Datenbeständigkeit, Verfügbarkeit und Skalierbarkeit Ihrer Anwendung. Die Cluster-Architektur von Neptune vereinfacht das Design von Anwendungen, die hohe Datenbeständigkeit, Verfügbarkeit und Skalierbarkeit erfordern. Mit einem Verständnis der Cluster-Architektur von Neptune können Sie dann eine Neptune-Cluster-Topologie entwerfen, die diese Anforderungen erfüllt.

Die Cluster-Architektur von Neo4j

Viele Produktionsanwendungen nutzen das kausale Clustering von Neo4j, um Datenbeständigkeit, hohe Verfügbarkeit und Skalierbarkeit zu gewährleisten. Die Clustering-Architektur von Neo4j verwendet Core-Server- und Lesereplikat-Instances:

- Core-Server sorgen für Datenbeständigkeit und Fehlertoleranz, indem sie Daten mithilfe des Raft-Protokolls replizieren.
- Lesereplikate verwenden den Versand von Transaktionsprotokollen, um Daten für Workloads mit hohem Lesedurchsatz asynchron zu replizieren.

Jede Instance in einem Cluster, ob Core-Server oder Lesereplikat, enthält eine vollständige Kopie der Graphdaten.

Die Cluster-Architektur von Neptune

Ein Neptune-Cluster besteht aus einer primären Writer-Instance und bis zu 15 Lesereplikat-Instances. Alle Instances im Cluster nutzen denselben zugrunde liegenden verteilten Speicherservice, der von den Instances getrennt ist.

- Die primäre Writer-Instance koordiniert alle Schreibvorgänge in der Datenbank und ist vertikal skalierbar, um flexible Unterstützung für unterschiedliche Schreib-Workloads zu bieten. Sie unterstützt auch Lesevorgänge.
- Lesereplikat-Instances unterstützen Lesevorgänge vom zugrunde liegenden Speicher-Volume aus und ermöglichen eine horizontale Skalierung, um hohe Lese-Workloads zu unterstützen. Sie sorgen auch für hohe Verfügbarkeit, indem sie als Failover-Ziele für die primäre Instance dienen.

> **Note**
>
> Bei hohen Schreib-Workloads empfiehlt es sich, die Lesereplikat-Instances auf die gleiche Größe wie die Writer-Instance zu skalieren, um sicherzustellen, dass die Reader bei den Datenänderungen konsistent bleiben können.

- Das zugrundeliegende Volume skaliert die Speicherkapazität automatisch, wenn die Datenmenge in Ihrer Datenbank zunimmt, und zwar bis zu 128 Tebibyte (TiB) Speicher.

Die Instance-Größen sind dynamisch und unabhängig. Die Größe jeder Instance kann geändert werden, während der Cluster ausgeführt wird, und es können auch Lesereplikate hinzugefügt oder entfernt werden, während der Cluster läuft.

Das Feature Neptune Serverless kann Ihre Rechenkapazität bei steigendem und fallendem Bedarf automatisch hoch- und herunterskalieren. Dadurch können Sie nicht nur Ihren Verwaltungsaufwand verringern, sondern die Datenbank auch so konfigurieren, dass sie große Bedarfsspitzen bewältigen kann, ohne dass die Leistung beeinträchtigt wird oder Sie zu viel bereitstellen müssen.

Sie können einen Neptune-Cluster für bis zu sieben Tage anhalten.

Neptune unterstützt auch Auto Scaling, um die Instance-Größen des Readers automatisch an den Workload anzupassen.

Mithilfe des globalen Datenbank-Features von Neptune können Sie einen Cluster in bis zu 5 anderen Regionen spiegeln.

Neptune ist außerdem vom Design her fehlertolerant:

- Das Cluster-Volume, das Datenspeicher für alle Instances im Cluster bereitstellt, erstreckt sich über mehrere Availability Zones (AZs) in einer einzigen. AWS-Region Jede AZ beinhaltet eine vollständige Kopie der Cluster-Daten.
- Wenn die primäre Instance nicht verfügbar ist, wechselt Neptune automatisch zu einem vorhandenen Lesereplikat ohne Datenverlust, normalerweise in weniger als 30 Sekunden. Wenn im Cluster keine Lesereplikate vorhanden sind, stellt Neptune automatisch eine neue primäre Instance bereit – auch dies ohne Datenverlust.

All dies bedeutet, dass Sie bei der Migration von einem Neo4j-Kausal-Cluster zu Neptune die Cluster-Topologie nicht explizit auf hohe Datenbeständigkeit und hohe Verfügbarkeit auslegen müssen. Auf diese Weise können Sie die Größe Ihres Clusters für die erwarteten Lese- und Schreib-Workloads und etwaige erhöhte Verfügbarkeitsanforderungen anpassen, und zwar auf verschiedene Arten:

- Um Lesevorgänge zu skalieren, fügen Sie Lesereplikat-Instances hinzu oder aktivieren Sie die Neptune-Serverless-Funktionalität.

- Um die Verfügbarkeit zu verbessern, verteilen Sie die primäre Instance und die Read Replicas in Ihrem Cluster auf mehrere Availability Zones ()AZs.
- Um die Failover-Zeit zu reduzieren, sollten Sie mindestens eine Lesereplikat-Instance bereitstellen, die als Failover-Ziel für die primäre Instance dienen kann. Sie können die Reihenfolge, in der Lesereplikat-Instances nach einem Ausfall zur primären Instance hochgestuft werden, anpassen, indem Sie jeder Replik eine Priorität zuweisen. Es hat sich bewährt, sicherzustellen, dass ein Failover-Ziel über eine Instance-Klasse verfügt, die in der Lage ist, die Schreib-Workloads Ihrer Anwendung zu bewältigen, wenn es zur primären Instance heraufgestuft wird.

Unterschiede bei der Datenspeicherung zwischen Neptune und Neo4j

Neptune verwendet ein Graphdatenmodell, das auf einem nativen Quad-Modell basiert. Bei der Migration Ihrer Daten zu Neptune gibt es mehrere Unterschiede bei der Architektur des Datenmodells und der Speicherebene, die Sie beachten sollten, um den von Neptune bereitgestellten verteilten und skalierbaren gemeinsamen Speicher optimal nutzen zu können:

- Neptune verwendet kein explizit definiertes Schema oder Einschränkungen. Damit können Sie Knoten, Edges und Eigenschaften dynamisch hinzufügen, ohne das Schema im Voraus definieren zu müssen. Neptune schränkt die Werte und Typen der gespeicherten Daten nicht ein, es sei denn, dies ist in den Neptune-Grenzwerten angegeben. Als Teil der Speicherarchitektur von Neptune werden Daten außerdem automatisch indiziert, so dass viele der gängigsten Zugriffsmuster verarbeitet werden. Durch diese Speicherarchitektur entfällt der betriebliche Aufwand für die Erstellung und Verwaltung des Datenbankschemas und der Indexoptimierung.
- Neptune bietet eine einzigartige verteilte und gemeinsam genutzte Speicherarchitektur, die automatisch in 10-GB-Blöcken skaliert wird, wenn der Speicherbedarf Ihrer Datenbank wächst, und zwar auf bis zu 128 Tebibyte (TiB). Diese Speicherebene ist zuverlässig, robust und fehlertolerant. Daten werden sechsmal kopiert, und zwar zweimal in jeder der drei Availability Zones. Dies bietet allen Neptune-Clustern standardmäßig eine hochverfügbare und fehlertolerante Datenspeicherebene. Die Speicherarchitektur von Neptune senkt die Kosten und macht die Bereitstellung oder Überversorgung von Speicher überflüssig, um künftiges Datenwachstum zu bewältigen.

Bevor Sie Ihre Daten zu Neptune migrieren, sollten Sie sich mit dem Eigenschaftsgraph-Datenmodell und der Transaktionssemantik von Neptune vertraut machen.

Operationale Unterschiede zwischen Neptune und Neo4j

Neptune ist ein vollständig verwalteter Service, der viele der normalen Betriebsaufgaben automatisiert, die Sie erledigen müssten, wenn Sie On-Premises- oder selbstverwaltete Datenbanken wie Neo4j Enterprise oder Community Edition verwenden würden:

- Automatisierte Backups – Neptune sichert Ihr Cluster-Volume automatisch und behält das Backup für einen von Ihnen festgelegten Aufbewahrungszeitraum bei (von 1 bis 35 Tagen). Diese Backups sind kontinuierlich und inkrementell, so dass Sie schnell eine Wiederherstellung zu einem beliebigen Punkt im Aufbewahrungszeitraum für Backups durchführen können. Es gibt keine Auswirkungen auf die Leistungsfähigkeit oder Unterbrechung von Datenbank-Services, während Backup-Daten geschrieben werden.
- Manuelle Snapshots – Mit Neptune können Sie einen Snapshot für das Volume mit Ihrem DB-Cluster erstellen, damit der gesamte DB-Cluster gesichert wird. Diese Art von Snapshot kann dann verwendet werden, um die Datenbank wiederherzustellen, eine Kopie davon zu erstellen und sie für mehrere Konten freizugeben.
- Klonen – Neptune unterstützt ein Klon-Feature, mit dem Sie schnell kostengünstige Klone einer Datenbank erstellen können. Die Klone verwenden ein copy-on-write Protokoll, sodass sie nach ihrer Erstellung nur minimalen zusätzlichen Speicherplatz benötigen. Das Klonen von Datenbanken ist eine effektive Methode, um neue Neptune-Features oder -Upgrades auszuprobieren, ohne den ursprünglichen Cluster zu beeinflussen.
- Überwachung – Neptune bietet verschiedene Methoden zur Überwachung der Leistung und Nutzung Ihres Clusters, darunter:
  - Instance-Status
  - Integration mit Amazon CloudWatch und AWS CloudTrail
  - Prüfprotokollfunktionen
  - Ereignismeldungen
  - Tagging
- Sicherheit – Neptune bietet standardmäßig eine sichere Umgebung. Ein Cluster befindet sich in einer privaten VPC, die die Netzwerkisolation von anderen Ressourcen ermöglicht. Der gesamte Datenverkehr wird über SSL verschlüsselt, und alle Daten werden im Ruhezustand mit SSL verschlüsselt AWS KMS.

  Darüber hinaus lässt sich Neptune in AWS Identity and Access Management (IAM) integrieren, um eine Authentifizierung zu ermöglichen. Durch die Angabe von IAM-Bedingungsschlüsseln

können Sie IAM-Richtlinien verwenden, um eine differenzierte Zugriffskontrolle für Datenaktionen zu gewährleisten.

Unterschiede bei Tools und Integration zwischen Neptune und Neo4j

Neptune hat eine andere Architektur für Integrationen und Tools als Neo4j, was sich auf die Architektur Ihrer Anwendung auswirken kann. Neptune verwendet die Rechenressourcen des Clusters, um Abfragen zu verarbeiten, nutzt aber auch andere best-in-class AWS Dienste für Funktionen wie Volltextsuche (mit OpenSearch), ETL (mit Glue) usw. Eine vollständige Liste dieser Integrationen finden Sie unter Neptune-Integrationen.

Bereitstellung der Infrastruktur bei der Migration von Neo4j zu Neptune

Amazon-Neptune-Cluster sind so konzipiert, dass sie in drei Dimensionen skalierbar sind: Speicher, Schreibkapazität und Lesekapazität. In den folgenden Abschnitten werden spezifische Optionen erörtert, die bei der Migration berücksichtigt werden sollten.

Bereitstellen von Speicher

Der Speicher für jeden Neptune-Cluster wird automatisch bereitgestellt, ohne dass Sie zusätzlichen Verwaltungsaufwand haben. Die Größe wird dynamisch in 10-GB-Blöcken angepasst, wenn der Speicherbedarf des Clusters steigt. Daher ist es nicht erforderlich, Speicherplatz zu schätzen und bereitzustellen oder zu viel bereitzustellen, um das künftige Datenwachstum zu bewältigen.

Bereitstellen von Schreibkapazität

Neptune bietet eine Single-Writer-Instance, die vertikal auf jede Instance-Größe skaliert werden kann, die auf der Neptune-Preisseite verfügbar ist. Beim Lesen und Schreiben von Daten in eine Writer-Instance sind alle Transaktionen ACID-konform, wobei die Datenisolation wie unter Transaktionsisolierungsstufen in Neptune definiert erfolgt.

Um die optimale Größe für eine Writer-Instance auszuwählen, müssen Lasttests durchgeführt werden, um die optimale Instance-Größe für Ihren Workload zu ermitteln. Die Größe jeder Instance in Neptune kann jederzeit geändert werden, indem die DB-Instance-Klasse geändert wird. Sie können die Größe einer Start-Instance auf der Grundlage der Parallelität und der durchschnittlichen Abfragelatenz schätzen, wie weiter unten unter Schätzen der optimalen Instance-Größe bei der Bereitstellung Ihres Clusters beschrieben.

Bereitstellen von Lesekapazität

Neptune wurde entwickelt, um Lesereplikat-Instances sowohl horizontal zu skalieren, indem bis zu 15 von ihnen innerhalb eines Clusters (oder mehr in einer globalen Neptune-Datenbank) hinzugefügt werden, als auch vertikal zu jeder beliebigen Instance-Größe, die auf der Neptune-Preisseite verfügbar ist. Alle Neptune-Lesereplikat-Instances verwenden dasselbe zugrunde liegende Speicher-Volume, was eine transparente Replikation von Daten mit minimaler Verzögerung ermöglicht.

Lesereplikate ermöglichen nicht nur die horizontale Skalierung von Leseanforderungen innerhalb eines Neptune-Clusters, sondern dienen auch als Failover-Ziele für die Writer-Instance, um eine hohe Verfügbarkeit zu gewährleisten. Vorschläge, wie Sie die richtige Anzahl und Platzierung von Lesereplikaten in Ihrem Cluster ermitteln können, finden Sie unter Grundlegende Anleitungen für den Amazon-Neptune-Betrieb.

Für Anwendungen, bei denen Konnektivität und Workload nicht vorhersehbar sind, unterstützt Neptune auch ein Auto-Scaling-Feature, mit dem die Anzahl der Neptune-Replikate anhand der von Ihnen angegebenen Kriterien automatisch angepasst werden kann.

Um eine optimale Größe und Anzahl von Lesereplikat-Instances zu ermitteln, müssen Lasttests durchgeführt werden, um die Eigenschaften des Lese-Workloads zu ermitteln, den sie unterstützen müssen. Die Größe jeder Instance in Neptune kann jederzeit geändert werden, indem die DB-Instance-Klasse geändert wird. Sie können die Größe einer Start-Instance auf der Grundlage der Parallelität und der durchschnittlichen Abfragelatenz schätzen, wie im nächsten Abschnitt beschrieben.

Verwenden Sie Neptune Serverless, um Reader- und Writer-Instances automatisch nach Bedarf zu skalieren

Es ist zwar oft hilfreich, die Rechenkapazität abschätzen zu können, die Ihre erwarteten Workloads benötigen, aber Sie können das Feature Neptune Serverless so konfigurieren, dass die Lese- und Schreibkapazität automatisch nach oben und unten skaliert wird. Auf diese Weise können Sie Spitzenanforderungen erfüllen und gleichzeitig bei sinkender Nachfrage automatisch zurückskalieren.

Schätzen der optimalen Instance-Größe bei der Bereitstellung Ihres Clusters

Um die optimale Instance-Größe zu schätzen, müssen Sie die durchschnittliche Abfragelatenz in Neptune kennen, wenn Ihr Workload ausgeführt wird, sowie die Anzahl der gleichzeitig verarbeiteten Abfragen. Eine grobe Schätzung der Instance-Größe kann berechnet werden, indem die durchschnittliche Abfragelatenz mit der Anzahl gleichzeitiger Abfragen multipliziert wird. Auf diese Weise erhalten Sie die durchschnittliche Anzahl gleichzeitiger Threads, die zur Bewältigung des Workloads benötigt werden.

Jede vCPU in einer Neptune-Instance kann zwei gleichzeitige Abfrage-Threads unterstützen. Wenn Sie also die Threads durch 2 teilen, erhalten Sie die Anzahl der CPUs erforderlichen v, die dann auf der Neptune-Preisseite mit der entsprechenden Instance-Größe korreliert werden kann. Beispiel:

```
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
```

Wenn wir dies mit der Anzahl von v CPUs in einer Instanz korrelieren, erhalten wir eine grobe Schätzung, dass a die empfohlene Instanz für diese `r5.4xlarge` Arbeitslast ist. Diese Schätzung ist

grob und dient nur als erste Orientierung bei der Auswahl der Instance-Größe. Bei jeder Anwendung sollte die korrekte Dimensionierung durchgeführt werden, um die richtige Anzahl und Art(en) von Instances zu ermitteln, die für den Workload geeignet sind.

Die Speicheranforderungen sollten ebenso wie die Verarbeitungsanforderungen berücksichtigt werden. Neptune ist am leistungsfähigsten, wenn die Daten, auf die durch Abfragen zugegriffen wird, im Pufferpool-Cache des Hauptspeichers verfügbar sind. Durch die Bereitstellung von ausreichend Arbeitsspeicher können auch die I/O Kosten erheblich gesenkt werden.

Weitere Informationen und Anleitungen zur Dimensionierung von Instances in einem Neptune-Cluster finden Sie auf der Dimensionieren von DB-Instances in einem Neptune-DB-Cluster-Seite.

Datenmigration von Neo4j zu Neptune

Bei der Migration von Neo4j zu Amazon Neptune ist die Migration der Daten ein wichtiger Schritt des Prozesses. Es gibt mehrere Vorgehensweisen für die Migration von Daten. Das richtige Konzept hängt von den Anforderungen der Anwendung, der Datengröße und der Art der gewünschten Migration ab. Bei vielen dieser Migrationen müssen jedoch dieselben Überlegungen berücksichtigt werden, von denen einige im Folgenden hervorgehoben werden.

> **Note**
>
> Einen vollständigen Überblick step-by-step über ein Beispiel für die Durchführung einer Offline-Datenmigration finden Sie im Datenbank-Blog unter Migration einer AWS Neo4j-Graphdatenbank zu Neptune mit einem vollautomatischen Hilfsprogramm.

Bewertung der Datenmigration von Neo4j zu Neptune

Der erste Schritt bei der Bewertung einer Datenmigration besteht darin, festzulegen, wie die Daten migriert werden sollen. Die Optionen hängen von der Architektur der zu migrierenden Anwendung, der Datengröße und den Verfügbarkeitsanforderungen während der Migration ab. Im Allgemeinen lassen sich Migrationen in eine von zwei Kategorien einteilen: online oder offline.

Offline-Migrationen sind in der Regel am einfachsten durchzuführen, da die Anwendung während der Migration keinen Lese- oder Schreibdatenverkehr akzeptiert. Wenn die Anwendung keinen Datenverkehr mehr akzeptiert, können die Daten exportiert, optimiert, importiert und die Anwendung getestet werden, bevor die Anwendung wieder aktiviert wird.

Online-Migrationen sind komplexer, da die Anwendung während der Datenmigration weiterhin Lese- und Schreibdatenverkehr akzeptieren muss. Die genauen Anforderungen der einzelnen Online-Migrationen können unterschiedlich sein, aber die allgemeine Architektur würde im Allgemeinen der folgenden ähneln:

- Ein Feed mit laufenden Änderungen an der Datenbank muss in Neo4j aktiviert werden, indem Neo4j Streams als Quelle für einen Kafka-Cluster konfiguriert wird.
- Sobald dies abgeschlossen ist, kann ein Export des laufenden Systems durchgeführt werden. Folgen Sie dabei den Anweisungen unter Exportieren von Daten aus Neo4j bei der Migration zu Neptune und der angegebenen Zeit, um später eine Korrelation zum Kafka-Thema vorzunehmen.

- Die exportierten Daten werden dann gemäß den Anweisungen unter Importieren von Daten aus Neo4j bei der Migration zu Neptune in Neptune importiert.
- Geänderte Daten aus dem Kafka-Stream können dann mithilfe einer Architektur, die der unter Schreiben zu Amazon Neptune von Amazon Kinesis Data Streams beschriebenen Architektur ähnelt, in den Neptune-Cluster kopiert werden. Beachten Sie, dass die Replikation der Änderungen parallel ausgeführt werden kann, um die neue Anwendungsarchitektur und Leistung zu validieren.
- Nachdem die Datenmigration validiert wurde, kann der Anwendungsdatenverkehr zum Neptune-Cluster umgeleitet und die Neo4j-Instance außer Betrieb genommen werden.

Datenmodelloptimierungen für die Migration von Neo4j zu Neptune

Sowohl Neptune als auch Neo4j unterstützen beschriftete Eigenschaftsgraphen (LPG). Neptune weist jedoch einige Architektur- und Datenmodellunterschiede auf, die Sie zur Leistungsoptimierung nutzen können:

Optimierung von Knoten und Edge IDs

Neo4j generiert automatisch eine lange Zahl. IDs Mit Cypher können Sie auf Knoten anhand ihrer ID verweisen, aber davon wird generell abgeraten; suchen Sie stattdessen anhand einer indizierten Eigenschaft nach Knoten.

Neptune ermöglicht es Ihnen, Ihre eigene Zeichenkettenbasis IDs für Scheitelpunkte und Kanten anzugeben. Wenn Sie keine eigenen angeben IDs, generiert Neptune automatisch Zeichenkettendarstellungen UUIDs für neue Kanten und Scheitelpunkte.

Wenn Sie Daten von Neo4j zu Neptune migrieren, indem Sie sie aus Neo4j exportieren und dann massenweise in Neptune importieren, können Sie die von Neo4j beibehalten. IDs Die von Neo4j generierten numerischen Werte können IDs beim Import in Neptune als vom Benutzer bereitgestellte Werte dienen, wo sie als Zeichenketten und nicht als numerische Werte dargestellt werden.

Es gibt jedoch Umstände, unter denen Sie eine Scheitelpunkteigenschaft möglicherweise zu einer Scheitelpunkt-ID heraufstufen möchten. So wie die Suche nach einem Knoten mithilfe einer indizierten Eigenschaft der schnellste Weg ist, einen Knoten in Neo4j zu finden, ist das Suchen eines Scheitelpunkts anhand der ID der schnellste Weg, einen Scheitelpunkt in Neptune zu finden. Wenn Sie also eine geeignete Scheitelpunkteigenschaft identifizieren können, die eindeutige Werte enthält, sollten Sie erwägen, die ~id des Scheitelpunkts durch den nominierten Eigenschaftswert in Ihren CSV-Dateien zum Massenladen zu ersetzen. Wenn Sie dies tun, müssen Sie auch alle entsprechenden ~from- und ~to-Edge-Werte in Ihren CSV-Dateien umschreiben.

Schemaeinschränkungen bei der Migration von Daten von Neo4j zu Neptune

In Neptune ist die einzige verfügbare Schemaeinschränkung die Eindeutigkeit der ID eines Knotens oder eines Edges. Für Anwendungen, die eine Eindeutigkeitseinschränkung nutzen müssen, sollte diese Vorgehensweise in Betracht gezogen werden, um eine solche durch Angabe der Knoten- oder Edge-ID zu erreichen. Wenn die Anwendung mehrere Spalten als Eindeutigkeitseinschränkung verwendet hat, kann die ID auf eine Kombination dieser Werte gesetzt werden. `id=123, code='SEA'` könnte beispielsweise als `ID='123_SEA'` dargestellt werden, um eine komplexe Eindeutigkeitsbeschränkung zu erreichen.

Optimierung der Edge-Richtung bei der Migration von Daten von Neo4j zu Neptune

Wenn Knoten, Edges oder Eigenschaften zu Neptune hinzugefügt werden, werden sie automatisch auf drei verschiedene Arten indiziert, mit einem optionalen vierten Index. Aufgrund der Art und Weise, in der Neptune die Indizes erstellt und verwendet, sind Abfragen, die ausgehenden Edges folgen, effizienter als Abfragen, die eingehende Kanten verwenden. Im Hinblick auf das Graphdatenspeichermodell von Neptune handelt es sich dabei um themenbasierte Suchen, die den SPOG-Index verwenden.

Wenn Sie bei der Migration Ihres Datenmodells und Ihrer Abfragen zu Neptune feststellen, dass Ihre wichtigsten Abfragen darauf beruhen, eingehende Edges zu durchqueren, bei denen ein hohes Maß an Fächerung besteht, sollten Sie in Betracht ziehen, Ihr Modell so zu ändern, dass diese Durchquerungen stattdessen ausgehenden Edges folgen, insbesondere wenn Sie nicht angeben können, welche Edge-Etiketten durchquert werden sollen. Kehren Sie dazu die Richtung der relevanten Edges um und aktualisieren Sie die Edge-Etiketten, um die Semantik dieser Richtungsänderung widerzuspiegeln. Sie können beispielsweise Folgendes ändern:

```
person_A – parent_of – person_B
   to:
person_B – child_of – person_A
```

Um diese Änderung in einer CSV-Datei mit Massenladefunktion für Edges vorzunehmen, tauschen Sie einfach die Spaltenüberschriften `~from` und `~to` gegeneinander aus und aktualisieren Sie die Werte der Spalte `~label`.

Als Alternative zur Umkehrung der Edge-Richtung können Sie einen vierten Neptune-Index, den OSGP-Index, aktivieren, der das Durchqueren eingehender Edges oder objektbasierte Suchvorgänge wesentlich effizienter macht. Die Aktivierung dieses vierten Index senkt jedoch die Einfügungsraten und erfordert mehr Speicherplatz.

Filteroptimierung bei der Migration von Daten von Neo4j zu Neptune

Neptune ist so optimiert, dass es am besten funktioniert, wenn Eigenschaften auf die selektivste verfügbare Eigenschaft gefiltert werden. Wenn mehrere Filter verwendet werden, wird für jeden Filter der Satz übereinstimmender Elemente gefunden und anschließend wird die Überlappung aller übereinstimmenden Sätze berechnet. Wenn möglich, wird durch die Kombination mehrerer Eigenschaften zu einer einzigen Eigenschaft die Anzahl der Index-Suchvorgänge minimiert und die Latenz einer Abfrage verringert.

Diese Abfrage verwendet beispielsweise zwei Index-Suchvorgänge und eine Verknüpfung:

```
MATCH (n) WHERE n.first_name='John' AND n.last_name='Doe' RETURN n
```

Diese Abfrage ruft dieselben Informationen mithilfe eines einzigen Index-Suchvorgangs ab:

```
MATCH (n) WHERE n.name='John Doe' RETURN n
```

Neptune unterstützt andere Datentypen als Neo4j.

Neo4j-Datentyp-Zuordnungen zu Datentypen, die Neptune unterstützt

- Logisch: `Boolean`

  Zuordnung in Neptune zu `Bool` oder `Boolean`.

- Numerisch: `Number`

  Zuordnung in Neptune zum engsten der folgenden Neptune-OpenCypher-Typen, der alle Werte der fraglichen numerischen Eigenschaft unterstützen kann:

  ```
  Byte
  Short
  Integer
  Long
  Float
  Double
  ```

- Text: `String`

  Zuordnung in Neptune zu `String`.

- Zeitpunkt:

```
Date
Time
LocalTime
DateTime
LocalDateTime
```

  Zuordnung in Neptune zu `Date` als UTC unter Verwendung eines der folgenden folgenden ISO-8601-Formate, die Neptune unterstützt:

```
yyyy-MM-dd
yyyy-MM-ddTHH:mm
yyyy-MM-ddTHH:mm:ss
yyyy-MM-ddTHH:mm:ssZ
```

- Zeitdauer: `Duration`

  Zuordnung in Neptune zu einem numerischen Wert für die Datumsarithmetik, falls erforderlich.

- Spatial: `Point`

  Zuordnung in Neptune zu numerischen Komponentenwerten, von denen jeder dann zu einer separaten Eigenschaft wird, oder Ausdruck als Zeichenfolgenwert, der von der Client-Anwendung interpretiert wird. Beachten Sie, dass Sie mithilfe der Volltextsuchintegration von Neptune Geolocation-Eigenschaften indizieren können. OpenSearch

Migration mehrwertiger Eigenschaften von Neo4j zu Neptune

Neo4j ermöglicht es, homogene Listen einfacher Typen als Eigenschaften von Knoten und Edges zu speichern. Diese Listen können doppelte Werte enthalten.

Neptune erlaubt jedoch nur feste oder einfache Kardinalität für Scheitelpunkteigenschaften und einfache Kardinalität für die Eigenschaften von Edges in Eigenschaftsgraphdaten. Daher gibt es keine einfache Migration von Neo4j-Knotenlisteneigenschaften, die doppelte Werte enthalten, zu Neptune-Scheitelpunkteigenschaften oder von Neo4j-Beziehungslisteneigenschaften zu Neptune-Edge-Eigenschaften.

Einige mögliche Strategien für die Migration mehrwertiger Neo4j-Knoteneigenschaften mit doppelten Werten zu Neptune sind:

- Verwerfen Sie die doppelten Werte und konvertieren Sie die mehrwertige Neo4j-Knoteneigenschaft in eine Neptune-Scheitelpunkteigenschaft mit festgelegter Kardinalität. Beachten Sie, dass der Neptune-Satz dann möglicherweise nicht die Reihenfolge der Elemente in der ursprünglichen mehrwertigen Neo4j-Eigenschaft widerspiegelt.
- Konvertieren Sie die mehrwertige Neo4j-Knoteneigenschaft zu einer Zeichenfolgendarstellung einer JSON-formatierten Liste in einer Neptune-Scheitelpunkt-Zeichenfolgeneigenschaft.
- Extrahieren Sie jeden der mehrwertigen Eigenschaftswerte in einen separaten Scheitelpunkt mit einer Werteigenschaft und verbinden Sie diese Scheitelpunkte mithilfe einer Edge, die mit dem Eigenschaftsnamen beschriftet ist, mit dem übergeordneten Scheitelpunkt.

Ähnlich sind einige mögliche Strategien für die Migration mehrwertiger Neo4j-Beziehungseigenschaften mit mehreren Werten zu Neptune:

- Konvertieren Sie die mehrwertige Neo4j-Beziehungseigenschaft zu einer Zeichenfolgendarstellung einer JSON-formatierten Liste und speichern Sie sie als Neptune-Edge-Zeichenfolgeneigenschaft.
- Führen Sie einen Faktorwechsel der Neo4j-Beziehung in eingehende und ausgehende Neptune-Edges, die an einen Zwischenscheitelpunkt angefügt sind, durch. Extrahieren Sie jeden der mehrwertigen Eigenschaftswerte in einen separaten Scheitelpunkt mit einer Werteigenschaft und verbinden Sie diese Scheitelpunkte mithilfe einer Edge, die mit dem Eigenschaftsnamen beschriftet ist, mit dem übergeordneten Scheitelpunkt.

Beachten Sie, dass die Zeichenfolgendarstellung einer Liste im JSON-Format für die openCypher-Abfragesprache intransparent ist, obwohl openCypher ein `CONTAINS`-Prädikat enthält, das einfache Suchen innerhalb von Zeichenfolgenwerten ermöglicht.

Exportieren von Daten aus Neo4j bei der Migration zu Neptune

Verwenden Sie beim Exportieren von Daten aus Neo4j die APOC-Verfahren, um entweder zuCSV oder zu GraphML zu exportieren. Obwohl es möglich ist, in andere Formate zu exportieren, gibt es Open-Source-Tools zum Konvertieren von CSV-Daten, die aus Neo4j exportiert wurden, in das Neptune-Bulk-Load-Format sowie Open-Source-Tools zum Konvertieren von GraphML-Daten, die aus Neo4j exportiert wurden, in das Neptune-Bulk-Load-Format.

Mit den verschiedenen APOC-Verfahren können Sie Daten auch direkt in Amazon S3 exportieren. Der Export in einen Amazon-S3-Bucket ist standardmäßig deaktiviert, kann aber mithilfe der Verfahren aktiviert werden, die unter Exportieren zu Amazon S3 in der Neo4j-APOC-Dokumentation beschrieben sind.

Importieren von Daten aus Neo4j bei der Migration zu Neptune

Sie können Daten entweder mithilfe des Neptune Bulk Loaders oder mithilfe der Anwendungslogik in einer unterstützten Abfragesprache wie openCypher in Neptune importieren.

Der Neptune Bulk Loader ist das bevorzugte Verfahren für den Import großer Datenmengen, da er eine optimierte Importleistung bietet, wenn Sie die Bewährten Methoden befolgen. Der Bulk Loader unterstützt zwei verschiedene CSV-Formate, in die aus Neo4j exportierte Daten mithilfe der oben im Abschnitt Exportieren von Daten genannten Open-Source-Dienstprogramme konvertiert werden können.

Sie können auch openCypher verwenden, um Daten mit benutzerdefinierter Logik für das Parsen, Transformieren und Importieren zu importieren. Sie können die openCypher-Abfragen entweder über den HTTPS-Endpunkt (was empfohlen wird) oder mithilfe des Bolt-Treibers übermitteln.

Anwendungsmigration von Neo4j zu Neptune

Nachdem Sie Ihre Daten von Neo4j zu Neptune migriert haben, besteht der nächste Schritt darin, die Anwendung selbst zu migrieren. Wie bei den Daten gibt es mehrere Vorgehensweise für die Migration Ihrer Anwendung, die auf den von Ihnen verwendeten Tools, Anforderungen, architektonischen Unterschieden usw. basieren. Dinge, die Sie bei diesem Prozess normalerweise berücksichtigen müssen, werden im Folgenden beschrieben.

Migration von Verbindungen von Neo4j zu Neptune

Wenn Sie die Bolt-Treiber derzeit nicht verwenden oder eine Alternative nutzen möchten, können Sie eine Verbindung zum HTTPS-Endpunkt herstellen, der vollständigen Zugriff auf die zurückgegebenen Daten bietet.

Wenn Sie eine Anwendung haben, die das Bolt-Protokoll verwendet, können Sie diese Verbindungen zu Neptune migrieren und Ihre Anwendungen mit denselben Treibern verbinden lassen wie in Neo4j. Um eine Verbindung zu Neptune herzustellen, müssen Sie möglicherweise eine oder mehrere der folgenden Änderungen an Ihrer Anwendung vornehmen:

- Die URL und der Port müssen aktualisiert werden, um die Cluster-Endpunkte und den Cluster-Port verwenden zu können (die Standardeinstellung ist 8182).
- Neptune erfordert, dass alle Verbindungen SSL verwenden, daher müssen Sie für jede Verbindung angeben, dass sie verschlüsselt ist.
- Neptune verwaltet die Authentifizierung durch die Zuweisung von IAM-Richtlinien und -Rollen. IAM-Richtlinien und -Rollen ermöglichen eine äußerst flexible Benutzerverwaltung innerhalb der Anwendung. Daher ist es wichtig, die Informationen in der IAM-Übersicht zu lesen und zu verstehen, bevor Sie Ihren Cluster konfigurieren.
- Bolt-Verbindungen verhalten sich in Neptune in mehrfacher Hinsicht anders als in Neo4j, wie in Bolt-Verbindungsverhalten in Neptune erläutert.
- Weitere Informationen und Vorschläge finden Sie unter Bewährte Methoden für Neptune mit openCypher und Bolt.

Codebeispiele für häufig verwendete Sprachen wie Java, Python, .NET und NodeJS sowie für Verbindungsszenarien wie die Verwendung der IAM-Authentifizierung finden Sie in Verwenden des Bolt-Protokolls für openCypher-Abfragen an Neptune.

Weiterleiten von Abfragen an Cluster-Instances beim Übergang von Neo4j zu Neptune

Neo4j-Clientanwendungen verwenden einen Routing-Treiber und spezifizieren einen Zugriffsmodus, um Lese- und Schreibanforderungen an einen geeigneten Server in einem Kausal-Cluster weiterzuleiten.

Wenn Sie eine Client-Anwendung zu Neptune migrieren, verwenden Sie Neptune-Endpunkte, um Abfragen effizient an eine geeignete Instance in Ihrem Cluster weiterzuleiten:

- Alle Verbindungen zu Neptune sollten `bolt://` statt `bolt+routing://` oder `neo4j://` in der URL verwenden.
- Der Cluster-Endpunkt verbindet sich mit der aktuellen primären Instance in Ihrem Cluster. Verwenden Sie den Cluster-Endpunkt, um Schreibanforderungen an die primäre Instance weiterzuleiten.
- Der Reader-Endpunkt verteilt Verbindungen auf Lesereplikat-Instances in Ihrem Cluster. Wenn Sie einen Single-Instance-Cluster ohne Lesereplikat-Instances haben, stellt der Reader-Endpunkt eine Verbindung zur primären Instance her, die Schreibvorgänge unterstützt. Wenn der Cluster eine oder mehrere Lesereplikat-Instances enthält, generiert das Senden einer Schreibanforderung an den Reader-Endpunkt eine Ausnahme.
- Jede Instance in Ihrem Cluster kann auch einen eigenen Instance-Endpunkt haben. Verwenden Sie einen Instance-Endpunkt, wenn Ihre Client-Anwendung eine Anfrage an eine bestimmte Instance im Cluster senden muss.

Weitere Informationen finden Sie unter Überlegungen zu Neptune-Endpunkten.

Datenkonsistenz in Neptune

Bei der Verwendung von Neo4j-Kausal-Clustern sind Lesereplikate letztlich konsistent mit den Core-Servern, aber Client-Anwendungen können die kausale Konsistenz sicherstellen, indem sie die kausale Verkettung verwenden. Bei der kausalen Verkettung werden Lesezeichen zwischen Transaktionen übergeben, so dass eine Client-Anwendung auf einen Hauptserver schreiben und dann ihren eigenen Schreibvorgang aus einem Lesereplikat lesen kann.

In Neptune sind Lesereplikat-Instances letztlich mit dem Writer konsistent, wobei die Replikatverzögerung normalerweise weniger als 100 Millisekunden beträgt. Aktualisierungen vorhandener Edges und Scheitelpunkte sowie das Hinzufügen neuer Edges und Scheitelpunkte sind auf einer Replikat-Instance jedoch erst sichtbar, wenn eine Änderung repliziert wurde. Wenn

Ihre Anwendung sofortige Konsistenz auf Neptune benötigt, indem sie jeden Schreibvorgang liest, verwenden Sie daher den Cluster-Endpunkt für den read-after-write Vorgang. Dies ist das einzige Mal, dass der Cluster-Endpunkt für Lesevorgänge genutzt wird. Verwenden Sie unter allen anderen Umständen den Reader-Endpunkt für Lesevorgänge.

Migrieren von Abfragen von Neo4j zu Neptune

Obwohl die Unterstützung für openCypher in Neptune den Arbeitsaufwand für die Migration von Abfragen aus Neo4j drastisch reduziert, sind bei der Migration dennoch einige Unterschiede zu berücksichtigen:

- Wie bereits unter Datenmodelloptimierungen erwähnt, müssen Sie möglicherweise Änderungen an Ihrem Datenmodell vornehmen, um ein optimiertes Graphdatenmodell für Neptune zu erstellen, was wiederum Änderungen an Ihren Abfragen und Testverfahren erfordert.
- Neo4j bietet eine Vielzahl von für Cypher spezifischen Spracherweiterungen, die nicht in der von Neptune implementierten openCypher-Spezifikation enthalten sind. Je nach Anwendungsfall und verwendetem Feature gibt es möglicherweise Problemumgehungen innerhalb der openCypher-Sprache bzw. mithilfe der Gremlin-Sprache oder durch andere Mechanismen, wie unter Umschreiben von Cypher-Abfragen zur Ausführung in openCypher auf Neptune beschrieben.
- Anwendungen verwenden anstelle der Bolt-Treiber selbst häufig andere Middleware-Komponenten, um mit der Datenbank zu interagieren. Bitte überprüfen Sie Neptune-Kompatibilität mit Neo4j, um festzustellen, ob die von Ihnen verwendeten Tools oder Middleware-Komponenten unterstützt werden.
- Im Falle eines Failovers stellt der Bolt-Treiber möglicherweise weiterhin eine Verbindung zur vorherigen Writer- oder Reader-Instance her, da der für die Verbindung bereitgestellte Cluster-Endpunkt zu einer IP-Adresse aufgelöst wurde. Die korrekte Fehlerbehandlung in Ihrer Anwendung sollte damit wie unter Erstellen einer neuen Verbindung nach einem Failover beschrieben umgehen.
- Wenn Transaktionen aufgrund von nicht auflösbaren Konflikten oder der Überschreitung von Sperrwartezeiten abgebrochen werden, reagiert Amazon Neptune mit einer `ConcurrentModificationException`. Weitere Informationen finden Sie unter Engine-Fehlercodes. Als bewährte Methode sollten Clients diese Ausnahmen immer abfangen und verarbeiten.

  Eine `ConcurrentModificationException` tritt gelegentlich auf, wenn mehrere Threads oder mehrere Anwendungen gleichzeitig in das System schreiben. Aufgrund der Isolationsstufen für Transaktionen können solche Konflikte manchmal unvermeidlich sein.

- Neptune unterstützt die Ausführung von Gremlin- und OpenCypher-Abfragen für dieselben Daten. Dies bedeutet, dass Sie in manchen Szenarien möglicherweise in Betracht ziehen müssen, Gremlin mit seinen leistungsfähigeren Abfragefunktionen zu verwenden, um einige der Funktionen Ihrer Abfragen auszuführen.

Wie bereits in Bereitstellen der Infrastruktur erwähnt, sollte für jede Anwendung die Größe korrekt angepasst werden, um sicherzustellen, dass die Anzahl der Instances, die Instance-Größen und die Cluster-Topologie sämtlich für den spezifischen Workload der Anwendung optimiert sind.

Die hier besprochenen Überlegungen zur Migration Ihrer Anwendung sind die üblichsten, diese Liste ist aber nicht vollständig. Jede Anwendung ist einzigartig. Bitte wenden Sie sich an den AWS Support oder wenden Sie sich an Ihr Account-Team, wenn Sie weitere Fragen haben.

Migration von Features und Tools, die spezifisch für Neo4j sind

Neo4j bietet eine Vielzahl von benutzerdefinierten Features und Add-Ons mit Funktionen, die Ihre Anwendung nutzen kann. Bei der Bewertung der Notwendigkeit, diese Funktionalität zu migrieren, ist es oft hilfreich, zu untersuchen, ob es einen besseren Ansatz gibt AWS , um dasselbe Ziel zu erreichen. Angesichts der architektonischen Unterschiede zwischen Neo4j und Neptune können Sie oft effektive Alternativen finden, die andere AWS Dienste oder Integrationen nutzen.

Eine Liste der für Neo4J spezifischen Features und Vorschläge für Problemumgehungen finden Sie unter Neptune-Kompatibilität mit Neo4j.

Neptune-Kompatibilität mit Neo4j

Neo4j verfolgt einen all-in-one architektonischen Ansatz, bei dem das Laden von Daten, Daten-ETL, Anwendungsabfragen, Datenspeicherung und Verwaltungsvorgänge alle in demselben Satz von Rechenressourcen, wie EC2 z. B. Instanzen, ablaufen. Amazon Neptune ist eine OLTP-orientierte Graphdatenbank mit offenen Spezifikationen, bei der die Architektur Abläufe voneinander trennt und Ressourcen entkoppelt, so dass sie dynamisch skaliert werden können.

Neo4j bietet eine Vielzahl von Features und Tools, einschließlich Tools von Drittanbietern, die nicht Teil der openCypher-Spezifikation, mit openCypher oder der openCypher-Implementierung von Neptune nicht kompatibel sind. Im Folgenden sind einige der gängigsten davon aufgeführt.

Für Neo4j spezifische Features, die in Neptune nicht vorhanden sind

- **`LOAD CSV`** – Neptune verfolgt ein anderes architektonisches Konzept zum Laden von Daten als Neo4j. Um eine bessere Skalierung und Kostenoptimierung zu ermöglichen, implementiert Neptune die Trennung der Bedenken in Bezug auf Ressourcen und empfiehlt, eine der AWS -Serviceintegrationen wie etwa AWS Glue zu verwenden, um die erforderlichen ETL-Prozesse durchzuführen, um Daten in einem Format vorzubereiten, das vom Neptune Bulk Loader unterstützt wird.

  Eine andere Möglichkeit besteht darin, dasselbe mit Anwendungscode zu tun, der auf AWS Rechenressourcen wie EC2 Amazon-Instances, Lambda-Funktionen, Amazon Elastic Container Service, AWS Batch Jobs usw. ausgeführt wird. Der Code könnte entweder den HTTPS-Endpunkt oder den Bolt-Endpunkt von Neptune verwenden.
- Präzise Zugriffskontrolle – Neptune unterstützt eine detaillierte Zugriffskontrolle für Datenzugriffsaktionen mithilfe von IAM-Bedingungsschlüsseln. Auf Anwendungsebene kann eine zusätzliche detaillierte Zugriffskontrolle implementiert werden.
- Neo4j Fabric – Neptune unterstützt den datenbankübergreifenden Abfrageverbund für RDF-Workloads mithilfe des SPARQL-Schlüsselworts `SERVICE`. Da es derzeit keinen offenen Standard und keine Spezifikation für den Abfrageverbund für Eigenschaftsgraph-Workloads gibt, müsste diese Funktionalität auf Anwendungsebene implementiert werden.
- Rollenbasierte Zugriffskontrolle (RBAC) – Neptune verwaltet die Authentifizierung durch die Zuweisung von IAM-Richtlinien und -Rollen. IAM-Richtlinien und -Rollen ermöglichen eine äußerst flexible Benutzerverwaltung innerhalb einer Anwendung. Daher ist es wichtig, die Informationen in der IAM-Übersicht zu lesen und zu verstehen, bevor Sie Ihren Cluster konfigurieren.

- Bookmarking – Neptune-Cluster bestehen aus einer einzigen Writer-Instance und bis zu 15 Lesereplikat-Instances. In die Writer-Instance geschriebene Daten sind ACID-konform und bieten eine starke Konsistenzgarantie bei nachfolgenden Lesevorgängen. Lesereplikate verwenden dasselbe Speicher-Volume wie die Writer-Instance und sind letztendlich konsistent, normalerweise in weniger als 100 ms nach dem Schreiben der Daten. Wenn in Ihrem Anwendungsfall die Lesekonsistenz neuer Schreibvorgänge sofort gewährleistet sein muss, sollten diese Lesevorgänge an den Cluster-Endpunkt statt an den Reader-Endpunkt geleitet werden.
- APOC-Verfahren – Da APOC-Verfahren nicht in der openCypher-Spezifikation enthalten sind, bietet Neptune keine direkte Unterstützung für externe Verfahren. Stattdessen setzt Neptune auf Integrationen mit anderen AWS Diensten, um ähnliche Endbenutzerfunktionen auf skalierbare, sichere und robuste Weise zu erreichen. Manchmal können APOC-Prozeduren in openCypher oder Gremlin umgeschrieben werden und manche sind für Neptune-Anwendungen nicht relevant.

  Im Allgemeinen lassen sich APOC-Verfahren in die folgenden Kategorien einteilen:

  - Import – Neptune unterstützt den Import von Daten in einer Vielzahl von Formaten mithilfe von Abfragesprachen, dem Neptune-Bulk-Loader oder als Ziel von AWS Database Migration Service. ETL-Operationen an Daten können unter Verwendung des `neptune-python-utils`Open-Source-Pakets AWS Glue ausgeführt werden.
  - Export – Neptune unterstützt den Export von Daten mithilfe des `neptune-export`-Dienstprogramms, das eine Vielzahl gängiger Exportformate und -methoden unterstützt.
  - Datenbankintegration — Neptune unterstützt die Integration mit anderen Datenbanken mithilfe von ETL-Tools wie AWS Glue oder Migrationstools wie dem. AWS Database Migration Service
  - Graph-Updates – Neptune unterstützt eine Vielzahl von Features zur Aktualisierung von Eigenschaftsgraphdaten, da es die Abfragesprachen openCypher und Gremlin unterstützt. Beispiele für Umschreibungen häufig verwendeter Verfahren finden Sie unter Cypher-Umschreibungen.
  - Datenstrukturen – Neptune unterstützt eine Vielzahl von Features zur Aktualisierung von Eigenschaftsgraphdaten, da es die Abfragesprachen openCypher und Gremlin unterstützt. Beispiele für Umschreibungen häufig verwendeter Verfahren finden Sie unter Cypher-Umschreibungen.
  - Zeitdaten (Datum/Uhrzeit) – Neptune unterstützt eine Vielzahl von Features zur Aktualisierung von Eigenschaftsgraphdaten, da es die Abfragesprachen openCypher und Gremlin unterstützt. Beispiele für Umschreibungen häufig verwendeter Verfahren finden Sie unter Cypher-Umschreibungen.

  - Mathematische Daten – Neptune unterstützt eine Vielzahl von Features zur Aktualisierung von Eigenschaftsgraphdaten, da es die Abfragesprachen openCypher und Gremlin unterstützt. Beispiele für Umschreibungen häufig verwendeter Verfahren finden Sie unter Cypher-Umschreibungen.
  - Erweiterte Graph-Abfragen – Neptune unterstützt eine Vielzahl von Features zur Aktualisierung von Eigenschaftsgraphdaten, da es die Abfragesprachen openCypher und Gremlin unterstützt. Beispiele für Umschreibungen häufig verwendeter Verfahren finden Sie unter Cypher-Umschreibungen.
  - Graph-Vergleich – Neptune unterstützt eine Vielzahl von Features zur Aktualisierung von Eigenschaftsgraphdaten, da es die Abfragesprachen openCypher und Gremlin unterstützt. Beispiele für Umschreibungen häufig verwendeter Verfahren finden Sie unter Cypher-Umschreibungen.
  - Cypher-Ausführung – Neptune unterstützt eine Vielzahl von Features zur Aktualisierung von Eigenschaftsgraphdaten, da es die Abfragesprachen openCypher und Gremlin unterstützt. Beispiele für Umschreibungen häufig verwendeter Verfahren finden Sie unter Cypher-Umschreibungen.
- Benutzerdefinierte Verfahren – Neptune unterstützt keine von Benutzern erstellten Verfahren. Diese Funktionalität müsste auf Anwendungsebene implementiert werden.
- Geospatial — Obwohl Neptune keine native Unterstützung für Geodatenfunktionen bietet, können ähnliche Funktionen durch die Integration mit anderen AWS Diensten erreicht werden, wie in diesem Blogbeitrag gezeigt wird: Kombinieren Sie Amazon Neptune und Amazon OpenSearch Service für Geodatenabfragen von Ross Gabay und Abhilash Vinod (1. Februar 2022).
- Graph Data Science – Neptune unterstützt Graphanalysen heute über Neptune Analytics, eine speicheroptimierte Engine, die eine Bibliothek von Graphanalysealgorithmen unterstützt.

  Neptune bietet außerdem eine Integration mit dem AWS -Pandas-SDK und mehrere Beispiel-Notebooks, die zeigen, wie diese Integration in Python-Umgebungen genutzt werden kann, um Analysen von Graphdaten durchzuführen.

- Schema-Einschränkungen – In Neptune ist die einzige verfügbare Schemaeinschränkung die Eindeutigkeit der ID eines Knotens oder eines Edges. Es gibt kein Feature, mit dem zusätzliche Schema- bzw. Eindeutigkeits- oder Werteinschränkungen für ein Element in einem Graphen angegeben werden könnten. ID-Werte in Neptune sind Zeichenfolgen und können mit Gremlin wie folgt gesetzt werden:

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

Für Anwendungen, die eine Eindeutigkeitseinschränkung nutzen müssen, sollte diese Vorgehensweise in Betracht gezogen werden, um eine solche zu erreichen. Wenn die Anwendung mehrere Spalten als Eindeutigkeitseinschränkung verwendet hat, kann die ID auf eine Kombination dieser Werte gesetzt werden. `id=123, code='SEA'` könnte beispielsweise als `ID='123_SEA'` repräsentiert werden, so dass eine komplexe Eindeutigkeitseinschränkung erreicht wird.

- Mehrere Mandanten – Neptune unterstützt nur einen Graphen pro Cluster. Um ein Mehrmandantensystem mit Neptune aufzubauen, verwenden Sie entweder mehrere Cluster oder partitionieren Sie die Mandanten logisch in einem einzigen Graphen und verwenden anwendungsseitige Logik, um die Trennung zu erzwingen. Fügen Sie beispielsweise eine `tenantId`-Eigenschaft wie folgt in in jede Abfrage ein:

  ```
  MATCH p=(n {tenantId:1})-[]->({tenantId:1}) RETURN p LIMIT 5)
  ```

  Neptune Serverless macht es relativ einfach, Multi-Tenancy mithilfe mehrerer DB-Cluster zu implementieren, von denen jeder unabhängig und automatisch nach Bedarf skaliert wird.

Neptune-Unterstützung für Neo4j-Tools

Neptune bietet die folgenden Alternativen zu Neo4j-Tools:

- Neo4j Browser – Neptune bietet Open-Source-Graph-Notebooks, die eine entwicklerorientierte IDE zum Ausführen von Abfragen und zur Visualisierung der Ergebnisse bieten.
- Neo4j Bloom – Neptune unterstützt umfangreiche Graphvisualisierungen mithilfe von Visualisierungslösungen von Drittanbietern wie Graph-Explorer, Tom Sawyer, Cambridge Intelligence, Graphistry, Metaphacts und G.V().
- GraphQL — Neptune unterstützt GraphQL derzeit durch benutzerdefinierte Integrationen. AWS AppSync Weitere Informationen finden Sie im Blogbeitrag Build a graph application with Amazon Neptune and AWS Amplify und im Beispielprojekt Building Serverless Calorie Tracker application with AWS AppSync and Amazon Neptune.
- NeoSemantics— Neptune unterstützt das RDF-Datenmodell nativ. Kunden, die RDF-Workloads ausführen möchten, wird daher empfohlen, die RDF-Modellunterstützung von Neptune zu verwenden.

- Arrows.app – Die Cypher, die beim Exportieren des Modells mit dem Exportbefehl erstellt wurde, ist mit Neptune kompatibel.
- Linkurious Ogma – Eine Beispielintegration mit Linkurious Ogma ist hier verfügbar.
- Spring Data Neo4j – Dies ist derzeit nicht mit Neptune kompatibel.
- Neo4j Spark Connector – Der Neo4j Spark Connector kann innerhalb eines Spark-Auftrags verwendet werden, um mithilfe von openCypher eine Verbindung zu Neptune herzustellen. Hier finden Sie Beispielcode und eine Anwendungskonfiguration:

  Beispiel-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();
```

  Anwendungskonfiguration:

```
<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>
```

Neo4j-Features und -Tools, die hier nicht aufgeführt sind

Wenn Sie ein Tool oder eine Funktion verwenden, die hier nicht aufgeführt ist, sind wir uns nicht sicher, ob es mit Neptune oder anderen darin enthaltenen Diensten kompatibel ist. AWS Bitte wenden Sie sich an den AWS Support oder wenden Sie sich an Ihr Account-Team, wenn Sie weitere Fragen haben.

Umschreiben von Cypher-Abfragen zur Ausführung in openCypher auf Neptune

openCypher ist eine deklarative Abfragesprache für Eigenschaftsdiagramme. Ursprünglich von Neo4j entwickelt, wurde sie 2015 als Open-Source-Software veröffentlicht und ist unter einer Apache-2-Open-Source-Lizenz für das openCypher-Projekt verfügbar. Wir bei glauben AWS, dass Open Source für alle gut ist, und wir setzen uns dafür ein, unseren Kunden den Wert von Open Source und den Open-Source-Communities die operative Exzellenz AWS zu bieten.

OpenCypher Die Syntax ist in der Cypher Query Language Reference, Version 9, dokumentiert.

Da OpenCypher eine Teilmenge der Syntax und Features der Cypher-Abfragesprache enthält, erfordern einige Migrationsszenarien entweder das Umschreiben von Abfragen in mit OpenCypher kompatible Form oder die Prüfung alternativer Methoden, um die gewünschte Funktionalität zu erreichen.

Dieser Abschnitt enthält Empfehlungen zum Umgang mit häufigen Unterschieden, die jedoch keineswegs erschöpfend sind. Sie sollten jede Anwendung, die diese Umschreibungen verwendet, gründlich testen, um sicherzustellen, dass die Ergebnisse Ihren Erwartungen entsprechen.

Umschreiben von **`None`**-, **`All`**- und **`Any`**-Prädikatfunktionen

Diese Funktionen sind nicht Teil der openCypher-Spezifikation. Vergleichbare Ergebnisse können mit List Comprehension in openCypher erzielt werden.

Finden Sie zum Beispiel alle Pfade, die von Knoten `Start` zu Knoten `End` führen, wobei aber kein Pfad einen Knoten passieren darf, dessen Klasseneigenschaft `D` ist:

```
# 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
```

List Comprehension kann diese Ergebnisse können wie folgt erreichen:

```
all  => size(list_comprehension(list)) = size(list)
```

```
any  => size(list_comprehension(list)) >= 1
none => size(list_comprehension(list)) = 0
```

Umschreiben der Cypher-Funktion **`reduce()`** zu openCypher

Die `reduce()`-Funktion ist nicht Teil der openCypher-Spezifikation. Sie wird häufig verwendet, um eine Aggregierung von Daten aus Elementen innerhalb einer Liste zu erstellen. In vielen Fällen können Sie eine Kombination aus List Comprehension und der `UNWIND`-Klausel verwenden, um ähnliche Ergebnisse zu erreichen.

Die folgende Cypher-Abfrage findet beispielsweise alle Flughäfen auf Wegen mit einer bis drei Stopps zwischen Anchorage (ANC) und Austin (AUS) und gibt die Gesamtentfernung für jeden Pfad zurück:

```
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
```

Sie können dieselbe Abfrage in openCypher für Neptune wie folgt schreiben:

```
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
```

Umschreiben der Cypher-Klausel FOREACH zu openCypher

Die FOREACH–Klausel ist nicht Teil der openCypher-Spezifikation. Sie wird häufig verwendet, um Daten während einer Abfrage zu aktualisieren, oft anhand von Aggregierungen oder Elementen innerhalb eines Pfads.

Finden Sie als Beispiel für einen Pfad nach allen Flughäfen auf einem Pfad mit nicht mehr als zwei Stopps zwischen Anchorage (ANC) und Austin (AUS) und legen Sie für jeden von ihnen die Eigenschaft „besucht“ fest:

```
# 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
```

Ein weiteres Beispiel ist:

```
# 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
```

Umschreiben der Neo4j-APOC-Prozeduren zu Neptune

In den folgenden Beispielen wird openCypher verwendet, um einige der am häufigsten verwendeten APOC-Prozeduren zu ersetzen. Diese Beispiele dienen nur als Referenz und sollen einige Vorschläge zum Umgang mit gängigen Szenarien geben. In der Praxis ist jede Anwendung anders und Sie müssen Ihre eigenen Strategien entwickeln, um alle Funktionen bereitzustellen, die Sie benötigen.

Umschreiben von **`apoc.export`**-Verfahren

Neptune bietet mithilfe des Neptune-Export-Dienstprogramms eine Reihe von Optionen sowohl für vollständige Graph- als auch für abfragebasierte Exporte in verschiedenen Ausgabeformaten wie CSV und JSON (siehe Exportieren von Daten aus einem Neptune-DB-Cluster).

Umschreiben von **`apoc.schema`**-Verfahren

Neptune hat kein explizit definiertes Schema, keine Indizes oder Einschränkungen, so dass viele `apoc.schema`-Prozeduren nicht mehr erforderlich sind. Beispiele sind:

- `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 unterstützt das Abrufen ähnlicher Werte wie die Prozeduren, wie unten gezeigt, kann jedoch bei größeren Graphen zu Leistungsproblemen führen, da dafür ein großer Teil des Graphen gescannt werden muss, um die Antwort zurückzugeben.

```
# 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)
```

Alternativen zu **`apoc.do`**-Prozeduren

Diese Prozeduren werden verwendet, um eine bedingte Abfrageausführung zu ermöglichen, die mit anderen OpenCypher-Klauseln einfach zu implementieren ist. In Neptune gibt es mindestens zwei Möglichkeiten, ein ähnliches Verhalten zu erreichen:

- Eine Möglichkeit besteht darin, die Listenverständnisfunktionen von OpenCypher mit der `UNWIND`-Klausel zu kombinieren.
- Eine andere Möglichkeit besteht darin, die Schritte choose() und coalesce() in Gremlin zu verwenden.

Beispiele für diese Vorgehensweisen sind unten aufgeführt.

Alternativen zu 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())
```

Alternativen zu apoc.do.when

```
# 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())
```

Alternativen zu listenbasierten Eigenschaften

Neptune unterstützt derzeit nicht das Speichern von auf Listen basierenden Eigenschaften. Ähnliche Ergebnisse können jedoch erzielt werden, wenn Listenwerte als kommagetrennte Zeichenfolge gespeichert und dann die Listeneigenschaft mithilfe der Funktionen `join()` und `split()` konstruiert und dekonstruiert wird.

Wenn wir beispielsweise eine Liste von Tags als Eigenschaft speichern möchten, könnten wir die Beispiel-Umschreibung verwenden, die zeigt, wie eine durch Kommata getrennte Eigenschaft abgerufen wird, und dann die Funktionen `split()` und `join()` mit List Comprehension verwenden, um vergleichbare Ergebnisse zu erzielen:

```
# 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
```

CALL-Unterabfragen umschreiben

`CALL`Neptune-Unterabfragen unterstützen die Syntax `CALL (friend) { ... }` für den Import von Variablen in den Unterabfragebereich nicht (in diesem `friend` Beispiel). Bitte verwenden Sie die `WITH` Klausel innerhalb der Unterabfrage für dasselbe, z. B. `CALL { WITH friend ... }`

Optionale `CALL` Unterabfragen werden derzeit nicht unterstützt.

Weitere Unterschiede zwischen Neptune openCypher und Cypher

- Neptune unterstützt nur TCP-Verbindungen für das Bolt-Protokoll. WebSocketsVerbindungen für Bolt werden nicht unterstützt.
- Neptune openCypher entfernt Leerzeichen in den Funktionen `trim()`, `ltrim()` und `rtrim()`, wie in Unicode definiert.
- In Neptune openCypher wechselt `tostring(double)` bei großen Double-Werten nicht automatisch zur E-Notation.

Ressourcen für die Migration von Neo4j zu Neptune

Neptune bietet verschiedene Tools und Ressourcen, die den Migrationsprozess unterstützen können.

Tools zur Unterstützung der Migration von Neo4j zu Neptune

- Die CheatSheetOpenCypher.
- neo4 j-to-neptune — Ein Befehlszeilenprogramm für die Migration von Daten von Neo4j nach Neptune. Dieses Tool bietet folgende Funktionen:
  - Exportieren Sie die Daten aus einem ordnungsgemäß konfigurierten Neo4j-Diagramm.
  - Konvertiert diese Daten in das Neptun-Format.
  - Laden Sie diese Daten massenweise in Neptune.
  - Führen Sie während der Konvertierung in das Neptun-Format einige grundlegende Datenkonvertierungen durch, z. B. das Umbenennen von Scheitelpunkt- oder Kantenbeschriftungen und das Generieren von Elementen.
  - Generieren Sie Eigenschaften für Knoten und Kanten mithilfe von Vorlagen (erstellen Sie beispielsweise einen `~id` Wert mithilfe einer Vorlage, z. B. `Person_{personid}` für Situationen, in denen Sie die eindeutige Kennung für ein Element erstellen müssen).
- OpenCypher Query Compatibility Checker — Dieses Tool nimmt die Eingabe von OpenCypher-Abfragen entgegen und wird:
  - Prüfen Sie die Kompatibilität mit der ausgewählten Version von Neptune.
  - Identifizieren Sie bestimmte Funktionen und Klauseln, die nicht unterstützt werden, mit ihren Positionen.
  - Schlagen Sie, falls verfügbar, Ersatzprodukte vor.
  - Geben Sie Fehlerbeschreibungen für alle anderen Syntaxfehler an.

Migrieren eines vorhandenen Graphen von einem Apache TinkerPop Gremlin-Server zu Amazon Neptune

Wenn Sie Graphdaten auf einem Apache TinkerPop Gremlin Server haben, die Sie zu Amazon Neptune migrieren möchten, würden Sie die folgenden Schritte ausführen:

1. Exportieren Sie die Daten vom Gremlin-Server in Amazon Simple Storage Service (Amazon S3).
2. Konvertieren Sie die exportierten Daten in ein Format, das der Neptune Bulk Loader importieren kann.
3. Importieren Sie mithilfe von Neptune-Massen-Loader die Daten in einen Neptune-DB-Cluster, den Sie vorbereitet haben.
4. Ändern Sie Ihre bestehende Anwendung so, dass sie eine Verbindung zum Gremlin-Endpunkt von Neptune herstellt, und nehmen Sie alle erforderlichen Änderungen vor, um den Unterschieden bei der Neptune Gremlin-Implementierung zu entsprechen.

Migrieren eines vorhandenen Graphen von einem RDF-Triple-Store zu Amazon Neptune

Wenn Sie über Grafikdaten verfügen und RDF/SPARQL zu Amazon Neptune migrieren möchten, gehen Sie wie folgt vor:

1. Exportieren Sie die Daten aus Ihrem RDF-Triple-Store.
2. Konvertieren Sie die exportierten Daten in ein Format, das der Neptune Bulk Loader importieren kann.
3. Speichern Sie die zu importierenden Daten in Amazon Simple Storage Service (Amazon S3).
4. Importieren Sie mithilfe von Neptune-Massen-Loader die Daten von Amazon S3 in einen Neptune-DB-Cluster, den Sie vorbereitet haben.
5. Ändern Sie Ihre bestehende Anwendung, um eine Verbindung zum SPARQL-Endpunkt von Neptune herzustellen.

Wenn Sie ausprobieren möchten, CSV-Daten aus Eigenschaftsgraphen zu RDF zu migrieren, können Sie den Amazon-Neptune-CSV-zu-RDF-Konverter verwenden.

Verwenden von AWS Database Migration Service (AWS DMS) zur Migration von einer relationalen oder NoSQL-Datenbank zu Amazon Neptune

AWS Database Migration Service (AWS DMS) ist ein Cloud-Dienst, mit dem relationale Datenbanken, Data Warehouses, NoSQL-Datenbanken und andere Arten von Datenspeichern einfach migriert werden können. Wenn Sie Graphdaten in einer der AWS DMS unterstützten relationalen oder NoSQL-Datenbanken gespeichert haben, AWS DMS kann dies Ihnen helfen, schnell und sicher zu Neptune zu migrieren, ohne dass Ausfallzeiten von Ihrer aktuellen Datenbank erforderlich sind. Details dazu finden Sie unter Verwenden AWS Database Migration Service um Daten aus einem anderen Datenspeicher in Amazon Neptune zu laden.

Der verwendete Migrationsdatenfluss sieht wie folgt aus: AWS DMS

- Erstellen Sie ein AWS DMS Tabellenzuordnungsobjekt. Dieses JSON-Objekt gibt an, welche Tabellen in welcher Reihenfolge gelesen werden sollen und wie deren Spalten benannt werden. Es kann auch die zu kopierenden Zeilen filtern und einfache Werttransformationen bereitstellen, wie zum Beispiel das Konvertieren in Kleinbuchstaben oder Rundungen.
- Erstellen Sie ein Neptune-`GraphMappingConfig`, um anzugeben, wie die aus der Quelldatenbank extrahierten Daten in Neptune geladen werden sollen.
  - Für RDF-Daten (mit SPARQL abgefragt) wird `GraphMappingConfig` in der Standard-Zuweisungssprache R2RM von W3 geschrieben.
  - Bei Eigenschaftsgraphdaten (abgefragt mit Gremlin) handelt es sich bei `GraphMappingConfig` um ein JSON-Objekt, wie unter GraphMappingConfig Layout für Property-Graph/Gremlin Daten beschrieben.
- Erstellen Sie eine AWS DMS Replikationsinstanz in derselben VPC wie Ihr Neptune-DB-Cluster, um die Migration durchzuführen.
- Erstellen Sie einen Amazon-S3-Bucket als Zwischenspeicher für die Bereitstellung der Migrationsdaten.
- Führen Sie die Migrationsaufgabe aus. AWS DMS

Einzelheiten finden Sie unter Verwenden AWS Database Migration Service um Daten aus einem anderen Datenspeicher in Amazon Neptune zu laden und in Chris Smiths vierteiligem Blog-Beitrag „Auffüllen Ihres Graphen in Amazon Neptune aus einer relationalen Datenbank mithilfe des AWS Database Migration Service (DMS):“

- Teil 1: Die Voraussetzungen
- Teil 2: Entwerfen des Eigenschaftsgraphmodells
- Teil 3: Entwerfen des RDF-Modells
- Part 4: Zusammenführung

Migration von Blazegraph zu Amazon Neptune

Wenn Sie einen Graphen im Open-Source-Blazegraph-RDF-Triplestore haben, können Sie mithilfe der folgenden Schritte zu Ihren Graph-Daten in Amazon Neptune migrieren:

- AWS Infrastruktur bereitstellen. Stellen Sie zunächst die erforderliche Neptune-Infrastruktur mithilfe einer AWS CloudFormation Vorlage bereit (siehe). Neptun-Cluster erstellen
- Exportieren von Daten aus Blazegraph. Es gibt zwei Hauptmethoden für den Export von Daten aus Blazegraph, nämlich die Verwendung von SPARQL CONSTRUCT-Abfragen oder die Verwendung des Blazegraph-Export-Dienstprogramms.
- Importieren der Daten in Neptune. Anschließend können Sie die exportierten Datendateien mit der Neptune-Workbench und Neptune-Massen-Loader in Neptune laden.

Diese Vorgehensweise ist auch generell für die Migration von anderen RDF-Triplestore-Datenbanken anwendbar.

Kompatibilität von Blazegraph mit Neptune

Bevor Sie Ihre Grafikdaten zu Neptune migrieren, sollten Sie sich einiger signifikanter Unterschiede zwischen Blazegraph und Neptune bewusst sein. Diese Unterschiede können Änderungen an den Abfragen, der Anwendungsarchitektur oder beidem erfordern oder die Migration sogar unmöglich machen:

- **`Full-text search`** – In Blazegraph können Sie entweder die interne Volltextsuche oder externe Volltextsuchfunktionen über eine Integration mit Apache Solr verwenden. Wenn Sie eines dieser Features verwenden, bleiben Sie bei den neuesten Updates der von Neptune unterstützten Volltextsuch-Features auf dem Laufenden. Siehe Neptun-Volltextsuche.
- **`Query hints`** – Sowohl Blazegraph als auch Neptune erweitern SPARQL mithilfe des Konzepts der Abfragehinweise. Während einer Migration müssen Sie alle von Ihnen verwendeten Abfragehinweise migrieren. Hinweise zu den neuesten Abfragehinweisen, die Neptune unterstützt, finden Sie unter SPARQL-Abfragehinweise.
- Inferenz – Blazegraph unterstützt Inferenz als konfigurierbare Option im Triple-Modus, aber nicht im Quads-Modus. Neptune unterstützt derzeit noch keine Inferenz.
- Geospatiale Suche – Blazegraph unterstützt die Konfiguration von Namespaces, die Geodatenunterstützung ermöglichen. Dieses Feature ist in Neptune noch nicht verfügbar.

- Mehrmandantenfähigkeit – Blazegraph unterstützt die Mehrmandantenfähigkeit innerhalb einer einzigen Datenbank. In Neptune wird die Multi-Tenancy entweder durch das Speichern von Daten in benannten Graphen und die Verwendung der USING NAMED-Klauseln für SPARQL-Abfragen oder durch die Erstellung eines separaten Datenbank-Clusters für jeden Mandanten unterstützt.
- Federation — Neptune unterstützt derzeit den SPARQL 1.1-Verbund zu Standorten, auf die die Neptune-Instanz zugreifen kann, z. B. innerhalb der privaten VPC VPCs, über oder für externe Internet-Endpunkte. Abhängig von der spezifischen Konfiguration und den erforderlichen Verbundendpunkten benötigen Sie möglicherweise eine zusätzliche Netzwerkkonfiguration.
- Blazegraph-Standarderweiterungen – Blazegraph enthält mehrere Erweiterungen sowohl der SPARQL- als auch der REST-API-Standards, wohingegen Neptune nur mit den Standardspezifikationen selbst kompatibel ist. Dies kann Änderungen an Ihrer Anwendung erfordern oder die Migration erschweren.

AWS Bereitstellungsinfrastruktur für Neptune

Sie können die erforderliche AWS Infrastruktur zwar manuell mithilfe von AWS-Managementkonsole oder erstellen AWS CLI, es ist jedoch häufig praktischer, stattdessen eine CloudFormation Vorlage zu verwenden, wie unten beschrieben:

Bereitstellung von Neptune mithilfe einer Vorlage: CloudFormation

1. Navigieren Sie zu Erstellen eines Amazon Neptune Neptune-Clusters mit AWS CloudFormation.
2. Wählen Sie Stack starten in Ihrer bevorzugten Region.
3. Stellen Sie die erforderlichen Parameter ein (Stackname und `EC2SSHKeyPairName`). Stellen Sie außerdem die folgenden optionalen Parameter ein, um den Migrationsprozess zu vereinfachen:

   - Setzen Sie `AttachBulkloadIAMRoleToNeptuneCluster` auf „true“. Dieser Parameter ermöglicht das Erstellen und Anfügen der entsprechenden IAM-Rolle an Ihren Cluster, um das Massenladen von Daten zu ermöglichen.
   - Stellen Sie `NotebookInstanceType` auf Ihren bevorzugten Instance-Typ ein. Dieser Parameter erstellt ein Neptune-Workbook, mit dem Sie den Massenladevorgang zu Neptune ausführen und die Migration validieren.
4. Wählen Sie Weiter aus.
5. Legen Sie alle anderen gewünschten Stack-Optionen fest.
6. Wählen Sie Weiter aus.

7. Überprüfen Sie Ihre Optionen und aktivieren Sie beide Kontrollkästchen, um zu bestätigen, dass AWS CloudFormation möglicherweise zusätzliche Funktionen erforderlich sind.
8. Wählen Sie Stack erstellen aus.

Der Vorgang der Stack-Erstellung kann einige Minuten dauern.

Exportieren von Daten aus Blazegraph

Der nächste Schritt besteht darin, Daten aus Blazegraph in einem Format zu exportieren, das mit dem Neptune Bulk Loader kompatibel ist.

Je nachdem, wie die Daten in Blazegraph gespeichert sind (Tripel oder Quads) und wie viele benannte Graphen verwendet werden, verlangt Blazegraph möglicherweise, dass Sie den Exportvorgang mehrmals durchführen und mehrere Datendateien generieren:

- Wenn die Daten als Tripel gespeichert sind , müssen Sie für jeden benannten Graphen einen Export ausführen.
- Wenn die Daten als Quads gespeichert sind, können Sie wählen, ob Sie die Daten entweder im N-Quads-Format oder jeden benannten Graphen im Triple-Format exportieren möchten.

Im Folgenden wird davon ausgegangen, dass Sie einen einzelnen Namespace als N-Quads exportieren. Sie können den Vorgang jedoch für weitere Namespaces oder gewünschte Exportformate wiederholen.

Wenn Blazegraph während der Migration online und verfügbar sein soll, verwenden Sie SPARQL CONSTRUCT-Abfragen. Dies erfordert, dass Sie eine Blazegraph-Instance mit einem zugänglichen SPARQL-Endpunkt installieren, konfigurieren und ausführen.

Wenn Blazegraph nicht online sein muss, verwenden Sie das BlazeGraph Exportprogramm. Dazu müssen Sie Blazegraph herunterladen und die Datendatei und die Konfigurationsdateien müssen zugänglich sein, der Server muss jedoch nicht ausgeführt werden.

Exportieren von Daten aus Blazegraph mit SPARQL CONSTRUCT

SPARQL CONSTRUCT ist ein Feature von SPARQL, das einen RDF-Graphen zurückgibt, der der angegebenen Abfragevorlage entspricht. In diesem Anwendungsfall verwenden Sie dies, um Ihre Daten mit einem Namespace nach dem anderen zu exportieren, indem Sie eine Abfrage wie die folgende verwenden:

```
CONSTRUCT WHERE { hint:Query hint:analytic "true" . hint:Query
 hint:constructDistinctSPO "false" . ?s ?p ?o }
```

Es gibt zwar andere RDF-Tools, um diese Daten zu exportieren, aber die einfachste Methode zur Durchführung dieser Abfrage besteht darin, den von Blazegraph bereitgestellten REST-API-Endpunkt zu verwenden. Das folgende Skript demonstriert, wie ein Python-Skript (3.6+) verwendet wird, um Daten als N-Quads zu exportieren:

```
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()
```

Wenn die Daten als Triples gespeichert werden, müssen Sie den `Accept` Header-Parameter ändern, um Daten in einem geeigneten Format (N-Triples, RDF/XML oder Turtle) unter Verwendung der im Blazegraph-Repository angegebenen Werte zu exportieren. GitHub

Verwenden des Blazegraph-Exportprogramms zum Exportieren von Daten

Blazegraph enthält eine Hilfsmethode zum Exportieren von Daten, nämlich die Klasse `ExportKB`. `ExportKB` ermöglicht das Exportieren von Daten aus Blazegraph, erfordert jedoch im Gegensatz zur vorherigen Methode, dass der Server offline ist, während der Export ausgeführt wird. Dadurch ist dies die ideale Methode, wenn Sie Blazegraph während der Migration offline schalten können oder die Migration anhand einer Sicherungskopie der Daten erfolgen kann.

Sie führen das Hilfsprogramm über eine Java-Befehlszeile auf einem Computer aus, auf dem Blazegraph installiert ist, aber nicht ausgeführt wird. Der einfachste Weg, diesen Befehl auszuführen,

besteht darin, die neueste Version von blazegraph.jar herunterzuladen, die sich unter befindet.
GitHub Für die Ausführung dieses Befehls sind mehrere Parameter erforderlich:

- **`log4j.primary.configuration`** – Der Speicherort der log4j-Eigenschaftendatei.
- **`log4j.configuration`** – Der Speicherort der log4j-Eigenschaftendatei.
- **`output`** – Das Ausgabeverzeichnis für die exportierten Daten. Die Dateien befinden sich als `tar.gz` in einem Unterverzeichnis, das so benannt ist wie in der Wissensdatenbank dokumentiert.
- **`format`** – Das gewünschte Ausgabeformat, gefolgt vom Speicherort der `RWStore.properties`-Datei. Wenn Sie mit Tripeln arbeiten, müssen Sie den `-format`-Parameter auf `N-Triples`, `Turtle` oder `RDF/XML` ändern.

Wenn Sie beispielsweise über die Blazegraph-Journaldatei und Eigenschaftendateien verfügen, exportieren Sie Daten mit dem folgenden Code als N-Quads:

```
java -cp blazegraph.jar \
    com.bigdata.rdf.sail.ExportKB \
    -outdir ~/temp/ \
    -format N-Quads \
    ./RWStore.properties
```

Wenn der Export erfolgreich ausgeführt wurde, wird eine Ausgabe wie diese angezeigt:

```
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
```

Erstellen Sie einen Amazon-Simple-Storage-Service- (Amazon S3) Bucket und kopieren Sie die exportierten Daten hinein

Nachdem Sie Ihre Daten aus Blazegraph exportiert haben, erstellen Sie einen Amazon-Simple-Storage-Service- (Amazon S3) Bucket in derselben Region wie der Ziel-Neptune-DB-Cluster, aus dem der Neptune-Bulk-Loader die Daten importieren kann.

Anweisungen zum Erstellen eines Amazon-S3-Buckets finden Sie unter Wie erstelle ich einen S3-Bucket? im Amazon-Simple-Storage-Service-Benutzerhandbuch und unter Beispiele für die Erstellung eines Buckets im Amazon-Simple-Storage-Service-Benutzerhandbuch.

Anweisungen zum Kopieren der Datendateien, die Sie in den neuen Amazon S3 S3-Bucket exportiert haben, finden Sie unter Hochladen eines Objekts in einen Bucket im Amazon Simple Storage Service-Benutzerhandbuch oder Verwenden von High-Level-Befehlen (s3) mit der AWS CLI. Sie können auch Python-Code wie den folgenden verwenden, um die Dateien einzeln zu kopieren:

```
import boto3

region = 'region name'
bucket_name = 'bucket name'
s3 = boto3.resource('s3')
s3.meta.client.upload_file('export.nq', bucket_name, 'export.nq')
```

Verwenden des Neptune Bulk Loader, um die Daten in Neptune zu importieren

Nachdem Sie Ihre Daten aus Blazegraph exportiert und in einen Amazon-S3-Bucket kopiert haben, können Sie die Daten in Neptune importieren. Neptune verfügt über einen Bulk-Loader, der Daten schneller und mit weniger Overhead lädt als Ladevorgänge mit SPARQL. Der Bulk-Loader-Prozess wird durch einen Aufruf der Loader-Endpunkt-API gestartet, um Daten, die im identifizierten S3-Bucket gespeichert sind, in Neptune zu laden.

Sie könnten dies zwar mit einem direkten Aufruf des REST-Endpunkts des Loaders tun, Sie müssen jedoch Zugriff auf die private VPC haben, in der die Neptune-Ziel-Instance ausgeführt wird. Sie könnten einen Bastion-Host einrichten, eine SSH-Verbindung zu diesem Computer herstellen und den cURL-Befehl ausführen, aber die Verwendung der Neptune Workbench ist einfacher.

Neptune Workbench ist ein vorkonfiguriertes Jupyter-Notebook, das als Amazon-Notebook läuft und auf dem mehrere Neptune-spezifische SageMaker Notebook-Magics installiert sind. Diese Magics vereinfachen gängige Neptune-Operationen wie das Überprüfen des Clusterstatus, das Ausführen von SPARQL- und Gremlin-Traversalen und das Ausführen eines Massenladevorgangs.

Um den Massenladevorgang zu starten, verwenden Sie das `%load`-Magic, das eine Schnittstelle bietet, über die Neptune-Loader-Befehl ausgeführt werden kann:

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die Amazon Neptune Neptune-Konsole zu Hausehttps://console.aws.amazon.com/neptune/.
2. Wählen Sie aws-neptune-blazegraph-to -neptune aus.
3. Wählen Sie Notebook öffnen.

4. Wählen Sie in der laufenden Instance von Jupyter entweder ein vorhandenes Notebook aus oder erstellen Sie ein neues mit dem Python-3-Kernel.
5. Öffnen Sie in Ihrem Notebook eine Zelle, geben Sie `%load` ein und führen Sie die Zelle aus.
6. Stellen Sie die Parameter für den Bulk-Loader ein:

   a. Geben Sie unter Quelle den Speicherort einer zu importierenden Quelldatei ein: `s3://{bucket_name}/{file_name}`.

   b. Wählen Sie für Format das entsprechende Format aus, in diesem Beispiel `nquads`.

   c. Geben Sie für ARN laden den ARN für die `IAMBulkLoad`-Rolle ein (diese Information befindet sich in der IAM-Konsole unter Rollen).
7. Wählen Sie Absenden aus.

Das Ergebnis enthält den Status der Anforderung. Bei Massenladevorgängen handelt es sich häufig um Prozesse mit langer Laufzeit. Die Antwort bedeutet also nicht, dass der Ladevorgang abgeschlossen ist, sondern nur, dass er begonnen hat. Diese Statusinformationen werden regelmäßig aktualisiert, bis gemeldet wird, dass der Auftrag abgeschlossen ist.

> **Note**
>
> Diese Informationen sind auch im Blog-Beitrag Der Weg in die Cloud: Migration von Blazegraph zu Amazon Neptune verfügbar.

Laden von Daten in Amazon Neptune

Es gibt verschiedene Möglichkeiten, Diagrammdaten in Amazon Neptune zu laden:

- Wenn Sie nur eine relativ kleine Datenmenge laden müssen, können Sie Abfragen wie `INSERT` SPARQL-Anweisungen oder Gremlin `mergeV` and `mergeE` Steps verwenden. OpenCypher enthält auch UND-Klauseln. `CREATE` `MERGE`
- Sie können Neptune-Massen-Loader dazu nutzen, große Datenmengen aus externen Dateien zu laden. Der Bulk-Loader-Befehl ist schneller und hat weniger Overhead als die Abfragesprache-Befehle. Er ist für große Datensätze optimiert und unterstützt sowohl Resource Description Framework (RDF)-Daten als auch Gremlin-Daten.
- Sie können AWS Database Migration Service (AWS DMS) verwenden, um Daten aus anderen Datenspeichern zu importieren (siehe, und Benutzerhandbuch). Verwenden AWS Database Migration Service um Daten aus einem anderen Datenspeicher in Amazon Neptune zu ladenAWS Database Migration Service
- Für kleinere Datensätze in einer oder mehreren Amazon S3-Dateien können Sie abfragebasierte Ladefunktionen verwenden, um Daten direkt in Ihren Abfragen zu lesen und zu verarbeiten. Weitere Details finden Sie unter Daten mithilfe von Abfragen in Amazon Neptune laden.

Themen

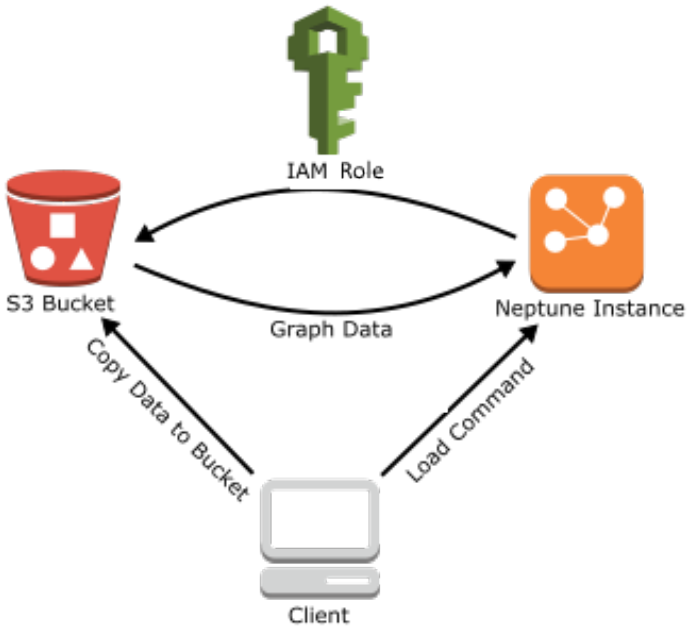
- Verwenden des Amazon Neptune Bulk Loaders zur Datenaufnahme
- Verwenden AWS Database Migration Service um Daten aus einem anderen Datenspeicher in Amazon Neptune zu laden
- Daten mithilfe von Abfragen in Amazon Neptune laden

Verwenden des Amazon Neptune Bulk Loaders zur Datenaufnahme

Amazone Neptune stellt den Befehl `Loader` bereit, um Daten aus externen Dateien direkt in einen Neptune-DB-Cluster zu laden. Sie können diesen Befehl verwenden, statt eine große Anzahl von `INSERT`-Anweisungen, `addV`- und `addE`-Schritten oder anderen API-Aufrufen auszuführen.

Der Neptune-Befehl Loader ist schneller, hat einen geringeren Overhead, ist für große Datensätze optimiert und unterstützt sowohl Gremlin-Daten als auch Resource-Description-Framework (RDF)-Daten, die von SPARQL verwendet werden.

Die folgende Abbildung zeigt eine Übersicht über den Ladevorgang:



Im Folgenden sind die Schritte des Ladevorgangs aufgeführt:

1. Kopieren Sie die Datendateien in einen Amazon-Simple-Storage-Service-Bucket (Amazon-S3-Bucket).
2. Erstellen Sie eine IAM-Rolle mit Lese- und Auflistungszugriff auf den Bucket.
3. Erstellen Sie einen Amazon-S3-VPC-Endpunkt.
4. Starten Sie den Neptune-Loader, indem Sie eine HTTP-Anforderung an die Neptune-DB-Instance senden.
5. Die Neptune-DB-Instance nimmt die IAM-Rolle an, um die Daten aus dem Bucket zu laden.

> **Note**
>
> Sie können verschlüsselte Daten aus Amazon S3 laden, wenn sie entweder mit Amazon S3 `SSE-S3` oder dem `SSE-KMS` Modus verschlüsselt wurden, vorausgesetzt, dass die Rolle, die Sie für das Massenladen verwenden, Zugriff auf das Amazon S3 S3-Objekt hat, und im

Fall von SSE-KMS auch auf`kms:decrypt`. Neptune kann dann Ihre Anmeldeinformationen vortäuschen und `s3:getObject`-Aufrufe in Ihrem Namen ausgeben.
Allerdings unterstützt Neptune zurzeit nicht das Laden von Daten, die mit dem Modus `SSE-C` verschlüsselt wurden.

Die folgenden Abschnitte stellen Anweisungen für die Vorbereitung von Daten und das Laden von Daten in Neptune bereit.

Themen

- Voraussetzungen: IAM-Rolle und Amazon-S3-Zugriff
- Ladedatenformate
- Beispiel: Laden von Daten in eine Neptune-DB-Instance
- Optimieren einer Amazon-Neptune-Massenladung
- Neptune-Loader-Referenz

Voraussetzungen: IAM-Rolle und Amazon-S3-Zugriff

Das Laden von Daten aus einem Amazon Simple Storage Service (Amazon S3) -Bucket erfordert eine AWS Identity and Access Management (IAM) -Rolle, die Zugriff auf den Bucket hat. Amazon Neptune nimmt diese Rolle an, um die Daten zu laden.

Note

Sie können verschlüsselte Daten aus Amazon S3 laden, wenn sie mit dem Amazon-S3-Modus `SSE-S3` verschlüsselt wurden. In diesem Fall kann Neptune Ihre Anmeldeinformationen vortäuschen und `s3:getObject`-Aufrufe in Ihrem Namen ausgeben.
Sie können auch verschlüsselte Daten aus Amazon S3 laden, die mit dem Modus `SSE-KMS` verschlüsselt wurden, wenn Ihre IAM-Rolle die notwendigen Berechtigungen für den Zugriff auf AWS KMS besitzt. Ohne die entsprechenden AWS KMS Berechtigungen schlägt der Massenladevorgang fehl und es wird eine `LOAD_FAILED` Antwort zurückgegeben.
Neptune unterstützt zurzeit nicht das Laden von Amazon-S3-Daten, die mit dem Modus `SSE-C` verschlüsselt wurden.

Die folgenden Abschnitte zeigen, wie Sie mit einer verwalteten IAM-Richtlinie eine IAM-Rolle für den Zugriff auf Amazon-S3-Ressourcen erstellen und die Rolle anschließend Ihrem Neptune-Cluster hinzufügen.

Themen

- Erstellen einer IAM-Rolle, um Amazon Neptune den Zugriff auf Amazon-S3-Ressourcen zu erlauben
- Hinzufügen der IAM-Rolle zu einem Amazon-Neptune-Cluster
- Erstellen des Amazon-S3-VPC-Endpunkts
- Verketten von IAM-Rollen in Amazon Neptune

> **Note**
>
> Diese Anleitung erfordert den Zugriff auf die IAM-Konsole und Berechtigungen für die Verwaltung von IAM-Rollen und -Richtlinien. Weitere Informationen finden Sie unter Berechtigungen für das Arbeiten in der AWS Management Console im IAM-Benutzerhandbuch.
> Die Amazon-Neptune-Konsole erfordert die folgenden IAM-Berechtigungen, um die Rolle dem Neptune-Cluster anfügen zu können:
>
> ```
> iam:GetAccountSummary on resource: *
> iam:ListAccountAliases on resource: *
> iam:PassRole on resource: * with iam:PassedToService restricted to
>  rds.amazonaws.com
> ```

Erstellen einer IAM-Rolle, um Amazon Neptune den Zugriff auf Amazon-S3-Ressourcen zu erlauben

Sie können mit der verwalteten IAM-Richtlinie `AmazonS3ReadOnlyAccess` eine neue IAM-Rolle erstellen, die Amazon Neptune den Zugriff auf Amazon-S3-Ressourcen ermöglicht.

Erstellen einer neuen IAM-Rolle, die Neptune den Zugriff auf Amazon S3 ermöglicht

1. Öffnen Sie unter https://console.aws.amazon.com/iam/ die IAM-Konsole.
2. Wählen Sie im Navigationsbereich Rollen aus.

3. Wählen Sie Create role (Rolle erstellen) aus.
4. Wählen Sie in AWS -Service die Option S3 aus.
5. Wählen Sie Weiter: Berechtigungen aus.
6. Verwenden Sie das Filterfeld, um nach dem Begriff S3 zu filtern, und aktivieren Sie das Kästchen neben AmazonS3ReadOnlyAccess.

> **Note**
>
> Diese Richtlinie erteilt `s3:Get*`- und `s3:List*`-Berechtigungen für alle Buckets.
> Mit später ausgeführten Schritten wird der Zugriff auf die Rolle mithilfe der Vertrauensrichtlinie beschränkt.
> Der Loader benötigt lediglich `s3:Get*`- und `s3:List*`-Berechtigungen für den Bucket, aus dem Sie Daten laden. Daher können Sie diese Berechtigungen auch nach Amazon-S3-Ressource einschränken.
> Wenn Ihr S3-Bucket verschlüsselt ist, müssen Sie `kms:Decrypt`-Berechtigungen hinzufügen.

7. Wählen Sie Weiter: Prüfen aus.
8. Legen Sie in Rollenname einen Namen für Ihre IAM-Rolle fest, zum Beispiel: `NeptuneLoadFromS3`. Sie können auch einen optionalen Wert für Rollenbeschreibung hinzufügen, z. B. „Ermöglicht Neptune den Zugriff auf Amazon-S3-Ressourcen in Ihrem Namen“.
9. Wählen Sie Rolle erstellen aus.
10. Wählen Sie im Navigationsbereich Rollen aus.
11. Geben Sie im Feld Search (Suchen) den Namen der von Ihnen erstellten Rolle ein und wählen Sie die Rolle aus, wenn sie in der Liste erscheint.
12. Klicken Sie auf der Registerkarte Trust Relationships (Vertrauensstellungen) auf Edit Trust Relationship (Vertrauensstellung bearbeiten).
13. Fügen Sie die folgende Vertrauensrichtlinie in das Textfeld ein.

    JSON

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

```
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "rds.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

14. Wählen Sie Update trust policy (Vertrauensrichtlinie aktualisieren) aus.
15. Führen Sie die Schritte unter Hinzufügen der IAM-Rolle zu einem Amazon-Neptune-Cluster aus.

Hinzufügen der IAM-Rolle zu einem Amazon-Neptune-Cluster

Verwenden Sie die Konsole, um die IAM-Rolle zu einem Amazon-Neptune-Cluster hinzuzufügen. So können alle Neptune-DB-Instances im Cluster die Rolle annehmen und Daten aus Amazon S3 laden.

> **Note**
>
> Die Amazon-Neptune-Konsole erfordert die folgenden IAM-Berechtigungen, um die Rolle dem Neptune-Cluster anfügen zu können:
>
> ```
> iam:GetAccountSummary on resource: *
> iam:ListAccountAliases on resource: *
> iam:PassRole on resource: * with iam:PassedToService restricted to
>  rds.amazonaws.com
> ```

Hinzufügen einer IAM-Rolle zu einem Amazon-Neptune-Cluster

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die Amazon Neptune Neptune-Konsole unter. https://console.aws.amazon.com/neptune/home
2. Wählen Sie im Navigationsbereich Datenbanken aus.
3. Wählen Sie die Cluster-ID für den Cluster, den Sie ändern möchten.
4. Wählen Sie die Registerkarte Konnektivität und Sicherheit.
5. Wählen Sie im Abschnitt IAM-Rollen die Rolle aus, die Sie im vorherigen Abschnitt erstellt haben.

6. Wählen Sie Rolle hinzufügen aus.
7. Warten Sie, bis die IAM-Rolle für den Cluster zugänglich ist, bevor Sie sie verwenden.

Erstellen des Amazon-S3-VPC-Endpunkts

Der Neptune-Loader erfordert einen VPC-Endpunkt des Typs Gateway für Amazon S3.

So richten Sie den Zugriff für Amazon S3 ein

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die Amazon VPC-Konsole unter https://console.aws.amazon.com/vpc/.
2. Wählen Sie im Navigationsbereich Endpunkte aus.
3. Klicken Sie auf Endpunkt erstellen.
4. Wählen Sie den Servicenamen `com.amazonaws.`*`region`*`.s3` für den Endpunkt vom Typ Gateway aus.

   > **Note**
   >
   > Wenn die Region hier nicht korrekt ist, stellen Sie sicher, dass die Konsolenregion richtig ist.

5. Wählen Sie die VPC aus, die Ihre Neptune-DB-Instance enthält (sie wird in der Neptune-Konsole für Ihre DB-Instance aufgelistet).
6. Aktivieren Sie das Kontrollkästchen neben den Routing-Tabellen, die den Subnetzen Ihres Clusters zugeordnet sind. Wenn Sie nur eine Routing-Tabelle vorliegen haben, müssen Sie das entsprechende Kontrollkästchen aktivieren.
7. Klicken Sie auf Endpunkt erstellen.

Informationen zum Erstellen von Endpunkten finden Sie unter VPC-Endpunkte im Amazon-VPC-Benutzerhandbuch. Informationen zu den Einschränkungen für VPC-Endpunkte finden Sie unter VPC-Endpunkte für Amazon S3.

Nächste Schritte

Nachdem Sie nun Zugriff auf den Amazon-S3-Bucket gewährt haben, können Sie das Laden von Daten vorbereiten. Informationen zu unterstützten Formaten finden Sie unter Ladedatenformate

Verketten von IAM-Rollen in Amazon Neptune

> **Important**
>
> Das in Engine-Version 1.2.1.0.R3 eingeführte neue kontoübergreifende Massenlade-Feature, das die Vorteile der IAM-Rollen-Verkettung nutzt, kann in einigen Fällen zu einer Verschlechterung der Massenlade-Leistung führen. Daher wurden Upgrades auf Engine-Versionen, die dieses Feature unterstützen, vorübergehend ausgesetzt, bis dieses Problem behoben ist.

Wenn Sie Ihrem Cluster eine Rolle anfügen, kann Ihr Cluster diese Rolle annehmen, um auf in Amazon S3 gespeicherte Daten zuzugreifen. Wenn diese Rolle nicht auf alle benötigten Ressourcen zugreifen kann, können Sie ab Engine-Version 1.2.1.0.R3 eine oder mehrere zusätzliche Rollen verketten, die Ihr Cluster annehmen kann, um Zugriff auf andere Ressourcen zu erhalten. Jede Rolle in der Kette nimmt die nächste Rolle in der Kette an, bis Ihr Cluster die Rolle am Ende der Kette angenommen hat.

Zum Verketten von Rollen richten Sie eine Vertrauensstellung zwischen den Rollen ein. Um beispielsweise `RoleB` mit `RoleA` zu verketten, muss `RoleA` eine Berechtigungsrichtlinie für die Annahme von `RoleB` besitzen. `RoleB` muss eine Vertrauensrichtlinie besitzen, die die Übergabe der Berechtigungen zurück an `RoleA` ermöglicht. Weitere Informationen finden Sie unter Verwenden von IAM-Rollen.

Die erste Rolle in einer Kette muss eine dem Cluster angefügte Rolle sein, die Daten lädt.

Die erste und jede folgende Rolle, die die folgende Rolle in der Kette annimmt, muss Folgendes besitzen:

- Eine Richtlinie, die eine spezifische Aussage zum `Allow`-Effekt auf die Aktion `sts:AssumeRole` enthält.
- Der Amazon-Ressourcenname (ARN) der nächsten Rolle in einem `Resource`-Element

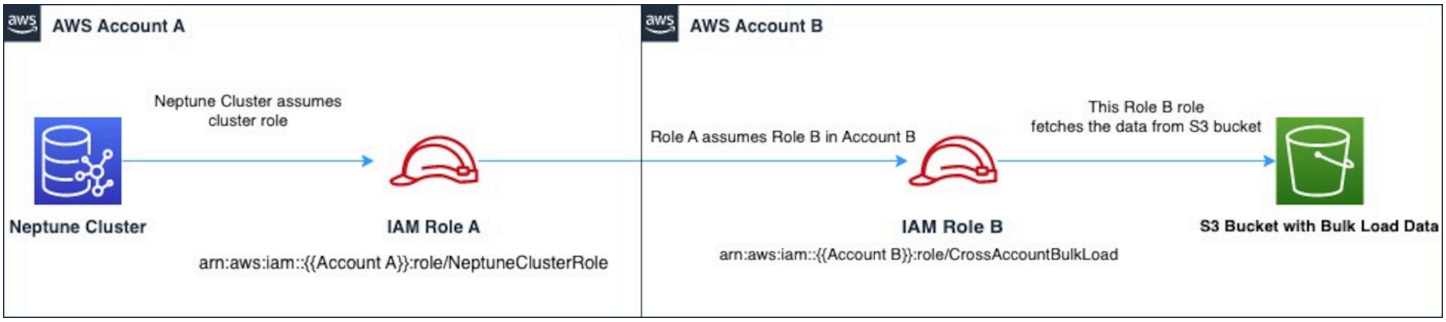
> **Note**
>
> Der Amazon S3 S3-Ziel-Bucket muss sich in derselben AWS Region wie der Cluster befinden.

Cross-account Zugriff mithilfe verketteter Rollen

Sie können einen kontoübergreifenden Zugriff gewähren, indem Sie eine oder mehrere Rollen verketten, die zu einem anderen Konto gehören. Wenn Ihr Cluster vorübergehend eine Rolle annimmt, die zu einem anderen Konto gehört, kann er Zugriff auf die Ressourcen in diesem Konto erhalten.

Angenommen, Konto A möchte auf Daten in einem Amazon-S3-Bucket zugreifen, der zu Konto B gehört:

- Konto A erstellt eine benannte AWS Servicerolle für Neptune `RoleA` und ordnet sie einem Cluster zu.
- Konto B erstellt eine Rolle mit dem Namen `RoleB`, die zum Zugriff auf die Daten in einem Bucket in Konto B berechtigt ist.
- Konto A fügt eine Berechtigungsrichtlinie an `RoleA` an, um `RoleB` anzunehmen.
- Konto B fügt eine Vertrauensrichtlinie an `RoleB` an, um die Rückgabe der Berechtigungen an `RoleA` zu ermöglichen.
- Um auf die Daten im Bucket von Konto B zuzugreifen, führt Konto A einen Loader-Befehl mit dem Parameter `iamRoleArn` aus, der `RoleA` und `RoleB` verkettet. Für die Dauer der Loader-Operation nimmt `RoleA` vorübergehend `RoleB` an, um auf den Amazon-S3-Bucket in Konto B zuzugreifen.



Beispielsweise würde `RoleA` eine Vertrauensrichtlinie besitzen, die eine Vertrauensstellung mit Neptune einrichtet:

JSON

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

```
    {
      "Effect": "Allow",
      "Principal": {
          "Service": "rds.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

`RoleA` würde auch eine Berechtigungsrichtlinie besitzen, die die Annahme von `RoleB` im Besitz von Konto B ermöglicht:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt1487639602000",
            "Effect": "Allow",
            "Action": [
                "sts:AssumeRole"
            ],
            "Resource": "arn:aws:iam::111122223333:role/RoleB"
        }
    ]
}
```

Umgekehrt würde `RoleB` eine Vertrauensrichtlinie für die Einrichtung einer Vertrauensstellung mit `RoleA` besitzen:

JSON

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

```
            "Action": "sts:AssumeRole",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/RoleA"
            }
        }
    ]
}
```

`RoleB` würde auch eine Berechtigung für den Zugriff auf Daten in dem Amazon-S3-Bucket benötigen, der sich in Konto B befindet.

Erstellen eines AWS -Security-Token-Service (STS) VPC-Endpunkt

Der Neptune-Loader benötigt einen VPC-Endpunkt, AWS STS wenn Sie IAM-Rollen verketten, um privat über private IP-Adressen auf AWS STS APIs zuzugreifen. Sie können auf sichere und skalierbare Weise eine direkte Verbindung von einer Amazon-VPC zu AWS STS einem VPC-Endpunkt herstellen. Wenn Sie einen Schnittstellen-VPC-Endpunkt verwenden, wird die Sicherheitslage verbessert, da Sie keine Firewalls für den ausgehenden Datenverkehr öffnen müssen. Sie erhalten außerdem die übrigen Vorteile der Verwendung von Amazon-VPC-Endpunkten.

Bei Verwendung eines VPC-Endpunkts wird der Datenverkehr AWS STS nicht über das Internet übertragen und verlässt niemals das Amazon-Netzwerk. Ihre VPC ist sicher verbunden, AWS STS ohne dass Verfügbarkeitsrisiken oder Bandbreitenbeschränkungen für Ihren Netzwerkverkehr auftreten. Weitere Informationen finden Sie unter Verwenden von AWS STS -Schnittstellen-VPC-Endpunkten.

Um den Zugriff einzurichten für AWS -Security-Token-Service (STS)

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die Amazon VPC-Konsole unter https://console.aws.amazon.com/vpc/.
2. Wählen Sie im Navigationsbereich Endpunkte aus.
3. Klicken Sie auf Endpunkt erstellen.
4. Wählen Sie den Servicenamen aus: `com.amazonaws.region.sts` für den Endpunkt vom Typ Schnittstelle.
5. Wählen Sie die VPC aus, die Ihre Neptune-DB-Instance und EC2-Instance enthält.
6. Aktivieren Sie das Kontrollkästchen neben dem Subnetz, in dem sich Ihre EC2-Instance befindet. Sie können nicht mehrere Subnetze aus derselben Availability Zone auswählen.
7. Wählen Sie für IP address type (IP-Adressentyp) eine der folgenden Optionen aus:

- IPv4 – Weisen Sie Ihren Endpunkt-Netzwerkschnittstellen IPv4-Adressen zu. Diese Option wird nur unterstützt, wenn alle ausgewählten Subnetze über IPv4-Adressbereiche verfügen.
- IPv6 – Weisen Sie Ihren Endpunktnetzwerkschnittstellen IPv6-Adressen zu. Diese Option wird nur unterstützt, wenn es sich bei allen ausgewählten Subnetzen um Subnetze handelt IPv6-only .
- Dualstack – Zuweisen von IPv4- und IPv6-Adressen zu Ihren Endpunktnetzwerkschnittstellen. Diese Option wird nur unterstützt, wenn alle ausgewählten Subnetze sowohl IPv4- als auch IPv6-Adressbereiche aufweisen.

8. Wählen Sie für Sicherheitsgruppen die Sicherheitsgruppen aus, die den Endpunkt-Netzwerkschnittstellen für den VPC-Endpunkt zugeordnet werden sollen. Sie müssten alle Sicherheitsgruppen auswählen, die Ihrer Neptune-DB-Instance und EC2-Instance angefügt sind.
9. Wählen Sie für Policy (Richtlinie) Full access (Vollzugriff), um alle Operationen aller Prinzipale auf allen Ressourcen über den VPC-Endpunkt zuzulassen. Wählen Sie andernfalls Custom (Benutzerdefiniert), um eine VPC-Endpunktrichtlinie anzufügen, die die Berechtigungen steuert, die Prinzipale zum Ausführen von Aktionen für Ressourcen über den VPC-Endpunkt haben. Diese Option ist nur verfügbar, wenn der Service VPC-Endpunktrichtlinien unterstützt. Weitere Informationen finden Sie unter Endpunktrichtlinien.
10. (Optional) Sie fügen ein Tag hinzu, indem Sie Neuen Tag hinzufügen auswählen und Schlüssel und Wert für das Tag eingeben.
11. Wählen Sie Endpunkt erstellen aus.

Informationen zum Erstellen von Endpunkten finden Sie unter VPC-Endpunkte im Amazon-VPC-Benutzerhandbuch. Bitte beachten Sie, dass Amazon-STS-VPC-Endpunkte eine Voraussetzung für die Verkettung von IAM-Rollen sind.

Nachdem Sie nun Zugriff auf den AWS STS Endpunkt gewährt haben, können Sie sich darauf vorbereiten, Daten zu laden. Weitere Informationen zu unterstützten Formaten finden Sie unter Ladedatenformate.

Verkettung von Rollen innerhalb eines Loader-Befehls

Sie können die Rollenverkettung angeben, wenn Sie einen Loader-Befehl ausführen, indem Sie eine durch Komma getrennte Liste von Rollen-ARNs in den Parameter `iamRoleArn` einfügen.

Obwohl Sie in der Regel nur zwei Rollen in einer Kette benötigen, ist es durchaus möglich, drei oder mehr Rollen zu verketten. Dieser Loader-Befehl verkettet beispielsweise drei Rollen:

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"
```

Weitere Informationen finden Sie unter start-loader-job in der Befehlsreferenz. 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.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**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

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"
     }'
```

Ladedatenformate

Die Amazon Neptune–LoadAPI unterstützt das Laden von Daten in verschiedenen Formaten.

Property-graph Formate laden

Daten, die in einem der folgenden Eigenschaftsdiagrammformate geladen wurden, können anschließend mit Gremlin und openCypher abgefragt werden:

- Gremlin-Ladedatenformat (`csv`): CSV-Format (durch Komma getrennte Werte).
- openCypher-Datenladeformat (`opencypher`): CSV-Format (durch Komma getrennte Werte).

RDF-Ladeformate

Zum Laden von Resource-Description-Framework (RDF)-Daten, die Sie mit SPARQL abfragen, können Sie eines der folgenden Standardformate (wie vom World Wide Web Consortium (W3C) definiert) verwenden:

- N-Triples (`ntriples`) aus der Spezifikation unterhttps://www.w3.org/TR/n-triples/.
- N-Quads (`nquads`) aus der Spezifikation unterhttps://www.w3.org/TR/n-quads/.
- RDF/XML (`rdfxml`) aus der Spezifikation unterhttps://www.w3.org/TR/rdf-syntax-grammar/.
- Turtle (`turtle`) aus der Spezifikation unter https://www.w3.org/TR/turtle/.

Beim Laden von Daten muss eine UTF-8 Kodierung verwendet werden

> **Important**
>
> Alle Ladedatendateien müssen formkodiert sein. UTF-8 Wenn eine Datei nicht UTF-8 codiert ist, versucht Neptune sie trotzdem zu laden. UTF-8

Für N-Quads N-triples Daten, die Unicode-Zeichen enthalten, werden \u*xxxxx* Escape-Sequenzen unterstützt. Neptune unterstützt jedoch keine Normalisierung. Wenn ein Wert vorhanden ist, für den eine Normalisierung erforderlich ist, erfolgt keine Byte-für-Byte-Übereinstimmung beim Abfragen. Weitere Informationen zur Normalisierung finden Sie auf der Normalisierungsseite unter. Unicode.org

Wenn Ihre Daten kein unterstütztes Format aufweisen, müssen Sie diese vor dem Laden konvertieren.

Ein Tool zur Konvertierung von GraphML in das Neptune CSV-Format ist im GraphML2CSV-Projekt unter verfügbar. GitHub

Komprimierungsunterstützung für Ladedatendateien

Neptune unterstützt die Komprimierung einzelner Dateien in den Formaten `gzip` oder `bzip2`.

Die komprimierte Datei muss die `.bz2` Erweiterung `.gz` OR haben und es muss sich um eine einzelne Textdatei handeln, die im Format codiert ist. UTF-8 Es können mehrere Dateien geladen werden. Bei jeder Datei muss es sich jedoch um eine getrennte `.gz`-, `.bz2`- oder unkomprimierte Textdatei handeln. Archivdateien mit Erweiterungen wie `.tar`, `.tar.gz` und `.tgz` werden nicht unterstützt.

In den folgenden Abschnitten werden die Formate ausführlich beschrieben.

Themen

- Gremlin-Ladedatenformat
- Ladeformat für openCypher-Daten
- RDF-Formate zum Laden von Daten

Gremlin-Ladedatenformat

Um Apache TinkerPop Gremlin-Daten im CSV-Format zu laden, müssen Sie die Scheitelpunkte und Kanten in separaten Dateien angeben.

Der Loader kann in einem einzigen Ladeauftrag aus mehreren Vertex-Dateien und mehreren Edge-Dateien laden.

Für jeden Ladebefehl muss sich der Satz der Dateien, die geladen werden sollen, im selben Ordner im Amazon-S3-Bucket befinden. Sie geben den Ordnernamen für den Parameter `source` an. Die Dateinamen und -erweiterungen sind nicht wichtig.

Das Amazon-Neptune-CSV-Format entspricht der CSV-Spezifikation RFC 4180. Weitere Informationen finden Sie unter Common Format and MIME Type for CSV Files auf der Internet Engineering Task Force (IETF)-Website.

> **Note**
>
> Alle Dateien müssen im Format codiert sein. UTF-8

Jede Datei verfügt über eine durch Komma getrennte Kopfzeile. Die Kopfzeile besteht aus System- und Eigenschaftenspalten-Headern.

Systemspalten-Header

Die erforderlichen und zulässigen Systemspalten-Header sind für Knoten- und Kantendateien unterschiedlich.

Jede Systemspalte kann nur einmal in einem Header enthalten sein.

Bei den Bezeichnungen muss die Groß- und Kleinschreibung beachtet werden.

Knoten-Header

- `~id` – Erforderlich

  Eine ID für den Knoten.
- `~label`

  Eine Bezeichnung für den Knoten. Es sind mehrere Bezeichnungswerte zulässig, getrennt durch Semikolon (`;`).

  Falls nicht vorhanden, `~label` wird eine TinkerPop Bezeichnung mit dem Wert bereitgestelltvertex, da jeder Scheitelpunkt mindestens eine Bezeichnung haben muss.

Kanten-Header

- `~id` – Erforderlich

  Eine ID für die Kante.
- `~from` – Erforderlich

  Die Knoten-ID des von-Knotens.
- `~to` – Erforderlich

  Die Knoten-ID des nach-Knotens.
- `~label`

  Eine Bezeichnung für die Kante. Kanten können immer nur eine Bezeichnung enthalten.

  Wenn nicht vorhanden, `~label` wird TinkerPop eine Bezeichnung mit dem Wert bereitgestelltedge, da jede Kante eine Bezeichnung haben muss.

Eigenschaftenspalten-Header

Sie können eine Spalte (`:`) für eine Eigenschaft angeben, indem Sie die folgende Syntax verwenden. Bei den Typnamen muss die Groß- und Kleinschreibung nicht berücksichtigt werden. Ein Doppelpunkt muss mit einem vorangestellten umgekehrten Schrägstrich maskiert werden, wenn er im Namen einer Eigenschaft erscheint: `\:`.

```
propertyname:type
```

> **Note**
>
> Leerzeichen, Kommas, Zeilenumbrüche und Zeilenumbrüche sind in den Spaltenüberschriften nicht zulässig, sodass Eigenschaftsnamen diese Zeichen nicht enthalten dürfen.

Sie können eine Spalte für einen Array-Typ festlegen, indem Sie dem Typ `[]` hinzufügen:

```
propertyname:type[]
```

> **Note**
>
> Edge-Eigenschaften können nur einen einzelnen Wert haben und verursachen einen Fehler, wenn ein Array-Typ oder ein zweiter Wert angegeben ist.

Das folgende Beispiel zeigt den Spalten-Header für eine Eigenschaft mit dem Namen `age` des Typs `Int`.

```
age:Int
```

Für jede Zeile in der Datei ist eine Ganzzahl in dieser Position erforderlich oder sie muss leer sein.

Zeichenfolgen-Arrays sind zulässig. Zeichenfolgen in einem Array dürfen jedoch kein Semikolon (`;`) enthalten, es sei denn, es wird mit einem umgekehrten Schrägstrich maskiert (wie hier: `\;`).

Angeben der Kardinalität einer Spalte

Die Spaltenüberschrift kann verwendet werden, um die Kardinalität für die durch die Spalte identifizierte Eigenschaft anzugeben. Auf diese Weise kann der Massen-Loader die Kardinalität ähnlich wie Gremlin-Abfragen berücksichtigen.

Sie geben die Kardinalität einer Spalte wie folgt an:

```
propertyname:type(cardinality)
```

Der *`cardinality`* Wert kann entweder oder sein. `single` `set` Als Standardwert wird `set` angenommen, Dies bedeutet, dass die Spalte mehrere Werte akzeptieren kann. Bei Edge-Dateien

ist die Kardinalität immer einzeln und die Angabe einer anderen Kardinalität bewirkt, dass der Loader eine Ausnahme auslöst.

Wenn die Kardinalität `single` lautet, gibt der Loader einen Fehler aus, wenn beim Laden eines Wertes bereits ein vorheriger Wert vorhanden ist oder wenn mehrere Werte geladen werden. Dieses Verhalten kann überschrieben werden, sodass ein vorhandener Wert ersetzt wird, wenn ein neuer Wert unter Verwendung des Flags `updateSingleCardinalityProperties` geladen wird. Siehe Loader-Befehl.

Es ist möglich, die Kardinalitätseinstellung mit einem Array-Typ zu vewenden, obwohl dies im Allgemeinen nicht erforderlich ist. Dies sind die möglichen Kombinationen:

- `name:type` – Die Kardinalität ist `set` und der Inhalt ist einwertig.
- `name:type[]` – Die Kardinalität ist `set` und der Inhalt ist mehrwertig.
- `name:type(single)` – Die Kardinalität ist `single` und der Inhalt ist einwertig.
- `name:type(set)` – Die Kardinalität ist `set`, was dem Standardwert entspricht, und der Inhalt ist einwertig.
- `name:type(set)[]` – Die Kardinalität ist `set` und der Inhalt ist mehrwertig.
- `name:type(single)[]` – Dies ist widersprüchlich und führt zu einem Fehler.

Im folgenden Abschnitt werden alle verfügbaren Gremlin-Datentypen aufgeführt.

Gremlin-Datentypen

Dies ist eine Liste der zulässigen Eigenschaftstypen mit einer Beschreibung des jeweiligen Typs.

Bool (oder Boolesch)

Gibt ein boolesches Feld an. Zulässige Werte: `false`, `true`

> **Note**
>
> Jeder andere Wert als `true` wird als „false“ behandelt.

Ganzzahltypen

Werte außerhalb der definierten Bereiche verursachen einen Fehler.

Typ	Bereich
Byte	-128 bis +127
Short	-32768 bis +32767
Int	-2^31 bis 2^31-1
Long	-2^63 bis 2^63-1

Dezimalzahltypen

Unterstützt sowohl die Dezimalschreibweise als auch die wissenschaftliche Notation. Außerdem können Symbole wie (+/-), Infinity oder NaN verwendet werden. INF wird nicht unterstützt.

Typ	Bereich
Gleitkommazahl	32-Bit IEEE 754-Gleitkommawert
Double	64-Bit IEEE 754-Gleitkommawert

Zu lange Gleitkommazahlen und Double-Werte werden geladen und auf den nächsten Wert für 24-Bit- (Gleitkommazahl) und 53-Bit-Genauigkeit (Double) gerundet. Ein in der Mitte liegender Wert wird für die letzte verbleibende Stelle auf Bit-Ebene auf 0 gerundet.

Zeichenfolge

Anführungszeichen sind optional. Kommas, Zeilenumbruch- und Wagenrücklaufzeichen werden automatisch mit Escape-Zeichen markiert, wenn sie in einer Zeichenfolge enthalten sind, die von doppelten Anführungszeichen (") umschlossen ist. Beispiel: `"Hello, World"`

Um Anführungszeichen in einer in Anführungszeichen gesetzte Zeichenfolge aufzunehmen, können Sie das Anführungszeichen mit Escape-Zeichen markieren, indem Sie zwei in einer Zeile verwenden: Beispiel: `"Hello ""World"""`

Zeichenfolgen-Arrays sind zulässig. Zeichenfolgen in einem Array dürfen jedoch kein Semikolon (`;`) enthalten, es sei denn, es wird mit einem umgekehrten Schrägstrich maskiert (wie hier: `\;`).

Wenn Sie Zeichenfolgen in einem Array in Anführungszeichen setzen möchten, müssen Sie das gesamte Array mit einem Satz Anführungszeichen versehen. Beispiel: `"String one; String 2; String 3"`

Date

Java-Datum im ISO-8601 Format. Unterstützt die folgenden Formate:`yyyy-MM-dd`,`yyyy-MM-ddTHH:mm`,`yyyy-MM-ddTHH:mm:ss`,`yyyy-MM-ddTHH:mm:ssZ`. Die Werte werden in Epochenzeit umgewandelt und gespeichert.

DateTime

Java-Datum im ISO-8601 Format. Unterstützt die folgenden Formate:`yyyy-MM-dd`,`yyyy-MM-ddTHH:mm`,`yyyy-MM-ddTHH:mm:ss`,`yyyy-MM-ddTHH:mm:ssZ`. Die Werte werden in Epochenzeit umgewandelt und gespeichert.

Gremlin-Zeilenformat

Trennzeichen

Felder in einer Zeile werden durch ein Komma getrennt. Die Datensätze werden durch einen Zeilenumbruch oder einen Zeilenumbruch gefolgt von einem Wagenrücklauf getrennt.

Leere Felder

Leere Felder sind für nicht erforderliche Spalten zulässig (z. B. benutzerdefinierte Eigenschaften). Ein leeres Feld erfordert dennoch ein Komma als Trennzeichen. Leere Felder in erforderlichen Spalten führen zu einem Analysefehler. Leere Zeichenkettenwerte werden als leerer Zeichenkettenwert für das Feld interpretiert, nicht als leeres Feld. Das Beispiel im nächsten Abschnitt enthält in jedem Beispielknoten ein leeres Feld.

Knoten-IDs

`~id`-Werte müssen für alle Knoten in allen Knotendateien eindeutig sein. Mehrere Knotenzeilen mit identischen `~id`-Werten werden auf einen einzigen Knoten im Graph angewendet. Eine leere Zeichenfolge (`""`) ist eine gültige ID, und der Scheitelpunkt wird mit einer leeren Zeichenfolge als ID erstellt.

Kanten-IDs

Darüber hinaus müssen `~id`-Werte für alle Kanten in allen Kantendateien eindeutig sein. Mehrere Kantenzeilen mit identischen `~id`-Werten werden auf die einzige Kante im Graph angewendet. Eine

leere Zeichenfolge (`""`) ist eine gültige ID, und die Kante wird mit einer leeren Zeichenfolge als ID erstellt.

Labels

Bei Labels wird zwischen Groß- und Kleinschreibung unterschieden und sie dürfen nicht leer sein. Ein Wert von `""` führt zu einem Fehler.

Zeichenfolgenwerte

Anführungszeichen sind optional. Kommas, Zeilenumbruch- und Wagenrücklaufzeichen werden automatisch mit Escape-Zeichen markiert, wenn sie in einer Zeichenfolge enthalten sind, die von doppelten Anführungszeichen (`"`) umschlossen ist. Leere Zeichenkettenwerte (`""`) werden als leerer Zeichenkettenwert für das Feld interpretiert, nicht als leeres Feld.

CSV-Formatspezifikation

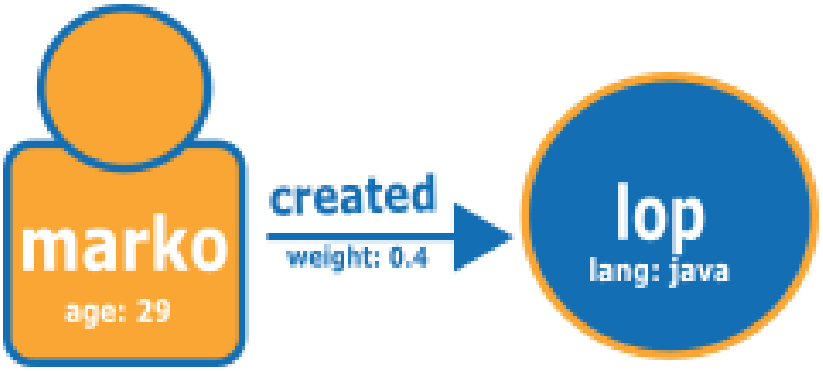
Das Neptune-CSV-Format entspricht der CSV-Spezifikation RFC 4180, einschließlich der folgenden Anforderungen.

- Es werden sowohl Unix- als auch Windows-Zeilenenden unterstützt (\n oder \r\n).
- Jedes Feld kann mit (doppelten) Anführungszeichen versehen werden.
- Felder, die einen Zeilenumbruch, doppelte Anführungszeichen oder Kommas enthalten, müssen in Anführungszeichen gesetzt werden. (Andernfalls wird der Ladevorgang sofort abgebrochen.)
- Eine doppeltes Anführungszeichen (`"`) in einem Feld muss durch zwei (doppelte) Anführungszeichen dargestellt werden. Beispiel: Eine Zeichenfolge `Hello "World"` muss als `"Hello ""World"""` in den Daten dargestellt werden.
- Umgebende Leerzeichen zwischen Trennzeichen werden ignoriert. Wenn eine Zeile als vorhanden ist`value1, value2`, werden sie als `"value1"` und gespeichert`"value2"`.
- Alle anderen Escapezeichen werden unverändert gespeichert. Zum Beispiel wird `"data1\tdata2"` als `"data1\tdata2"` gespeichert. Es sind keine weiteren Escapezeichen erforderlich, solange diese Zeichen in Anführungszeichen gesetzt sind.
- Leere Felder sind zulässig. Ein leeres Feld wird als leerer Wert interpretiert.
- Mehrere Werte für ein Feld werden mit einem Semikolon (`;`) zwischen den Werten angegeben.

Weitere Informationen finden Sie unter Common Format and MIME Type for CSV Files auf der Internet Engineering Task Force (IETF)-Website.

Gremlin-Beispiel

Das folgende Diagramm zeigt ein Beispiel für zwei Scheitelpunkte und eine Kante aus dem TinkerPop Modern Graph.



Die folgende Abbildung zeigt das Diagramm im Neptune-CSV-Ladeformat.

Knotendatei:

```
~id,name:String,age:Int,lang:String,interests:String[],~label
v1,"marko",29,,"sailing;graphs",person
v2,"lop",,"java",,software
```

Tabellarische Ansicht der Knotendatei:

~id	name:String	age:Int	lang:String	Interessen: Zeichenfolge []	~label
v1	"marko"	29		["Segeln“, „Graphen"]	Person
v2	"lop"		"java"		software

Kantendatei:

```
~id,~from,~to,~label,weight:Double
e1,v1,v2,created,0.4
```

Tabellarische Ansicht der Kantendatei:

~id	~from	~auf	~label	weight:Double
e1	v1	v2	created	0.4

Nächste Schritte

Da Sie jetzt über ausführlichere Kenntnisse von Ladeformaten verfügen, fahren Sie mit Beispiel: Laden von Daten in eine Neptune-DB-Instance fort.

Ladeformat für openCypher-Daten

Um openCypher-Daten im openCypher-CSV-Format zu laden, müssen Sie Knoten und Beziehungen in getrennten Dateien angeben. Der Loader kann Daten aus mehreren Knoten- und Beziehungsdateien in einem einzigen Ladeauftrag laden.

Für jeden Ladebefehl muss der Satz von Dateien, die geladen werden sollen, dasselbe Pfadpräfix in einem Amazon-Simple-Storage-Service-Bucket aufweisen. Sie geben dieses Präfix im Quellparameter an. Die tatsächlichen Dateinamen und Dateierweiterungen sind nicht wichtig.

In Amazon Neptune entspricht das openCypher-CSV-Format der CSV-Spezifikation RFC 4180. Weitere Informationen finden Sie unter Gemeinsames Format und MIME-Typ für CSV-Dateien (https://tools.ietf.org/html/rfc4180) auf der Website der Internet Engineering Task Force (IETF).

> **Note**
>
> Diese Dateien MÜSSEN im UTF-8 Format codiert sein.

Jede Datei hat eine durch Komma getrennte Überschriftenzeile, die Systemspaltenüberschriften und Eigenschaftsspaltenüberschriften enthält.

Systemspaltenüberschriften in openCypher-Dateien zum Laden von Daten

Jede Systemspalte kann nur einmal in einer Überschrift enthalten sein. Bei allen Systemspaltenüberschriften muss die Groß- und Kleinschreibung beachtet werden.

openCypher-Knotenladedateien und Beziehungsladedateien unterscheiden sich hinsichtlich der erforderlichen und zulässigen Systemspaltenüberschriften:

Systemspaltenüberschriften in Knotendateien

- **`:ID`** – (Erforderlich) Eine ID für den Knoten.

  Der Knoten–`:ID`Spaltenüberschrift kann ein optionaler ID-Bereich wie folgt hinzugefügt werden: `:ID(`*`ID Space`*`)`. Ein Beispiel ist `:ID(movies)`.

  Verwenden Sie beim Laden von Beziehungen, die die Knoten in dieser Datei verbinden, dieselben ID-Leerzeichen in den Spalten der Beziehungsdateien. `:START_ID` and/or `:END_ID`

  Die Knoten-`:ID`-Spalte kann optional als Eigenschaft im Formular gespeichert werden, *`property name`*`:ID`. Ein Beispiel ist `name:ID`.

  Knoten-IDs sollten für alle Knotendateien in den aktuellen und vorherigen Ladevorgängen eindeutig sein. Bei Verwendung eines ID-Bereichs sollten die Knoten-IDs für alle Knotendateien eindeutig sein, die in aktuellen und vorherigen Ladevorgängen denselben ID-Bereich verwenden.
- **`:LABEL`** – Eine Bezeichnung für den Knoten.

  Wenn Sie mehrere Labelwerte für einen einzelnen Knoten verwenden, sollte jedes Label durch Semikolons () getrennt werden. `;`

Systemspaltenüberschriften in Beziehungsdateien

- **`:ID`** – Eine ID für die Beziehung. Dies ist erforderlich, wenn `userProvidedEdgeIds` wahr ist (Standard), jedoch ungültig, wenn `userProvidedEdgeIds` `false` ist.

  Beziehungs-IDs sollten für alle Beziehungsdateien in den aktuellen und vorherigen Ladevorgängen eindeutig sein.
- **`:START_ID`** – (Erforderlich) Die Knoten-ID des Knotens, an dem diese Beziehung beginnt.

  Optional kann der Start-ID-Spalte ein ID-Bereich im Format `:START_ID(`*`ID Space`*`)` zugeordnet werden. Der ID-Bereich der Startknoten-ID sollte mit dem ID-Bereich übereinstimmen, der dem Knoten in dessen Knotendatei zugewiesen ist.
- **`:END_ID`** – (Erforderlich) Die Knoten-ID des Knotens, an dem diese Beziehung endet.

  Optional kann der End-ID-Spalte ein ID-Bereich im Format `:END_ID(`*`ID Space`*`)` zugeordnet werden. Der ID-Bereich der Endknoten-ID sollte mit dem ID-Bereich übereinstimmen, der dem Knoten in dessen Knotendatei zugewiesen ist.
- **`:TYPE`** – Ein Typ für die Beziehung. Beziehungen können nur einen einzigen Typ haben.

> **Note**
>
> Informationen zur Behandlung duplizierter Knoten- oder Beziehungs-IDs beim Massenladen finden Sie unter Laden von openCypher-Daten.

Eigenschaftsspaltenüberschriften in openCypher-Dateien zum Laden von Daten

Sie können mit einer Eigenschaftsspaltenüberschrift angeben, dass eine Spalte die Werte für eine bestimmte Eigenschaft enthält. Die Überschrift muss das folgende Format haben:

```
propertyname:type
```

Leerzeichen, Kommas, Zeilenumbrüche und Zeilenumbrüche sind in den Spaltenüberschriften nicht zulässig, sodass Eigenschaftsnamen diese Zeichen nicht enthalten dürfen. Dies ist ein Beispiel für die Spaltenüberschrift einer Eigenschaft mit dem Namen `age` und dem Typ `Int`.

```
age:Int
```

Die Spalte mit `age:Int` als Spaltenüberschrift müsste dann in jeder Zeile eine Ganzzahl oder einen leeren Wert enthalten.

Datentypen in Neptune openCypher-Dateien zum Laden von Daten

- **`Bool`** oder **`Boolean`** – Ein boolesches Feld. Zulässige Werte sind `true` und `false`.

  Jeder andere Wert als `true` wird als `false` behandelt.
- **`Byte`** – Eine ganze Zahl im Bereich von `-128` bis `127`.
- **`Short`** – Eine ganze Zahl im Bereich von `-32,768` bis `32,767`.
- **`Int`** – Eine ganze Zahl im Bereich von `-2^31` bis `2^31 - 1`.
- **`Long`** – Eine ganze Zahl im Bereich von `-2^63` bis `2^63 - 1`.
- **`Float`** – Eine 32-Bit-Gleitkommazahl nach IEEE 754. Dezimalschreibweise und wissenschaftliche Notation werden unterstützt. `Infinity`, `-Infinity` und `NaN` werden erkannt, `INF` jedoch nicht.

  Werte mit zu vielen Stellen werden auf den nächsten Wert gerundet. (Ein in der Mitte liegender Wert Wert wird für die letzte Stelle auf Bit-Ebene auf 0 gerundet.)

- **`Double`** – Eine 64-Bit-Gleitkommazahl nach IEEE 754. Dezimalschreibweise und wissenschaftliche Notation werden unterstützt. `Infinity`, `-Infinity` und `NaN` werden erkannt, `INF` jedoch nicht.

  Werte mit zu vielen Stellen werden auf den nächsten Wert gerundet. (Ein in der Mitte liegender Wert Wert wird für die letzte Stelle auf Bit-Ebene auf 0 gerundet.)

- **`String`** – Anführungszeichen sind optional. Kommas, Zeilenumbruchzeichen und Zeilenumschaltzeichen werden automatisch mit Escape-Zeichen markiert, wenn sie in einer von doppelten Anführungszeichen (") umschlossenen Zeichenfolge enthalten sind, z. B. `"Hello, World"`.

  Sie können Anführungszeichen in einer Zeichenfolge mit Anführungszeichen verwenden, indem Sie zwei in einer Zeile verwenden, z. B. `"Hello ""World"""`.

- **`DateTime`**— Ein Java-Datum in einem der folgenden Formate: ISO-8601
  - `yyyy-MM-dd`
  - `yyyy-MM-ddTHH:mm`
  - `yyyy-MM-ddTHH:mm:ss`
  - `yyyy-MM-ddTHH:mm:ssZ`

Auto-cast Datentypen in Neptune OpenCypher Daten laden Dateien

Auto-cast Datentypen werden bereitgestellt, um Datentypen zu laden, die derzeit nicht von Neptune nativ unterstützt werden. Daten in solchen Spalten werden in unveränderter Form als Zeichenfolgen gespeichert, ohne sie anhand des beabsichtigten Formats zu verifizieren. Die folgenden Auto-Cast-Datentypen sind zulässig:

- **`Char`** – Ein `Char`-Feld. Als Zeichenfolge gespeichert.
- **`Date`**, **`LocalDate`** und **`LocalDateTime`** – Siehe Zeitliche Neo4j-Instantwerte für eine Beschreibung der Typen `date`, `localdate` und `localdatetime`. Die Werte werden in unveränderter Form als Zeichenfolgen ohne Validierung geladen.
- **`Duration`** – Siehe das Neo4j-Dauerformat. Die Werte werden in unveränderter Form als Zeichenfolgen ohne Validierung geladen.
- Punkt – Ein Punktfeld zum Speichern räumlicher Daten. Siehe Räumliche Instantwerte. Die Werte werden in unveränderter Form als Zeichenfolgen ohne Validierung geladen.

Beispiel für das openCypher-Ladeformat

Das folgende Diagramm aus dem TinkerPop Modern Graph zeigt ein Beispiel für zwei Knoten und eine Beziehung:



Die folgende Abbildung zeigt das Diagramm im normalen Neptune-openCypher-Ladeformat.

Knotendatei:

```
:ID,name:String,age:Int,lang:String,:LABEL
v1,"marko",29,,person
v2,"lop",,"java",software
```

Beziehungsdatei:

```
:ID,:START_ID,:END_ID,:TYPE,weight:Double
e1,v1,v2,created,0.4
```

Alternativ könnten Sie ID-Bereiche und ID wie folgt als Eigenschaft verwenden:

Erste Knotendatei:

```
name:ID(person),age:Int,lang:String,:LABEL
"marko",29,,person
```

Zweite Knotendatei:

```
name:ID(software),age:Int,lang:String,:LABEL
"lop",,"java",software
```

Beziehungsdatei:

```
:ID,:START_ID(person),:END_ID(software),:TYPE,weight:Double
e1,"marko","lop",created,0.4
```

RDF-Formate zum Laden von Daten

Zum Laden von Resource Description Framework (RDF)-Daten können Sie eines der folgenden Standardformate wie vom World Wide Web Consortium (W3C) angegeben verwenden:

- N-Triples (`ntriples`) aus der Spezifikation unter https://www.w3.org/TR/n-triples/
- N-Quads (`nquads`) aus der Spezifikation unter https://www.w3.org/TR/n-quads/
- RDF/XML (`rdfxml`) aus der Spezifikation unter https://www.w3.org/TR/rdf-syntax-grammar/
- Turtle (`turtle`) aus der Spezifikation unter https://www.w3.org/TR/turtle/

> **Important**
>
> Alle Dateien müssen im UTF-8 Format kodiert sein.
> Für N-Quads N-triples Daten, die Unicode-Zeichen enthalten, werden \u*xxxxx* Escape-Sequenzen unterstützt. Neptune unterstützt jedoch keine Normalisierung. Wenn ein Wert vorhanden ist, für den eine Normalisierung erforderlich ist, erfolgt keine Byte-für-Byte-Übereinstimmung beim Abfragen. Weitere Informationen zur Normalisierung finden Sie auf der Normalisierungsseite unter. Unicode.org

Nächste Schritte

Da Sie jetzt über ausführlichere Kenntnisse von Ladeformaten verfügen, fahren Sie mit Beispiel: Laden von Daten in eine Neptune-DB-Instance fort.

Beispiel: Laden von Daten in eine Neptune-DB-Instance

Dieses Beispiel zeigt, wie Daten in Amazon Neptune geladen werden. Wenn nicht anders angegeben, müssen Sie diese Schritte über eine Amazon-Elastic-Compute-Cloud-Instance (Amazon-EC2-Instance) in derselben Amazon Virtual Private Cloud (VPC) ausführen, in der sich auch Ihre Neptune-DB-Instance befindet.

Beispiel für die Voraussetzungen für das Laden von Daten

Sie benötigen Folgendes, um starten zu können:

- Eine Neptune-DB-Instance.

  Informationen zum Starten einer Neptune-DB-Instance finden Sie unter Einen Amazon Neptune Neptune-Cluster erstellen.

- Einen Amazon-Simple-Storage-Service-Bucket (Amazon-S3-Bucket) zum Speichern der Datendateien.

  Sie können einen vorhandenen Bucket verwenden. Wenn Sie nicht über einen S3-Bucket verfügen, finden Sie weitere Informationen unter Einen Bucket erstellen im Amazon-S3-Handbuch für die ersten Schritte.

- Die Diagrammdaten, die geladen werden sollen, in einem Format, das vom Neptune-Loader unterstützt wird:

  Wenn Sie Gremlin zum Abfragen Ihres Diagramms verwenden, kann Neptune Daten in einem Format mit durch Komma getrennten Werten (CSV) laden, wie in Gremlin-Ladedatenformat beschrieben.

  Wenn Sie OpenCypher verwenden, um Ihr Diagramm abzufragen, kann Neptune auch Daten in einem offenen Cypher-specific CSV Format laden, wie unter beschrieben. Ladeformat für openCypher-Daten

  Wenn Sie SPARQL verwenden, kann Neptune die Daten in einer Reihe von RDF-Formaten laden, wie in RDF-Formate zum Laden von Daten beschrieben.

- Eine IAM-Rolle zur Übernahme durch die Neptune-DB-Instance, die über eine IAM-Richtlinie verfügt, die den Zugriff auf die Datendateien im S3-Bucket erlaubt. Die Richtlinie muss Lese- und Auflistungsberechtigungen erteilen.

  Informationen zum Erstellen einer Rolle, die über Zugriff auf Amazon S3 verfügt, und zum Verknüpfen dieser Rolle mit einem Neptune-Cluster finden Sie unter Voraussetzungen: IAM-Rolle und Amazon-S3-Zugriff.

  > **Note**
  >
  > Die Neptune-Load-API benötigt nur Lesezugriff auf die Datendateien. Die IAM-Richtlinie muss keinen Schreibzugriff oder Zugriff auf den gesamten Bucket zulassen.

- Einen Amazon-S3-VPC-Endpunkt. Weitere Informationen finden Sie im Abschnitt Erstellen eines Amazon-S3-VPC-Endpunkts.

Erstellen eines Amazon-S3-VPC-Endpunkts

Der Neptune Loader erfordert einen VPC-Endpunkt für Amazon S3.

So richten Sie den Zugriff für Amazon S3 ein

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die Amazon VPC-Konsole unter https://console.aws.amazon.com/vpc/.
2. Wählen Sie im linken Navigationsbereich die Option Endpoints (Endpunkte) aus.
3. Klicken Sie auf Endpunkt erstellen.
4. Wählen Sie unter Service Name (Service-Name) `com.amazonaws.region.s3` aus.

   > **Note**
   >
   > Wenn die Region hier nicht korrekt ist, stellen Sie sicher, dass die Konsolenregion richtig ist.

5. Wählen Sie die VPC aus, die Ihre Neptune-DB-Instance enthält.
6. Aktivieren Sie das Kontrollkästchen neben den Routing-Tabellen, die den Subnetzen Ihres Clusters zugeordnet sind. Wenn Sie nur eine Routing-Tabelle vorliegen haben, müssen Sie das entsprechende Kontrollkästchen aktivieren.
7. Klicken Sie auf Endpunkt erstellen.

Informationen zum Erstellen von Endpunkten finden Sie unter VPC-Endpunkte im Amazon-VPC-Benutzerhandbuch. Informationen zu den Einschränkungen für VPC-Endpunkte finden Sie unter VPC-Endpunkte für Amazon S3.

Laden von Daten in eine Neptune-DB-Instance

1. Kopieren Sie die Datendateien in einen Amazon-S3-Bucket. Der S3-Bucket muss sich in derselben AWS Region befinden wie der Cluster, der die Daten lädt.

   Sie können den folgenden AWS CLI Befehl verwenden, um die Dateien in den Bucket zu kopieren.

> **Note**
>
> Dieser Befehl muss nicht über die Amazon-EC2-Instance ausgeführt werden.

```
aws s3 cp data-file-name s3://bucket-name/object-key-name
```

> **Note**
>
> In Amazon S3 stellt ein Objektschlüsselname den gesamten Pfad einer Datei dar, einschließlich des Dateinamens.
> Beispiel: Im Befehl `aws s3 cp datafile.txt s3://examplebucket/mydirectory/datafile.txt` lautet der Objektschlüsselname **`mydirectory/datafile.txt`**.

Alternativ können Sie den verwenden, AWS-Managementkonsole um Dateien in den S3-Bucket hochzuladen. Öffnen Sie die Amazon S3 S3-Konsole unter https://console.aws.amazon.com/s3/und wählen Sie einen Bucket aus. Wählen Sie oben links Hochladen aus, um Dateien hochzuladen.

2. Geben Sie in einem Befehlszeilenfenster Folgendes ein, um den Neptune-Loader auszuführen. Verwenden Sie dabei die korrekten Werte für den Endpunkt, den Amazon-S3-Pfad, das Format und den IAM-Rollen-ARN.

   Der Parameter `format` kann einen der folgenden Werte haben: `csv` für Gremlin, `opencypher` für openCypher oder `ntriples`, `nquads`, `turtle` und `rdfxml` für RDF. Informationen zu den anderen Parametern finden Sie unter Neptune-Loader-Befehl.

   Informationen zum Ermitteln des Hostnamens Ihrer Neptune-DB-Instance finden Sie im Abschnitt Verbinden mit Amazo-Neptune-Endpunkten.

   Der Parameter für die Region muss mit der Region des Clusters und des S3-Buckets übereinstimmen.

   Amazon Neptune ist in den folgenden AWS Regionen verfügbar:

   - USA Ost (Nord-Virginia): `us-east-1`

- USA Ost (Ohio): `us-east-2`
- USA West (Nordkalifornien): `us-west-1`
- USA West (Oregon): `us-west-2`
- Kanada (Zentral): `ca-central-1`
- Kanada West (Calgary): `ca-west-1`
- Südamerika (São Paulo): `sa-east-1`
- Europa (Stockholm): `eu-north-1`
- Europa (Spanien): `eu-south-2`
- Europa (Irland): `eu-west-1`
- Europa (London): `eu-west-2`
- Europa (Paris): `eu-west-3`
- Europa (Frankfurt): `eu-central-1`
- Naher Osten (Bahrain): `me-south-1`
- Naher Osten (VAE): `me-central-1`
- Israel (Tel Aviv): `il-central-1`
- Afrika (Kapstadt): `af-south-1`
- Asien-Pazifik (Hongkong): `ap-east-1`
- Asien-Pazifik (Tokio): `ap-northeast-1`
- Asien-Pazifik (Seoul): `ap-northeast-2`
- Asien-Pazifik (Osaka): `ap-northeast-3`
- Asien-Pazifik (Singapur): `ap-southeast-1`
- Asien-Pazifik (Sydney): `ap-southeast-2`
- Asien-Pazifik (Jakarta): `ap-southeast-3`
- Asien-Pazifik (Melbourne): `ap-southeast-4`
- Asien-Pazifik (Malaysia): `ap-southeast-5`
- Asien-Pazifik (Mumbai): `ap-south-1`
- Asien-Pazifik (Hyderabad): `ap-south-2`
- China (Peking): `cn-north-1`
- China (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"
```

Weitere Informationen finden Sie unter start-loader-job in der Befehlsreferenz. 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.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**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

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"]
```

```
    }'
```

Informationen zum Erstellen und Zuordnen einer IAM-Rolle zu einem Neptune-Cluster finden Sie unter Voraussetzungen: IAM-Rolle und Amazon-S3-Zugriff.

> **Note**
>
> Unter Neptune-Loader-Anforderungsparameter) finden Sie detaillierte Informationen zu Ladeanforderungsparametern. In Kürze:
> Der Parameter `source` akzeptiert einen Amazon–S3-URI, der auf eine einzelne Datei oder auf einen Ordner verweist. Wenn Sie einen Ordner angeben, lädt Neptune jede Datendatei in den Ordner.
> Der Ordner enthält möglicherweise mehrere Eckpunkt- und Kantendateien.
> Der URI kann in einem der folgenden Formate angegeben sein.
>
> - `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`
>
> Der Parameter `format` kann eines der folgenden Formate angeben:
>
> - Gremlin-CSV-Format (`csv`) für Gremlin-Eigenschaftsdiagramme
> - OpenCypher-CSV-Format (`opencypher`) für openCypher-Eigenschaftsdiagramme
> - N-Triples (`ntriples`)-Format für RDF/SPARQL
> - N-Quads (`nquads`) Format für RDF/SPARQL
> - RDF/XML (`rdfxml`) Format für RDF/SPARQL
> - Turtle (`turtle`)-Format für RDF/SPARQL
>
> Mit dem optionalen Parameter `parallelism` können Sie die Anzahl der im Massen-Ladevorgang verwendeten Threads einschränken. Mögliche Einstellungen sind `LOW`, `MEDIUM`, `HIGH` oder `OVERSUBSCRIBE`.
> Wenn `updateSingleCardinalityProperties` auf `"FALSE"` eingestellt ist, gibt der Loader einen Fehler zurück, wenn mehr als ein Wert in einer Quelldatei bereitgestellt wird, die für eine Edge- oder Einzel-Kardinalität-Vertex-Eigenschaft geladen wird.

Wenn `queueRequest` auf `"TRUE"` eingestellt wird, hat dies zur Folge, dass die Ladeanforderung in eine Warteschlange gestellt wird, wenn bereits ein Ladeauftrag ausgeführt wird.
Mit dem Parameter `dependencies` wird die Ausführung der Ladeanforderung vom erfolgreichen Abschluss eines oder mehrerer Ladeaufträge abhängig gemacht, die bereits in die Warteschlange gestellt wurden.

3. Der Neptune-Loader gibt eine Auftrags-`id` zurück, mit der Sie den Status prüfen oder den Ladevorgang abbrechen können, z. B.:

```
{
    "status" : "200 OK",
    "payload" : {
        "loadId" : "ef478d76-d9da-4d94-8ff1-08d9d4863aa5"
    }
}
```

4. Geben Sie Folgendes ein, um den Status des Ladevorgangs mit der `loadId` aus Schritt 3 abzurufen:

   AWS CLI

```
aws neptunedata get-loader-job-status \
  --endpoint-url https://your-neptune-endpoint:port \
  --load-id ef478d76-d9da-4d94-8ff1-08d9d4863aa5
```

   Weitere Informationen finden Sie unter get-loader-job-status in der Befehlsreferenz. 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**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

curl

```
curl -G 'https://your-neptune-endpoint:port/loader/ef478d76-
d9da-4d94-8ff1-08d9d4863aa5'
```

Wenn der Status des Ladevorgangs einen Fehler angibt, können Sie mehr Details zum Status und eine Liste der Fehler anfordern. Weitere Informationen und Beispiele finden Sie unter Neptune Loader-API Get-Status.

5. (Optional) Brechen Sie den `Load`-Auftrag ab.

   Geben Sie Folgendes ein, um den Befehl `Delete` für den Loader-Auftrag mit der Auftrags-`id` aus Schritt 3 auszuführen:

   AWS CLI

   ```
   aws neptunedata cancel-loader-job \
     --endpoint-url https://your-neptune-endpoint:port \
     --load-id ef478d76-d9da-4d94-8ff1-08d9d4863aa5
   ```

   Weitere Informationen finden Sie unter cancel-loader-job in der Befehlsreferenz. 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**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

curl

```
curl -X DELETE 'https://your-neptune-endpoint:port/loader/ef478d76-
d9da-4d94-8ff1-08d9d4863aa5'
```

Der DELETE-Befehl gibt den HTTP-Code `200 OK` nach erfolgtem Abbruch zurück.

Die Daten aus Dateien des Ladeauftrags, der beendet wurde, werden nicht zurückgesetzt. Die Daten bleiben in der Neptune-DB-Instance.

Optimieren einer Amazon-Neptune-Massenladung

Sie können mit den folgenden Strategien die Ladezeit für eine Neptune-Massenladung minimieren:

- Bereinigen der Daten:
  - Achten Sie darauf, Ihre Daten vor dem Laden in ein unterstütztes Datenformat zu konvertieren.
  - Entfernen Sie alle Duplikate oder bekannten Fehler.
  - Reduzieren Sie die Anzahl der eindeutigen Prädikate (wie Eigenschaften von Kanten und Eckpunkten) so weit wie möglich.
- Optimieren der Dateien:
  - Wenn Sie große Dateien wie CSV-Dateien aus einem Amazon-S3-Bucket laden, verwaltet der Loader die Gleichzeitigkeit für Sie, indem er diese Dateien in Blöcke aufteilt, die parallel geladen werden können. Die Verwendung einer sehr großen Anzahl sehr kleiner Dateien kann diesen Prozess verlangsamen.
  - Wenn Sie mehrere Dateien aus einem Amazon S3 S3-Präfix laden, lädt der Loader automatisch zuerst Vertex-Dateien und anschließend Edge-Dateien. Wenn Sie jedoch wissen, dass Sie nur Kantendateien laden, `edgeOnlyLoad` können Sie festlegen, dass der erste Durchlauf, bei dem alle Dateien gescannt werden, um `TRUE` ihren Inhalt (Scheitelpunkte oder Kanten) zu bestimmen, übersprungen wird, sodass alle gefundenen Vertex-Dateien vor allen Edge-Dateien geladen werden. Dies kann die Ladezeit erheblich beschleunigen, insbesondere wenn es sich um viele Edge-Dateien handelt. Falls einige Vertex-Dateien auch im selben Amazon S3 S3-Präfix (`source`Parameter) vorhanden sind, werden sie geladen, jedoch ohne jegliche Sortiergarantie im Vergleich zu anderen Dateien. Wenn einige `from` oder `to` Scheitelpunkte nicht in der Datenbank vorhanden sind, kann es außerdem vorkommen, dass beim Einfügen von Kanten Fehler mit der Meldung gemeldet werden. `FROM_OR_TO_VERTEX_ARE_MISSING` Es hat sich bewährt, Knoten und Kanten in ein separates Amazon S3-Präfix einzufügen.
- Überprüfen der Loader-Einstellungen:
  - Wenn Sie während des Ladens keine anderen Operationen ausführen müssen, verwenden Sie den Parameter `OVERSUBSCRIBE parallelism`. Diese Parametereinstellung bewirkt, dass der Bulk-Loader alle verfügbaren CPU-Ressourcen verwendet, wenn er ausgeführt wird. In der Regel

werden 60 bis 70% der CPU-Kapazität benötigt, um den Vorgang so schnell laufen zu lassen, wie es die Einschränkungen zulassen. I/O

> **Note**
>
> Wenn `parallelism` auf `OVERSUBSCRIBE` oder `HIGH` festgelegt ist (die Standardeinstellung), besteht beim Laden von openCypher-Daten die Gefahr, dass Threads auf eine Race-Bedingung und einen Deadlock treffen, was zum Fehler `LOAD_DATA_DEADLOCK` führt. Legen Sie in diesem Fall `parallelism` auf eine niedrigere Einstellung fest und wiederholen Sie den Ladevorgang.

- Wenn Ihr Ladeauftrag mehrere Ladeanforderungen umfasst, verwenden Sie den Parameter `queueRequest`. Wenn Sie `queueRequest` auf `TRUE` festlegen, kann Neptune Ihre Abfragen in eine Warteschlange einstellen. So müssen Sie nicht warten, bis eine Abfrage abgeschlossen ist, bevor Sie eine weitere senden.
- Wenn Ihre Ladeanforderungen in eine Warteschlange eingestellt werden, können Sie mit dem Parameter `dependencies` Abhängigkeitsstufen einrichten, sodass der Ausfall eines Auftrags zum Ausfall abhängiger Aufträge führt. Dies kann Inkonsistenzen in den geladenen Daten vermeiden.
- Wenn ein Ladeauftrag die Aktualisierung zuvor geladener Werte umfasst, müssen Sie den Parameter `updateSingleCardinalityProperties` auf `TRUE` festlegen. Andernfalls behandelt der Loader den Versuch, einen vorhandenen Wert mit einfacher Kardinalität zu aktualisieren, als Fehler. Bei Gremlin-Daten wird die Kardinalität auch in den Überschriften der Eigenschaftsspalten angegeben (siehe Eigenschaftenspalten-Header).

  > **Note**
  >
  > Der `updateSingleCardinalityProperties`-Parameter ist für RDF-Daten (Resource Description Framework) nicht verfügbar.

- Sie können den Parameter `failOnError` verwenden, um festzulegen, ob Massenladeoperationen fehlschlagen oder fortgesetzt werden sollen, wenn ein Fehler auftritt. Sie können den Parameter `mode` auch verwenden, um sicherzustellen, dass ein Ladeauftrag ab dem Punkt weiter geladen wird, an dem ein früherer Auftrag fehlgeschlagen ist, anstatt bereits geladene Daten erneut zu laden.

- Skalieren – Legen Sie die Sie die Writer-Instance Ihres DB-Clusters vor dem Massenladen auf die maximale Größe fest. Beachten Sie, dass Sie in diesem Fall alle Read-Replica-Instances

im DB-Cluster entweder hochskalieren oder entfernen müssen, bis das Laden der Daten abgeschlossen ist.

Wenn der Massenladevorgang abgeschlossen ist, müssen Sie die Writer-Instance wieder herunterskalieren.

> **Important**
>
> Wenn es aufgrund von Replikationsverzögerungen während eines Massenladens zu einem Zyklus wiederholter Read-Replica-Neustarts kommt, können Ihre Replikate wahrscheinlich nicht mit dem Writer im DB-Cluster Schritt halten. Sie können die Reader so skalieren, dass sie größer als der Writer sind. Sie können sie auch während des Massenladens vorübergehend entfernen und nach dem Abschluss des Vorgangs erneut erstellen.

Weitere Informationen zum Einrichten von Loader-Anforderungsparametern finden Sie unter Anforderungsparameter.

Neptune-Loader-Referenz

Dieser Abschnitt beschreibt die `Loader`-APIs für Amazon Neptune die über den HTTP-Endpunkt einer Neptune-DB-Instance verfügbar sind.

> **Note**
>
> Die Liste der Fehler- und Feed-Meldungen, die der Loader im Fehlerfall zurückgibt, finden Sie unter Neptune Loader-Fehler- und Feed-Nachrichten.

Inhalt

- Neptune-Loader-Befehl
  - Neptune-Loader-Anforderungssyntax
  - Neptune-Loader-Anforderungsparameter
    - Besondere Überlegungen beim Laden von openCypher-Daten
  - Neptune-Loader-Antwortsyntax
  - Neptune-Loader-Fehler

  - Neptune-Loader-Beispiele
- Neptune Loader-API Get-Status
  - Neptune Loader-Anfragen Get-Status
    - Syntax der Loader-Anfrage Get-Status
    - Neptune Loader-Anforderungsparameter Get-Status
  - Antworten von Neptune Loader Get-Status
    - JSON-Layout für Neptune Get-Status Loader-Antworten
    - Neptune Loader Get-Status OverallStatus - und FailedFeed-Antwortobjekte
    - Neptune Get-Status Loader-Fehlerantwortobjekt
    - Neptune Loader Get-Status ErrorLogs-Antwortobjekt
  - Neptune Loader-Beispiele Get-Status
    - Beispiel für eine Ladestatusanforderung
    - Beispiel für eine loadIds-Anforderung
    - Beispiel für eine Detailstatusanforderung
  - Beispiele für Neptune Loader ErrorLogs Get-Status
    - Beispiel für eine Detailstatusantwort bei Fehlern
    - Beispiel für einen Fehler, bei dem die Aufgabe beim Prefetch unterbrochen wurde
- Neptune-Loader – Auftrag abbrechen
  - Anforderungssyntax – Auftrag abbrechen
  - Anforderungsparameter – Auftrag abbrechen
  - Antwortsyntax– Auftrag abbrechen
  - Fehler– Auftrag abbrechen
  - Fehlermeldungen – Auftrag abbrechen
  - Beispiele – Auftrag stornieren

Neptune-Loader-Befehl

Lädt Daten aus einem Amazon-S3-Bucket in eine Neptune-DB-Instance.

Zum Laden der Daten müssen Sie eine HTTP-POST-Anforderung an den `https://`*`your-neptune-endpoint`*`:`*`port`*`/loader`-Endpunkt senden. Die Parameter für die `loader` Anfrage können im POST Hauptteil oder als Parameter gesendet werden. URL-encoded

> **Important**
>
> Der MIME-Typ muss `application/json` sein.

Der Amazon S3 S3-Bucket muss sich in derselben AWS Region wie der Cluster befinden.

> **Note**
>
> Sie können verschlüsselte Daten aus Amazon S3 laden, wenn sie mit dem Amazon-S3-Modus `SSE-S3` verschlüsselt wurden. In diesem Fall kann Neptune Ihre Anmeldeinformationen vortäuschen und `s3:getObject`-Aufrufe in Ihrem Namen ausgeben. Sie können auch verschlüsselte Daten aus Amazon S3 laden, die mit dem Modus `SSE-KMS` verschlüsselt wurden, wenn Ihre IAM-Rolle die notwendigen Berechtigungen für den Zugriff auf AWS KMS besitzt. Ohne die entsprechenden AWS KMS Berechtigungen schlägt der Massenladevorgang fehl und es wird eine `LOAD_FAILED` Antwort zurückgegeben. Neptune unterstützt zurzeit nicht das Laden von Amazon-S3-Daten, die mit dem Modus `SSE-C` verschlüsselt wurden.

Sie müssen nicht warten, bis ein Ladeauftrag abgeschlossen ist, bevor Sie einen weiteren Ladeauftrag starten. Neptune kann bis zu 64 Auftragsanforderungen gleichzeitig in eine Warteschlange einstellen, wenn ihre `queueRequest`-Parameter auf `"TRUE"` festgelegt sind. Die Aufträge werden nach dem First-In-First-Out (FIFO)-Prinzip eingereiht. Wenn Sie nicht möchten, dass ein Ladeauftrag in die Warteschlange eingereiht wird, können Sie seinen Parameter `queueRequest` auf `"FALSE"` (den Standardwert) festlegen, damit der Ladeauftrag fehlschlägt, wenn ein bereits ein anderer Ladeauftrag ausgeführt wird.

Sie können mit dem Parameter `dependencies` einen Auftrag in die Warteschlange stellen, der nur ausgeführt werden darf, nachdem bestimmte vorherige Aufträge in der Warteschlange erfolgreich durchgeführt wurden. Wenn Sie so verfahren und einer dieser angegebenen Aufträge fehlschlägt, wird Ihr Auftrag nicht ausgeführt und sein Status wird auf `LOAD_FAILED_BECAUSE_DEPENDENCY_NOT_SATISFIED` gesetzt.

Neptune-Loader-Anforderungssyntax

```
{
  "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 Edge-Syntax

Für eine `edgeOnlyLoad` wäre die Syntax:

```
{
"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"]
}
```

Neptune-Loader-Anforderungsparameter

- **`source`** – Ein Amazon-S3-URI.

  Der Parameter `SOURCE` akzeptiert Amazon-S3-URIs, die eine einzelne Datei, mehrere Dateien, einen Ordner oder mehrere Ordner angeben. Neptune lädt jede Datendatei in den Ordner, der angegeben ist.

Der URI kann in einem der folgenden Formate angegeben sein.

- `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`

Das `object-key-name` Element der URI entspricht dem Präfixparameter in einem Amazon S3 ListObjectsS3-API-Aufruf. Es gibt alle Objekte im angegebenen Amazon-S3-Bucket an, deren Namen mit diesem Präfix beginnen. Das kann eine einzelne Datei oder ein Ordner oder mehrere and/or Dateiordner sein.

Die angegebenen Ordner können mehrere Eckpunkt- und Kantendateien enthalten.

Wenn Sie beispielsweise die folgende Ordnerstruktur und die folgenden Dateien in einem Amazon S3 S3-Bucket mit dem Namen haben`bucket-name`:

```
s3://bucket-name/a/bc
s3://bucket-name/ab/c
s3://bucket-name/ade
s3://bucket-name/bcd
```

Wenn der Quellparameter als angegeben ists`3://bucket-name/a`, werden die ersten drei Dateien geladen.

```
s3://bucket-name/a/bc
s3://bucket-name/ab/c
s3://bucket-name/ade
```

- **`format`** – Das Format der Daten. Weitere Informationen zu Datenformaten für den Neptune-Befehl `Loader` finden Sie unter Verwenden des Amazon Neptune Bulk Loaders zur Datenaufnahme.

  Zulässige Werte

  - **`csv`** für das Gremlin-CSV-Datenformat.
  - **`opencypher`** für das openCypher-CSV-Datenformat.
  - **`ntriples`**für das N-Triples RDF-Datenformat.
  - **`nquads`**für das N-Quads RDF-Datenformat.
  - **`rdfxml`** für das RDF\XML-RDF-Datenformat.

- **`turtle`** für das Turtle-RDF-Datenformat.

- **`iamRoleArn`** – Der Amazon-Ressourcenname (ARN) für eine IAM-Rolle, die von der Neptune-DB-Instance für den Zugriff auf den S3-Bucket angenommen wird. Informationen zum Erstellen einer Rolle, die über Zugriff auf Amazon S3 verfügt, und zum Verknüpfen dieser Rolle mit einem Neptune-Cluster finden Sie unter Voraussetzungen: IAM-Rolle und Amazon-S3-Zugriff.

  Ab Engine-Version 1.2.1.0.R3 können Sie auch mehrere IAM-Rollen verketten, wenn sich die Neptune-DB-Instance und der Amazon S3 S3-Bucket in unterschiedlichen Konten befinden. AWS In diesem Fall enthält `iamRoleArn` eine durch Komma getrennte Liste von Rollen-ARNs, wie in Verketten von IAM-Rollen in Amazon Neptune beschrieben. Beispiel:

  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"
  ```

  Weitere Informationen finden Sie unter start-loader-job in der Befehlsreferenz. 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.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**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

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`**— Der `region` Parameter muss mit der AWS Region des Clusters und dem S3-Bucket übereinstimmen.

Amazon Neptune ist in den folgenden -Regionen verfügbar:

- USA Ost (Nord-Virginia): `us-east-1`
- USA Ost (Ohio): `us-east-2`
- USA West (Nordkalifornien): `us-west-1`
- USA West (Oregon): `us-west-2`
- Kanada (Zentral): `ca-central-1`
- Kanada West (Calgary): `ca-west-1`
- Südamerika (São Paulo): `sa-east-1`
- Europa (Stockholm): `eu-north-1`
- Europa (Spanien): `eu-south-2`
- Europa (Irland): `eu-west-1`
- Europa (London): `eu-west-2`
- Europa (Paris): `eu-west-3`
- Europa (Frankfurt): `eu-central-1`
- Naher Osten (Bahrain): `me-south-1`
- Naher Osten (VAE): `me-central-1`
- Israel (Tel Aviv): `il-central-1`
- Afrika (Kapstadt): `af-south-1`
- Asien-Pazifik (Hongkong): `ap-east-1`
- Asien-Pazifik (Tokio): `ap-northeast-1`
- Asien-Pazifik (Seoul): `ap-northeast-2`
- Asien-Pazifik (Osaka): `ap-northeast-3`
- Asien-Pazifik (Singapur): `ap-southeast-1`
- Asien-Pazifik (Sydney): `ap-southeast-2`
- Asien-Pazifik (Jakarta): `ap-southeast-3`
- Asien-Pazifik (Melbourne): `ap-southeast-4`
- Asien-Pazifik (Malaysia): `ap-southeast-5`
- Asien-Pazifik (Mumbai): `ap-south-1`
- Asien-Pazifik (Hyderabad): `ap-south-2`
- China (Peking): `cn-north-1`

  - China (Ningxia): `cn-northwest-1`
  - AWS GovCloud (US-West): `us-gov-west-1`
  - AWS GovCloud (US-East): `us-gov-east-1`
- **`mode`** – Der Ladeauftragsmodus.

  Zulässige Werte: `RESUME`, `NEW`, `AUTO`.

  Standardwert: `AUTO`

  - `RESUME` – Im Modus RESUME sucht der Loader nach einem vorherigen Ladevorgang aus dieser Quelle. Wenn der Loader einen vorherigen Ladevorgang findet, setzt er diesen Ladeauftrag fort. Wenn kein vorheriger Ladeauftrag gefunden wird, stoppt der Loader.

    Der Loader vermeidet das erneute Laden von Dateien, die in einem früheren Auftrag erfolgreich geladen wurden. Er versucht nur, fehlgeschlagene Dateien zu verarbeiten. Wenn Sie zuvor geladene Daten aus dem Neptune-Cluster gelöscht haben, werden diese Daten in diesem Modus nicht neu geladen. Wenn ein vorheriger Ladeauftrag alle Dateien aus derselben Quelle erfolgreich geladen hat, wird nichts neu geladen und der Loader gibt eine Erfolgsmeldung zurück.
  - `NEW` – Der Modus NEW erstellt eine neue Ladeanforderung unabhängig von vorherigen Ladevorgängen. Sie können diesen Modus verwenden, um alle Daten aus einer Quelle erneut zu laden, nachdem die zuvor aus Ihrem Neptune-Cluster geladenen Daten abgelegt wurden, oder um neue Daten zu laden, die in derselben Quelle verfügbar sind.
  - `AUTO` – Im Modus AUTO sucht der Loader nach einem vorherigen Ladeauftrag aus derselben Quelle. Wenn er einen Ladeauftrag findet, setzt er diesen Auftrag wie im Modus `RESUME` fort.

    Wenn der Loader keinen vorherigen Ladeauftrag aus derselben Quelle findet, lädt er wie im Modus `NEW` alle Daten aus der Quelle.
- **`edgeOnlyLoad`**— Ein Kennzeichen, das die Reihenfolge der Dateiverarbeitung beim Massenladen steuert.

  Zulässige Werte: `"TRUE"`, `"FALSE"`.

  Standardwert: `"FALSE"`.

  Wenn dieser Parameter auf „FALSE“ gesetzt ist, lädt der Loader automatisch zuerst Vertex-Dateien und danach Edge-Dateien. Dazu werden zunächst alle Dateien gescannt, um deren Inhalt

(Scheitelpunkte oder Kanten) zu bestimmen. Wenn dieser Parameter auf „TRUE“ gesetzt ist, überspringt der Loader die anfängliche Scanphase und lädt sofort alle Dateien in der Reihenfolge, in der sie erscheinen. Weitere Informationen finden Sie unter Bulk Load Optimize.

- **`failOnError`** – Ein Flag für vollständiges Anhalten bei einem Fehler.

  Zulässige Werte: `"TRUE"`, `"FALSE"`.

  Standardwert: `"TRUE"`.

  Wenn dieser Parameter auf `"FALSE"` gesetzt ist, versucht der Loader, alle Daten am angegebenen Speicherort zu laden, wobei alle Einträge mit Fehlern übersprungen werden.

  Wenn dieser Parameter auf `"TRUE"` eingestellt ist, stoppt der Loader, sobald ein Fehler auftritt. Bis zu diesem Punkt geladene Daten bleiben bestehen.

- **`parallelism`** – Dies ist ein optionaler Parameter, der festgelegt werden kann, um die Anzahl der vom Massenladevorgang verwendeten Threads zu reduzieren.

  Zulässige Werte:

  - `LOW` – Die Anzahl der verwendeten Threads entspricht der Anzahl der verfügbaren vCPUs dividiert durch 8.
  - `MEDIUM` – Die Anzahl der verwendeten Threads entspricht der Anzahl der verfügbaren vCPUs dividiert durch 2.
  - `HIGH` – Die Anzahl der verwendeten Threads entspricht der Anzahl der verfügbaren vCPUs.
  - `OVERSUBSCRIBE` – Die Anzahl der verwendeten Threads entspricht der Anzahl der verfügbaren vCPUs multipliziert mit 2. Wenn dieser Wert verwendet wird, nimmt der Massen-Loader alle verfügbaren Ressourcen in Anspruch.

    Das bedeutet jedoch nicht, dass die Einstellung `OVERSUBSCRIBE` zu einer CPU-Auslastung von 100 % führt. Da der Ladevorgang I/O gebunden ist, liegt die höchste zu erwartende CPU-Auslastung im Bereich von 60 bis 70%

  Standardwert: `HIGH`

  Die Einstellung `parallelism` kann manchmal beim Laden von openCypher-Daten zu einem Deadlock zwischen Threads führen. In diesem Fall gibt Neptune den Fehler `LOAD_DATA_DEADLOCK` zurück. Sie können das Problem im Allgemeinen beheben, indem Sie `parallelism` auf einen niedrigeren Wert festlegen und den Ladebefehl wiederholen.

- **`parserConfiguration`** – Ein optionales Objekt mit zusätzlichen Parser-Konfigurationswerten. Jeder der untergeordneten Parameter ist ebenfalls optional:

Name	Beispielwert	Description
`namedGraphUri`	*`http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph`*	Der Standardgraph für alle RDF-Formate, wenn kein Graph angegeben ist (für nicht-Quads-Formate und NQUAD-Einträge ohne Graph). Der Standardwert ist `http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph`
`baseUri`	*`http://aws.amazon.com/neptune/default`*	Der Basis-URI für die Formate RDF/XML und Turtle. Der Standardwert ist `http://aws.amazon.com/neptune/default` .

`allowEmptyStrings`	*true*	Gremlin-Benutzer müssen beim Laden von CSV-Daten leere Zeichenfolgenwerte ("") als Knoten- und Kanteneigenschaften übergeben können. Wenn `allowEmptyStrings` auf `false` (Standard) festgelegt ist, werden diese leeren Zeichenfolgen als Nullen behandelt und nicht geladen. Wenn `allowEmptyStrings` auf `true` festgelegt ist, behandelt der Loader leere Zeichenfolgen als gültige Eigenschaftswerte und lädt sie entsprechend.

Weitere Informationen finden Sie unter SPARQL-Standard-Graph und benannte Graphen.

- **`updateSingleCardinalityProperties`** – Dies ist ein optionaler Parameter, der steuert, wie der Massen-Loader einen neuen Wert für Eckpunkt- oder Kanteneigenschaften mit einfacher Kardinalität behandelt.

  Zulässige Werte: `"TRUE"`, `"FALSE"`.

  Standardwert: `"FALSE"`.

  Standardmäßig oder wenn `updateSingleCardinalityProperties` explizit als `"FALSE"` festgelegt ist, behandelt der Loader einen neuen Wert als Fehler, da er gegen die Einzel-Kardinalität verstößt.

  Wenn `updateSingleCardinalityProperties` als `"TRUE"` festgelegt ist, ersetzt der Massen-Loader auf der anderen Seite den vorhandenen Wert durch den neuen. Wenn in den Quelldateien, die geladen werden, mehrere Edge- oder Einzel-Kardinalität-Vertex-Eigenschaftswerte angegeben werden, kann der endgültige Wert am Ende des Massenladevorgangs jeder dieser neuen Werte sein. Der Loader stellt nur sicher, dass der vorhandene Wert durch einen der neuen ersetzt wurde.

- **`queueRequest`** – Dies ist ein optionaler Flag-Parameter, der angibt, ob die Ladeanforderung in die Warteschlange eingereiht werden kann oder nicht.

  Sie müssen mit dem nächsten Ladeauftrag nicht warten, bis ein Ladeauftrag abgeschlossen ist, da Neptune bis zu 64 Aufträge gleichzeitig in die Warteschlange einreihen kann, wenn alle `queueRequest`-Parameter auf `"TRUE"` festgelegt sind. Die Aufträge werden nach dem First-In-First-Out (FIFO)-Prinzip eingereiht.

  Wenn der Parameter `queueRequest` ausgelassen oder auf `"FALSE"` festgelegt wird, schlägt die Ladeanforderung fehl, wenn bereits ein anderer Ladeauftrag ausgeführt wird.

  Zulässige Werte: `"TRUE"`, `"FALSE"`.

  Standardwert: `"FALSE"`.

- **`dependencies`** – Dies ist ein optionaler Parameter, mit dem eine Ladeanforderung in der Warteschlange vom erfolgreichen Abschluss eines oder mehrerer früherer Aufträge in der Warteschlange abhängig gemacht werden kann.

  Neptune kann bis zu 64 Ladeanforderungen gleichzeitig in die Warteschlange einreihen, wenn die `queueRequest`-Parameter auf `"TRUE"` festgelegt sind. Mit dem Parameter `dependencies` können Sie die Ausführung einer solchen Anforderung in der Warteschlange vom erfolgreichen Abschluss einer oder mehrerer spezifizierter früherer Anforderungen in der Warteschlange abhängig machen.

  Wenn Ladung `Job-A` und `Job-B` beispielsweise unabhängig voneinander sind, mit Ladung `Job-C` aber erst nach Abschluss von `Job-A` und `Job-B` begonnen werden kann, gehen Sie wie folgt vor:

  1. Senden Sie `load-job-A` und `load-job-B` nacheinander in beliebiger Reihenfolge, und speichern Sie ihre Ladekennungen.
  2. Senden Sie `load-job-C` mit den Ladekennungen der beiden Aufträge in seinem `dependencies`-Feld:

  ```
  "dependencies" : ["job_A_load_id", "job_B_load_id"]
  ```

  Aufgrund des Parameters `dependencies` startet der Bulk-Loader `Job-C` erst dann, nachdem `Job-A` und `Job-B` erfolgreich abgeschlossen wurden. Wenn einer von ihnen fehlschlägt, Job-C wird er nicht ausgeführt und sein Status wird auf gesetzt`LOAD_FAILED_BECAUSE_DEPENDENCY_NOT_SATISFIED`.

Auf diese Weise können Sie mehrere Abhängigkeitsebenen einrichten, sodass das Fehlschlagen eines Auftrags dazu führt, dass alle Anforderungen, die direkt oder indirekt davon abhängig sind, abgebrochen werden.

- **`userProvidedEdgeIds`** – Dieser Parameter ist nur erforderlich, wenn openCypher-Daten geladen werden, die Beziehungs-IDs enthalten. Er muss enthalten und auf `True` festgelegt sein, wenn openCypher-Beziehungs-IDs explizit in den Ladedaten angegeben werden (empfohlen).

  Wenn `userProvidedEdgeIds` fehlt oder auf `True` festgelegt ist, muss in jeder Beziehungsdatei im Ladevorgang die Spalte `:ID` vorhanden sein.

  Wenn `userProvidedEdgeIds` vorhanden und auf `False` festgelegt ist, dürfen Beziehungsdateien im Ladevorgang die Spalte `:ID` nicht enthalten. Stattdessen generiert der Neptune-Loader automatisch eine ID für jede Beziehung.

  Wenn Beziehungs-IDs explizit angegeben sind, kann der Loader den Ladevorgang nach der Behebung eines Fehlers in den CSV-Daten fortsetzen, ohne bereits geladene Beziehungen erneut laden zu müssen. Wenn Beziehungs-IDs nicht explizit angegeben sind, kann der Loader einen fehlgeschlagenen Ladevorgang nach der Korrektur einer Beziehungsdatei nicht fortsetzen und muss stattdessen alle Beziehungen erneut laden.

- `accessKey` – [veraltet] Die Zugriffsschlüssel-ID einer IAM-Rolle mit Zugriff auf den S3-Bucket und Datendateien.

  Stattdessen wird der Parameter `iamRoleArn` empfohlen. Informationen zum Erstellen einer Rolle, die über Zugriff auf Amazon S3 verfügt, und zum Verknüpfen dieser Rolle mit einem Neptune-Cluster finden Sie unter Voraussetzungen: IAM-Rolle und Amazon-S3-Zugriff.

  Weitere Informationen finden Sie unter Zugriffsschlüssel (Zugriffsschlüssel-ID und geheimer Zugriffsschlüssel).

- `secretKey` – [veraltet] Stattdessen wird der Parameter `iamRoleArn` empfohlen. Informationen zum Erstellen einer Rolle, die über Zugriff auf Amazon S3 verfügt, und zum Verknüpfen dieser Rolle mit einem Neptune-Cluster finden Sie unter Voraussetzungen: IAM-Rolle und Amazon-S3-Zugriff.

  Weitere Informationen finden Sie unter Zugriffsschlüssel (Zugriffsschlüssel-ID und geheimer Zugriffsschlüssel).

Besondere Überlegungen beim Laden von openCypher-Daten

- Beim Laden von openCypher-Daten im CSV-Format muss der Formatparameter auf `opencypher` festgelegt werden.
- Der `updateSingleCardinalityProperties` Parameter wird für OpenCypher-Ladevorgänge unterstützt. Alle OpenCypher-Eigenschaften haben eine einfache Kardinalität, und wenn ein ID-Wert mehr als einmal vorkommt, wird er als Duplikat oder als Einfügefehler behandelt (siehe unten).
- Der Neptune-Loader behandelt Duplikate in openCypher-Daten wie folgt:
  - Wenn der Loader mehrere Zeilen mit derselben Knoten-ID findet, werden sie entsprechend der folgenden Regel zusammengeführt:
    - Alle Bezeichnungen in den Zeilen werden dem Knoten hinzugefügt.
    - Für jede Eigenschaft wird nur einer der Eigenschaftswerte geladen. Die Auswahl des Werts, der geladen wird, ist nicht deterministisch.
  - Wenn der Loader mehrere Zeilen mit derselben Beziehungs-ID findet, wird nur eine dieser Zeilen geladen. Die Auswahl der Zeile, die geladen wird, ist nicht deterministisch.
  - Standardmäßig aktualisiert der Loader keine Eigenschaftswerte eines vorhandenen Knotens oder einer Beziehung in der Datenbank, wenn er Ladedaten mit der ID des vorhandenen Knotens oder der Beziehung findet. Es werden jedoch Knotenbezeichnungen und -eigenschaften geladen, die in vorhandenen Knoten oder Beziehungen nicht vorhanden sind. Um vorhandene Eigenschaftswerte zu aktualisieren, legen Sie `updateSingleCardinalityProperties` den Wert auf fest`"TRUE"`.
- Sie müssen Beziehungen zwar keine IDs zuweisen, dies wird jedoch in der Regel empfohlen (siehe den Parameter `userProvidedEdgeIds` oben). Ohne explizite Beziehungs-IDs muss der Loader bei einem Fehler in einer Beziehungsdatei alle Beziehungen erneut laden, anstatt den Ladevorgang ab der Stelle fortzusetzen, an der er fehlgeschlagen ist.

  Wenn die Ladedaten keine expliziten Beziehungs-IDs enthalten, kann der Loader keine duplizierten Beziehungen erkennen.

Dies ist ein Beispiel für einen openCypher-Ladebefehl:

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"
```

Weitere Informationen finden Sie unter start-loader-job in der Befehlsreferenz. 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.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**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

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"
      }'
```

Die Loader-Antwort ist dieselbe wie normal. Beispiel:

```
{
  "status" : "200 OK",
  "payload" : {
    "loadId" : "guid_as_string"
  }
}
```

Neptune-Loader-Antwortsyntax

```
{
```

```
    "status" : "200 OK",
    "payload" : {
        "loadId" : "guid_as_string"
    }
}
```

200 OK

Bei einem erfolgreich gestarteten Ladeauftrag wird ein `200`-Code zurückgegeben.

Neptune-Loader-Fehler

Wenn ein Fehler auftritt, wird ein JSON-Objekt im `BODY` der Antwort zurückgegeben. Das `message`-Objekt enthält eine Beschreibung des Fehlers.

Fehlerkategorien

- `Error 400` – Syntaxfehler geben den HTTP-Fehler `400` (ungültige Anforderung) zurück. Die Nachricht beschreibt den Fehler.
- `Error 500` – Eine gültige Anforderung, die nicht verarbeitet werden kann, gibt den internen HTTP-Serverfehler `500` zurück. Die Nachricht beschreibt den Fehler.

Im Folgenden sind mögliche Fehlermeldungen des Loaders mit einer Beschreibung des jeweiligen Fehlers aufgeführt.

Loader-Fehlermeldungen

- `Couldn't find the AWS credential for iam_role_arn` (HTTP 400)

  Die Anmeldeinformationen wurden nicht gefunden. Überprüfen Sie die angegebenen Anmeldeinformationen anhand der IAM-Konsole oder -Ausgabe. AWS CLI Stellen Sie sicher, dass Sie die in `iamRoleArn` angegebene IAM-Rolle zum Cluster hinzugefügt haben.
- `S3 bucket not found for source` (HTTP 400)

  Der S3-Bucket ist nicht vorhanden. Überprüfen Sie den Namen des Buckets.
- `The source source-uri does not exist/not reachable` (HTTP 400)

  Es wurden keine übereinstimmenden Dateien im S3-Bucket gefunden.
- `Unable to connect to S3 endpoint. Provided source = source-uri and region = aws-region` (HTTP 500)

Verbindung zu Amazon S3 kann nicht hergestellt werden. Die Region muss mit der Cluster-Region übereinstimmen. Stellen Sie sicher, dass Sie über einen VPC-Endpunkt verfügen. Informationen zum Erstellen eines VPC-Endpunkts finden Sie unter Erstellen eines Amazon-S3-VPC-Endpunkts.

- `Bucket is not in provided Region (aws-region)` (HTTP 400)

  Der Bucket muss sich in derselben AWS Region wie Ihre Neptune-DB-Instance befinden.

- `Unable to perform S3 list operation` (HTTP 400)

  Der IAM-Benutzer oder die angegebene Rolle verfügt nicht über `List`-Berechtigungen für den Bucket oder Ordner. Prüfen Sie die Richtlinie oder die Zugriffskontrollliste (ACL) für den Bucket.

- `Start new load operation not permitted on a read replica instance` (HTTP 405)

  Das Laden ist ein Schreibvorgang. Versuchen Sie erneut, den Cluster-Endpunkt zu laden. read/write

- `Failed to start load because of unknown error from S3` (HTTP 500)

  Amazon S3 hat einen unbekannten Fehler zurückgegeben. Wenden Sie sich an AWS Support.

- `Invalid S3 access key` (HTTP 400)

  Der Zugriffsschlüssel ist ungültig. Prüfen Sie die angegebenen Anmeldeinformationen.

- `Invalid S3 secret key` (HTTP 400)

  Der geheime Schlüssel ist ungültig. Prüfen Sie die angegebenen Anmeldeinformationen.

- `Max concurrent load limit breached` (HTTP 400)

  Wenn eine Ladeanforderung ohne `"queueRequest" : "TRUE"` gesendet wird und derzeit ein Ladeauftrag ausgeführt wird, schlägt die Anforderung mit diesem Fehler fehl.

- `Failed to start new load for the source "source name". Max load task queue size limit breached. Limit is 64` (HTTP 400)

  Neptune unterstützt Warteschlangen mit bis zu 64 Loader-Aufträgen gleichzeitig. Wenn eine zusätzliche Ladeanforderung an eine Warteschlange mit bereits 64 Aufträgen gesendet wird, schlägt die Anforderung mit dieser Meldung fehl.

Neptune-Loader-Beispiele

Dieses Beispiel zeigt, wie der Neptune-Loader verwendet wird, um Daten im Gremlin CSV-Format in eine Neptune-Graphdatenbank zu laden. Die Anfrage wird als HTTP-POST-Anfrage an den Neptune-Loader-Endpunkt gesendet, und der Anforderungstext enthält die erforderlichen Parameter, um die Datenquelle, das Format, die IAM-Rolle und andere Konfigurationsoptionen anzugeben. Die Antwort enthält die Lade-ID, mit der der Fortschritt des Datenladevorgangs verfolgt werden kann.

Example Anforderung

Im Folgenden ist eine Anforderung dargestellt, die über HTTP POST mit dem `curl`-Befehl gesendet wird. Dabei wird eine Datei im Neptune-CSV-Format geladen. Weitere Informationen finden Sie unter Gremlin-Ladedatenformat.

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
```

Weitere Informationen finden Sie unter start-loader-job in der Befehlsreferenz. 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.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**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

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 Antwort

```
{
    "status" : "200 OK",
    "payload" : {
        "loadId" : "ef478d76-d9da-4d94-8ff1-08d9d4863aa5"
    }
}
```

Neptune Loader-API Get-Status

Ruft den Status eines `loader`-Auftrags ab.

Zum Abrufen des Ladestatus müssen Sie eine HTTP-`GET`-Anforderung an den `https://`*`your-neptune-endpoint`*`:`*`port`*`/loader`-Endpunkt senden. Zum Abrufen des Status für eine bestimmte Ladeanforderung müssen Sie die `loadId` als URL-Parameter einschließen oder die `loadId` dem URL-Pfad voranstellen.

Neptune verfolgt nur die letzten 1.024 Massenladeaufgaben und speichert pro Auftrag nur die letzten 10 000 Fehlerdetails.

Die Liste der Fehler- und Feed-Meldungen, die der Loader im Fehlerfall zurückgibt, finden Sie unter Neptune Loader-Fehler- und Feed-Nachrichten.

Inhalt

- Neptune Loader-Anfragen Get-Status
  - Syntax der Loader-Anfrage Get-Status
  - Neptune Loader-Anforderungsparameter Get-Status
- Antworten von Neptune Loader Get-Status
  - JSON-Layout für Neptune Get-Status Loader-Antworten

  - Neptune Loader Get-Status OverallStatus - und FailedFeed-Antwortobjekte
  - Neptune Get-Status Loader-Fehlerantwortobjekt
  - Neptune Loader Get-Status ErrorLogs-Antwortobjekt
- Neptune Loader-Beispiele Get-Status
  - Beispiel für eine Ladestatusanforderung
  - Beispiel für eine loadIds-Anforderung
  - Beispiel für eine Detailstatusanforderung
- Beispiele für Neptune Loader ErrorLogs Get-Status
  - Beispiel für eine Detailstatusantwort bei Fehlern
  - Beispiel für einen Fehler, bei dem die Aufgabe beim Prefetch unterbrochen wurde

Neptune Loader-Anfragen Get-Status

Syntax der Loader-Anfrage Get-Status

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

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

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

Neptune Loader-Anforderungsparameter Get-Status

- **`loadId`** – Die ID des Ladeauftrags. Wenn Sie keine `loadId` angeben, wird eine Liste der Lade-IDs zurückgegeben.
- **`details`** – Fügt über den Gesamtstatus hinausgehende Details hinzu.

  Zulässige Werte: `TRUE`, `FALSE`.

  Standardwert: `FALSE`.
- **`errors`** – Fügt die Liste der Fehler ein.

  Zulässige Werte: `TRUE`, `FALSE`.

  Standardwert: `FALSE`.

Die Fehlerliste ist segmentiert. Die Parameter `page` und `errorsPerPage` erlauben Ihnen das seitenweise Durchlaufen aller Fehler.

- **`page`** – Die Fehlerseitenzahl. Dieser Wert ist nur gültig, wenn der `errors`-Parameter auf `TRUE` eingestellt ist.

  Zulässige Werte: Positive Ganzzahlen.

  Standardwert: 1.

- **`errorsPerPage`** – Die Anzahl der Fehler pro Seite. Dieser Wert ist nur gültig, wenn der `errors`-Parameter auf `TRUE` eingestellt ist.

  Zulässige Werte: Positive Ganzzahlen.

  Standardwert: 10.

- **`limit`** – Die Anzahl der Lade-IDs, die aufgelistet werden sollen. Nur gültig, wenn eine Liste der Lade-IDs angefordert wird, indem eine `GET`-Anforderung ohne angegebene `loadId` gesendet wird.

  Zulässige Werte: Positive Ganzzahlen von 1 bis 100.

  Standardwert: 100.

- **`includeQueuedLoads`** – Ein optionaler Parameter, mit dem die Lade-IDs von Ladeanforderungen in der Warteschlange ausgeschlossen werden können, wenn eine Liste von Ladekennungen angefordert wird.

  Standardmäßig werden die Ladekennungen aller Ladeaufträge mit dem Status `LOAD_IN_QUEUE` in eine solche Liste aufgenommen. Sie werden vor den Ladekennungen anderer Aufträge sortiert nach dem Zeitpunkt angezeigt, zu dem sie der Warteschlange vom letzten bis zum frühesten hinzugefügt wurden.

  Zulässige Werte: `TRUE`, `FALSE`.

  Standardwert: `TRUE`.

Antworten von Neptune Loader Get-Status

Die folgende Beispielantwort der Get-Status Neptune-API beschreibt die Gesamtstruktur der Antwort, erklärt die verschiedenen Felder und ihre Datentypen sowie die Fehlerbehandlung und die Fehlerprotokolldetails.

JSON-Layout für Neptune Get-Status Loader-Antworten

Das allgemeine Layout einer Loader-Status-Antwort sieht wie folgt aus:

```
{
    "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" : [ ]
        }
    }
}
```

Neptune Loader Get-Status OverallStatus - und FailedFeed-Antwortobjekte

Die möglichen Antworten, die für jeden fehlgeschlagenen Feed zurückgegeben werden, einschließlich Fehlerbeschreibungen, sind die gleichen wie für das Objekt `overallStatus` in einer `Get-Status`-Antwort.

Die folgenden Felder werden im Objekt `overallStatus` für alle Ladevorgänge und im Objekt `failedFeeds` für jeden fehlgeschlagenen Feed angegeben:

- **`fullUri`** – Der URI der Datei/der Dateien, die geladen werden soll(en).

  Typ: Zeichenfolge

  Format: `s3://`*`bucket`*`/`*`key`*.
- **`runNumber`** – Die Zahl der Ausführungen dieses Ladevorgangs oder Feeds. Diese wird um eins erhöht, wenn der Ladevorgang neu gestartet wird.

  Typ: unsigned long.
- **`retryNumber`** – Die Zahl der Wiederholungen dieses Ladevorgangs oder Feeds. Diese wird um eins erhöht, wenn der Loader einen Feed oder Ladevorgang automatisch wiederholt.

  Typ: unsigned long.
- **`status`** – Der zurückgegebene Status des Ladevorgangs oder Feeds. `LOAD_COMPLETED` gibt einen erfolgreichen Ladevorgang ohne Probleme an. Eine Liste weiterer Ladestatusmeldungen finden Sie unter Neptune Loader-Fehler- und Feed-Nachrichten.

  Typ: Zeichenfolge.
- **`totalTimeSpent`** – Die Zeit in Sekunden, die für das Analysieren und Einfügen von Daten für den Ladevorgang oder Feed verbraucht wurde. Dabei wird nicht die Zeit für das Abrufen der Liste der Quelldateien berücksichtigt.

  Typ: unsigned long.
- **`totalRecords`** – Die Gesamtzahl der Datensätze, die geladen wurden oder deren Laden versucht wurde.

Typ: unsigned long.

Beachten Sie, dass sich die Zahl der Datensätze beim Laden aus einer CSV-Datei nicht auf die Anzahl der geladenen Zeilen bezieht, sondern auf die Anzahl der einzelnen Datensätze in diesen Zeilen. Betrachten Sie beispielsweise eine winzige CSV-Datei wie diese:

```
~id,~label,name,team
'P-1','Player','Stokes','England'
```

Neptune würde annehmen, dass diese Datei 3 Datensätze enthält:

```
P-1  label Player
P-1  name  Stokes
P-1  team  England
```

- **`totalDuplicates`** – Die Anzahl der gefundenen duplizierten Datensätze.

  Typ: unsigned long.

  Wie im Fall von `totalRecords` enthält dieser Wert die Anzahl der einzelnen duplizierten Datensätze in einer CSV-Datei, nicht die Anzahl der duplizierten Zeilen. Betrachten Sie beispielsweise diese kleine CSV-Datei:

  ```
  ~id,~label,name,team
  P-2,Player,Kohli,India
  P-2,Player,Kohli,India
  ```

  Der nach dem Laden zurückgegebene Status würde wie folgt aussehen und insgesamt 6 Datensätze melden, von denen 3 Duplikate sind:

  ```
  {
    "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
    }
  }
}
```

Bei openCypher-Ladevorgängen wird ein Duplikat gezählt, wenn:

- Der Loader erkennt, dass eine Zeile in einer Knotendatei eine ID ohne ID-Bereich hat, die mit einem anderen ID-Wert ohne ID-Bereich identisch ist, entweder in einer anderen Zeile oder als Teil eines vorhandenen Knotens.
- Der Loader erkennt, dass eine Zeile in einer Knotendatei eine ID mit ID-Bereich hat, die mit einem anderen ID-Wert mit ID-Bereich identisch ist, entweder in einer anderen Zeile oder als Teil eines vorhandenen Knotens.

Siehe Besondere Überlegungen beim Laden von openCypher-Daten.

- **`parsingErrors`** – Die Anzahl der aufgetretenen Analysefehler.

  Typ: unsigned long.

- **`datatypeMismatchErrors`** – Die Anzahl der Datensätze mit einem Datentyp, der nicht mit den angegebenen Daten übereinstimmt.

  Typ: unsigned long.

- **`insertErrors`** – Die Anzahl der Datensätze, die aufgrund von Fehlern nicht eingefügt werden konnten.

  Typ: unsigned long.

Neptune Get-Status **`Loader-Fehlerantwortobjekt`**

Fehler lassen sich in die folgenden Kategorien einteilen:

- **`Error 400`** – Eine ungültige `loadId` gibt den HTTP-Fehler `400` für ungültige Anforderungen zurück. Die Nachricht beschreibt den Fehler.
- **`Error 500`** – Eine gültige Anforderung, die nicht verarbeitet werden kann, gibt den internen HTTP-Serverfehler `500` zurück. Die Nachricht beschreibt den Fehler.

Die Liste der Fehler- und Feed-Meldungen, die der Loader im Fehlerfall zurückgibt, finden Sie unter Neptune Loader-Fehler- und Feed-Nachrichten.

Wenn ein Fehler auftritt, wird das JSON-Objekt `errors` im `BODY` der Antwort mit den folgenden Feldern zurückgegeben:

- **`startIndex`** – Der Index des ersten enthaltenen Fehlers.

  Typ: unsigned long.
- **`endIndex`** – Der Index des letzten enthaltenen Fehlers.

  Typ: unsigned long.
- **`loadId`** – Die ID des Ladeauftrags. Sie können diese ID verwenden, um die Fehler für den Ladevorgang zu drucken, indem Sie den `errors`-Parameter auf `TRUE` festlegen.

  Typ: Zeichenfolge.
- **`errorLogs`** – Eine Liste der Fehler.

  Typ: Liste.

Neptune Loader Get-Status **`ErrorLogs-Antwortobjekt`**

Das `errorLogs` Objekt unter `errors` der Get-Status Loader-Antwort enthält ein Objekt, das jeden Fehler anhand der folgenden Felder beschreibt:

- **`errorCode`** – Identifiziert die Art des Fehlers.

  Dabei kann es sich um einen der folgenden Werte handeln:
  - `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` (wird zurückgegeben, wenn der Massen-Loader die Art des Fehlers nicht bestimmen kann).
- **`errorMessage`** – Eine Meldung mit einer Beschreibung des Fehlers.

  Dabei kann es sich um eine generische Meldung handeln, die mit dem Fehlercode verknüpft ist, oder um eine spezifische Meldung mit Details, beispielsweise zu einem fehlenden from/to Eckpunkt oder zu einem Analysefehler.
- **`fileName`** – Der Name des Feeds.
- **`recordNum`** – Bei einem Analysefehler ist dies die Nummer des Datensatzes in der Datensatzdatei, der nicht analysiert werden konnte. Sie ist auf Null festgelegt, wenn die Datensatznummer nicht auf den Fehler zutrifft oder wenn sie nicht bestimmt werden konnte.

Beispielsweise würde der Massen-Loader einen Analysefehler generieren, wenn er in einer RDF-`nquads`-Datei eine fehlerhafte Zeile wie die folgende finden würde:

```
<http://base#subject> |http://base#predicate> <http://base#true> .
```

Wie Sie sehen können, sollte dem zweiten `http` in der Zeile oben eher ein `<` als ein `|` vorangestellt werden. Das resultierende Fehlerobjekt unter `errorLogs` in einer Statusantwort würde wie folgt aussehen:

```
{
    "errorCode" : "PARSING_ERROR",
    "errorMessage" : "Expected '<', found: |",
    "fileName" : "s3://bucket/key",
    "recordNum" : 12345
},
```

Neptune Loader-Beispiele Get-Status

Die folgenden Beispiele zeigen die Verwendung der GET-Status API des Neptune Loaders, mit der Sie Informationen über den Status Ihrer Datenladungen in die Amazon Neptune Graph-Datenbank abrufen können. Diese Beispiele decken drei Hauptszenarien ab: das Abrufen des Status

einer bestimmten Ladung, das Auflisten der verfügbaren Lade-IDs und das Anfordern detaillierter Statusinformationen für eine bestimmte Ladung.

Beispiel für eine Ladestatusanforderung

Dies ist eine Anforderung, die über HTTP `GET` mit dem Befehl `curl` gesendet wird.

AWS CLI

```
aws neptunedata get-loader-job-status \
  --endpoint-url https://your-neptune-endpoint:port \
  --load-id loadId (a UUID)
```

Weitere Informationen finden Sie unter get-loader-job-status in der Befehlsreferenz. 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**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

curl

```
curl -X GET 'https://your-neptune-endpoint:port/loader/loadId (a UUID)'
```

Example Antwort

```
{
    "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
        }
    }
}
```

Beispiel für eine loadIds-Anforderung

Dies ist eine Anforderung, die über HTTP `GET` mit dem Befehl `curl` gesendet wird.

AWS CLI

```
aws neptunedata list-loader-jobs \
  --endpoint-url https://your-neptune-endpoint:port \
  --limit 3
```

Weitere Informationen finden Sie unter list-loader-jobs in der Befehlsreferenz. 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**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

curl

```
curl -X GET 'https://your-neptune-endpoint:port/loader?limit=3'
```

Example Antwort

```
{
    "status" : "200 OK",
    "payload" : {
         "loadIds" : [
            "a2c0ce44-a44b-4517-8cd4-1dc144a8e5b5",
            "09683a01-6f37-4774-bb1b-5620d87f1931",
            "58085eb8-ceb4-4029-a3dc-3840969826b9"
        ]
    }
}
```

Beispiel für eine Detailstatusanforderung

Dies ist eine Anforderung, die über HTTP `GET` mit dem Befehl `curl` gesendet wird.

AWS CLI

```
aws neptunedata get-loader-job-status \
  --endpoint-url https://your-neptune-endpoint:port \
  --load-id loadId (a UUID) \
  --details
```

Weitere Informationen finden Sie unter get-loader-job-status in der Befehlsreferenz. 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**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

curl

```
curl -X GET 'https://your-neptune-endpoint:port/loader/loadId (a UUID)?details=true'
```

Example Antwort


```
{
    "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
        }
    }
}
```

Beispiele für Neptune Loader ErrorLogs Get-Status

Die folgenden Beispiele zeigen die detaillierte Statusantwort des Neptune-Loaders, wenn beim Laden der Daten Fehler aufgetreten sind. Die Beispiele veranschaulichen die Struktur der Antwort, einschließlich Informationen über fehlgeschlagene Feeds, den Gesamtstatus und detaillierte Fehlerprotokolle.

Beispiel für eine Detailstatusantwort bei Fehlern

Dies ist eine Anforderung, die über HTTP `GET` mit `curl` gesendet wird:

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
```

Weitere Informationen finden Sie unter get-loader-job-status in der Befehlsreferenz. 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**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

curl

```
curl -X GET 'https://your-neptune-endpoint:port/loader/0a237328-afd5-4574-a0bc-c29ce5f54802?details=true&errors=true&page=1&errorsPerPage=3'
```

Example Beispiel für eine Detailantwort bei Fehlern

Dies ist ein Beispiel für die Antwort, die Sie möglicherweise für die Abfrage oben erhalten, wobei das Objekt `errorLogs` die aufgetretenen Ladefehler auflistet:

```
{
    "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
        }
    }
}
```

Beispiel für einen Fehler, bei dem die **`Aufgabe beim Prefetch unterbrochen wurde`**

Wenn Sie den Status `LOAD_FAILED` erhalten und anschließend detailliertere Informationen anfordern, kann als Fehler `PARSING_ERROR` mit der Meldung `Data prefetch task interrupted` zurückgegeben werden:

```
"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
    }
]
```

Dieser Fehler tritt bei einer temporären Unterbrechung des Datenladevorgangs auf, die in der Regel nicht durch Ihre Anforderung oder Ihre Daten verursacht wurde. Er kann in der Regel behoben werden, indem Sie einfach die Massen-Upload-Anforderung wiederholen. Wenn Sie Standardeinstellungen verwenden, nämlich `"mode":"AUTO"` und `"failOnError":"TRUE"`, überspringt der Loader Dateien, die er bereits erfolgreich geladen hat und fährt mit dem Laden von Dateien fort, die noch nicht geladen wurden, als die Unterbrechung stattfand.

Neptune-Loader – Auftrag abbrechen

Bricht einen Ladeauftrag ab.

Zum Abbrechen eines Auftrags müssen Sie eine HTTP-DELETE-Anforderung an den `https://your-neptune-endpoint:port/loader`-Endpunkt senden. Die `loadId` kann dem `/loader`-URL-Pfad vorangestellt werden oder als Variable in der URL eingeschlossen werden.

Anforderungssyntax – Auftrag abbrechen

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

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

Anforderungsparameter – Auftrag abbrechen

loadId

Die ID des Ladeauftrags.

Antwortsyntax– Auftrag abbrechen

```
no response body
```

200 OK

Bei erfolgreicher Löschung eines Ladeauftrags wird der Code `200` zurückgegeben.

Fehler– Auftrag abbrechen

Wenn ein Fehler auftritt, wird ein JSON-Objekt im `BODY` der Antwort zurückgegeben. Das `message`-Objekt enthält eine Beschreibung des Fehlers.

Fehlerkategorien

- **`Error 400`** – Eine ungültige `loadId` gibt den HTTP-Fehler `400` für ungültige Anforderungen zurück. Die Nachricht beschreibt den Fehler.
- **`Error 500`** – Eine gültige Anforderung, die nicht verarbeitet werden kann, gibt den internen HTTP-Serverfehler `500` zurück. Die Nachricht beschreibt den Fehler.

Fehlermeldungen – Auftrag abbrechen

Im Folgenden sind mögliche Fehlermeldungen der Abbruch-API mit einer Beschreibung des jeweiligen Fehlers aufgeführt.

- `The load with id = load_id does not exist or not active` (HTTP 404) – Der Ladevorgang wurde nicht gefunden. Prüfen Sie den Wert des `id`-Parameters.
- `Load cancellation is not permitted on a read replica instance.` (HTTP 405) – Laden ist ein Schreibvorgang. Versuchen Sie erneut, den read/write Cluster-Endpunkt zu laden.

Beispiele – Auftrag stornieren

Example Anforderung

Dies ist eine Anforderung, die über HTTP `DELETE` mit dem Befehl `curl` gesendet wird.

AWS CLI

```
aws neptunedata cancel-loader-job \
  --endpoint-url https://your-neptune-endpoint:port \
  --load-id 0a237328-afd5-4574-a0bc-c29ce5f54802
```

Weitere Informationen finden Sie unter cancel-loader-job in der Befehlsreferenz. 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**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

curl

```
curl -X DELETE 'https://your-neptune-endpoint:port/loader/0a237328-afd5-4574-a0bc-
c29ce5f54802'
```

Verwenden AWS Database Migration Service um Daten aus einem anderen Datenspeicher in Amazon Neptune zu laden

AWS Database Migration Service (AWS DMS) kann Daten aus unterstützten Quelldatenbanken schnell und sicher in Neptune laden. Die Quelldatenbank bleibt während der Migration voll betriebsbereit, wodurch die Ausfallzeiten für Anwendungen, die von ihr abhängig sind, minimiert werden.

Detaillierte Informationen dazu finden Sie AWS DMS im AWS Database Migration Service Benutzerhandbuch und in der AWS Database Migration Service API-Referenz. In Verwenden von Neptune als Ziel für AWS Database Migration Service erfahren Sie, wie Sie einen Neptune-Cluster als Ziel für die Migration einrichten können.

Dies sind einige Voraussetzungen für den Import von Daten in Neptune mit AWS DMS:

- Sie müssen ein AWS DMS Tabellenzuordnungsobjekt erstellen, um zu definieren, wie Daten aus der Quelldatenbank extrahiert werden sollen (weitere Informationen finden Sie unter Angabe der Tabellenauswahl und der Transformationen durch Tabellenzuordnung mithilfe von JSON im

AWS DMS Benutzerhandbuch). Dieses Tabellenzuweisungs-Konfigurationsobjekt gibt an, welche Tabellen in welcher Reihenfolge gelesen werden sollen und wie ihre Spalten benannt werden. Es kann auch die zu kopierenden Zeilen filtern und einfache Werttransformationen bereitstellen, wie zum Beispiel das Konvertieren in Kleinbuchstaben oder Rundungen.

- Sie müssen eine `GraphMappingConfig` in Neptune erstellen, um anzugeben, wie die aus der Quelldatenbank extrahierten Daten in Neptune geladen werden sollen. Für RDF-Daten (mit SPARQL abgefragt) wird `GraphMappingConfig` in der Standard-Zuweisungssprache R2RM von W3 geschrieben. Bei Eigenschaftsdiagrammdaten (abgefragt mit Gremlin) handelt es sich bei `GraphMappingConfig` um ein JSON-Objekt, das unter GraphMappingConfig Layout für Property-Graph/Gremlin Daten beschrieben wird.
- Sie müssen verwenden AWS DMS , um eine Replikationsinstanz in derselben VPC wie Ihr Neptune-DB-Cluster zu erstellen, um die Datenübertragung zu vermitteln.
- Außerdem benötigen Sie einen Amazon–S3-Bucket als Zwischenspeicher für die Bereitstellung der Migrationsdaten.

Einen Neptune erschaffen GraphMappingConfig

Die von Ihnen erstellte `GraphMappingConfig` gibt an, wie aus einem Quelldatenspeicher extrahierte Daten in einen Neptune-DB-Cluster geladen werden sollen. Das Format ist davon abhängig, ob sie zum Laden von RDF-Daten oder zum Laden von Eigenschaftsdiagrammdaten verwendet wird.

Für RDF-Daten können Sie die W3-Sprache R2RML verwenden, um relationale Daten zu RDF zuzuordnen.

Wenn Sie Eigenschaftsdiagrammdaten laden, die mit Gremlin abgefragt werden sollen, erstellen Sie ein JSON-Objekt für `GraphMappingConfig`.

GraphMappingConfig Layout für Daten RDF/SPARQL

Wenn Sie RDF-Daten laden, die mit SPARQL abgefragt werden sollen, schreiben Sie die `GraphMappingConfig` in R2RML. `R2RML` ist eine Standard-W3-Sprache für die Zuweisung relationaler Daten zu RDF. Hier ist ein Beispiel:

```
@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" ];
    ] .
```

Im Folgenden ein weiteres Beispiel:

```
@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" ]
    ] .
```

Die W3-Empfehlung unter R2RML: RDB to RDF Mapping Language enthält Details zu dieser Sprache.

GraphMappingConfig Layout für Property-Graph/Gremlin Daten

Ein vergleichbarer `GraphMappingConfig` für Eigenschaftsdiagrammdaten ist ein JSON-Objekt, das eine Zuweisungsregel für jede aus den Quelldaten generierte Diagrammentität bereitstellt. Die folgende Vorlage zeigt, wie die einzelnen Regeln in diesem Objekt aussehen:

```
{
  "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)"
            }
          ]
```

```
        }
      ]
    }
  ]
}
```

Beachten Sie, dass das Vorhandensein einer Vertex-Beschriftung impliziert, dass die Vertex hier erstellt wird, während ihre Abwesenheit impliziert, dass die Vertex von einer anderen Quelle erstellt wird und diese Definition nur Vertexeigenschaften hinzufügt.

Hier ist eine Beispielregel für einen Mitarbeiterdatensatz:

```
{
  "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"
            }
          ]
        }
      ]
    }
  ]
}
```

Erstellen eines AWS DMS Replikationsaufgabe mit Neptune als Ziel

Nach der Erstellung der Konfigurationen für die Tabellen- und Diagrammzuordnung verwenden Sie den folgenden Prozess, um Daten aus dem Quellspeicher in Neptune zu laden. Weitere Informationen zu den fraglichen APIs finden Sie in der AWS DMS Dokumentation.

Erstellen Sie eine AWS DMS replication instance

Erstellen Sie eine AWS DMS Replikationsinstanz in der VPC, in der Ihr Neptune-DB-Cluster läuft (siehe Arbeiten mit einer AWS DMS-Replikationsinstanz und CreateReplicationInstanceim AWS DMS Benutzerhandbuch). Dazu können Sie einen AWS CLI Befehl wie den folgenden verwenden:

```
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)
```

Erstellen Sie ein AWS DMS Endpunkt für die Quelldatenbank

Der nächste Schritt besteht darin, einen AWS DMS Endpunkt für Ihren Quelldatenspeicher zu erstellen. Sie können die AWS DMS CreateEndpointAPI AWS CLI wie folgt verwenden:

```
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)
```

Richten Sie einen Amazon S3 S3-Bucket für Neptune ein, der für das Staging-Daten verwendet werden soll

Wenn Sie keinen Amazon S3 S3-Bucket haben, den Sie für das Staging von Daten verwenden können, erstellen Sie einen, wie unter Erstellen eines Buckets im Amazon S3 Getting-Started S3-Handbuch oder Wie erstelle ich einen S3-Bucket? beschrieben. im Konsolen-Benutzerhandbuch.

Sie müssen eine IAM-Richtlinie erstellen, die `GetObject`, `PutObject`, `DeleteObject` und `ListObject` Berechtigungen für den Bucket gewährt, wenn noch nicht geschehen.

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/*"
      ]
    }
  ]
}
```

Wenn für Ihren Neptune-DB-Cluster die IAM-Authentifizierung aktiviert ist, müssen Sie außerdem die folgende Richtlinie einschließen:

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/
*"
    }
  ]
}
```

Erstellen Sie eine IAM-Rolle als Vertrauensdokument, an das Sie die Richtlinie anfügen können:

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"
    }
  ]
}
```

Nachdem Sie der Rolle die Richtlinie angefügt haben, fügen Sie die Rolle Ihrem Neptune-DB-Cluster an. Auf diese Weise können AWS DMS Sie den Bucket für das Staging der geladenen Daten verwenden.

Erstellen Sie einen Amazon S3 S3-Endpunkt in der Neptune VPC

Erstellen Sie jetzt einen VPC-Gateway-Endpunkt für Ihren Amazon-S3-Zwischen-Bucket in der VPC, in der sich Ihr Neptune-Cluster befindet. Sie können AWS CLI dazu das AWS-Managementkonsole oder das verwenden, wie unter Einen Gateway-Endpunkt erstellen beschrieben.

Erstellen Sie ein AWS DMS Zielendpunkt für Neptune

Erstellen Sie einen AWS DMS Endpunkt für Ihren Neptune-DB-Zielcluster. Sie können die AWS DMS CreateEndpointAPI mit dem folgenden `NeptuneSettings` Parameter verwenden:

```
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) }'
```

Das JSON-Objekt, das in seinem `NeptuneSettings` Parameter an die AWS DMS `CreateEndpoint` API übergeben wurde, hat die folgenden Felder:

- **`ServiceAccessRoleArn`** – (erforderlich) Der ARN einer IAM-Rolle, die einen detaillierten Zugriff auf den S3-Bucket ermöglicht, der für die Migration der Daten zu Neptune verwendet wird. Diese Rolle sollte auch Berechtigungen für den Zugriff auf Ihren Neptune-DB-Cluster haben, wenn die IAM-Autorisierung für ihn aktiviert ist.
- **`S3BucketName`** – (erforderlich) Für eine Volllast-Migration konvertiert die Replikations-Instance alle RDS-Daten in CSV- und Quad-Dateien und lädt sie zu diesem S3-Staging-Bucket und anschließend per Massenladevorgang zu Neptune hoch.
- **`S3BucketFolder`** – (erforderlich) Der Ordner, der im S3-Staging-Bucket verwendet werden soll.
- **`ErrorRetryDuration`** – (optional) Die Anzahl der Millisekunden, die nach dem Fehlschlag einer Neptune-Anforderung gewartet werden muss, bevor sie wiederholt werden kann. Der Standardwert ist 250.
- **`MaxRetryCount`**— (optional) Die maximale Anzahl von Wiederholungsanfragen, die nach einem wiederholten Fehler gestellt werden AWS DMS sollten. Der Standardwert ist 5.
- **`MaxFileSize`** – (optional) Die maximale Größe in Byte für jede Staging-Datei, die während der Migration in S3 gespeichert wurde. Der Standardwert ist 1.048.576 KB (1 GB)
- **`IsIAMAuthEnabled`** (optional)`true` Auf festgelegt, wenn IAM-Authentifizierung für den Neptune-DB-Cluster aktiviert ist, andernfalls auf `false`. Der Standardwert ist `false`.

Testen Sie die Verbindungen zu den neuen Endpunkten

Sie können die Verbindung zu jedem dieser neuen Endpunkte mithilfe der AWS DMS TestConnectionAPI wie folgt testen:

```
aws dms test-connection \
    --replication-instance-arn (the ARN of the replication instance) \
    --endpoint-arn (the ARN of the endpoint you are testing)
```

Erstellen Sie eine AWS DMS Replikationsaufgabe

Nachdem Sie die vorherigen Schritte erfolgreich abgeschlossen haben, erstellen Sie eine Replikationsaufgabe für die Migration von Daten aus Ihrem Quelldatenspeicher nach Neptune, indem Sie die AWS DMS CreateReplicationTaskAPI wie folgt verwenden:

```
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')
```

Der Parameter `TaskData` stellt die GraphMappingConfig bereit, die angibt, wie die kopierten Daten in Neptune gespeichert werden sollen.

Starten Sie das AWS DMS Replikationsaufgabe

Jetzt können Sie die Replikationsaufgabe starten:

```
aws dms start-replication-task
    --replication-task-arn (ARN of the replication task started in the previous step)
    --start-replication-task-type start-replication
```

Daten mithilfe von Abfragen in Amazon Neptune laden

Neptune unterstützt das direkte Schreiben von Daten über Abfragesprachenoperationen. Sie können Standard-Schreiboperationen wie `CREATE` und `MERGE` in OpenCypher, in SPARQL oder `INSERT`

in Gremlin verwenden, um Daten `mergeE()` in Ihrem Diagramm hinzuzufügen oder `mergeV()` zu ändern. Diese Operationen eignen sich für inkrementelle Aktualisierungen und transaktionale Schreibvorgänge.

Verwenden Sie zum Laden von Daten aus Amazon S3 den Verwenden des Amazon Neptune Bulk Loaders zur Datenaufnahme für große Datensätze, die eine optimierte Leistung erfordern. Für kleinere Datensätze in einer oder mehreren Amazon S3-Dateien können Sie abfragebasierte Ladefunktionen verwenden, um Daten direkt in Ihren Abfragen zu lesen und zu verarbeiten.

Die folgenden abfragebasierten Ladefunktionen sind verfügbar:

OpenCypher: neptune.read ()

Die `neptune.read()` Funktion liest CSV- oder Parquet-Dateien aus Amazon S3 innerhalb einer `CALL` Unterabfrage, sodass Sie Daten zur Abfragezeit verarbeiten und laden können.

```
CALL neptune.read({
  source: "s3://bucket/data.csv",
  format: "csv"
})
YIELD row
CREATE (n:Person {id: row.id, name: row.name})
```

Eine vollständige Dokumentation finden Sie unterneptune.read ().

SPARQL: LADEN und ENTLADEN

`LOAD`SPARQL-Operationen importieren RDF-Daten von einer URI in ein benanntes Diagramm. `UNLOAD`exportiert Daten aus einem Diagramm nach Amazon S3.

```
LOAD <s3://bucket/data.ttl> INTO GRAPH <http://example.org/graph>
```

Eine vollständige Dokumentation finden Sie unterVerwenden von SPARQL UPDATE LOAD zum Importieren von Daten in Neptune.

Gremlin: io () -Schritt

Sie können Gremlins `g.io(URL).read()` Step auch verwenden, um Datendateien in GraphML (einem XML-Format), GraphSon (einem JSON-Format) und anderen Formaten einzulesen.

```
g.io("s3://bucket/data.graphml").read().iterate()
```

Einzelheiten finden Sie in der DokumentationTinkerPop.

Speicher

Neptune unterstützt Dictionary Garbage Collection (GC) für Eigenschaftsgraphdaten, die über den `neptune_lab_mode` Parameter aktiviert werden kann, wenn er nicht aktiv `neptune_streams` ist. Wenn er aktiviert ist, bereinigt dieser Hintergrundjob ungenutzte Glossareinträge und reduziert so potenziell die Geschwindigkeit des Datenwachstums. Die Funktion kann in zwei Modi ausgeführt werden: soft_delete (Markierung von Einträgen als gelöscht, ohne sie explizit zu entfernen) und aktiviert (explizites Löschen von Einträgen). Der GC-Prozess kann die Systemleistung beeinträchtigen, indem er mit Abfrage-Threads für Ressourcen wie CPU und Puffercache zu kämpfen hat. Er kann mit einer maximalen Parallelität von 16 Threads ausgeführt werden.

Neptune unterstützt auch vom Server generierte Inline-Edge-IDs, die über einen Konfigurationsparameter aktiviert werden können, wenn neptune_streams nicht aktiv ist. Wenn diese Funktion aktiviert ist, generiert der Server eindeutige Inline-IDs für Kanten, die keine benutzerdefinierte ID haben. Dabei wird das reservierte Präfix „neptune_reserved“ verwendet. Diese Inline-IDs werden nicht im Wörterbuch gespeichert, wodurch die Speichereffizienz verbessert werden kann.

Themen

- Müllabfuhr im Neptun-Wörterbuch
- Neptune-integrierte servergenerierte Edge-ID

Müllabfuhr im Neptun-Wörterbuch

Neptune unterstützt die Dictionary Garbage Collection (GC), die über den `neptune_lab_mode` Parameter für Eigenschaftsdiagrammdaten aktiviert werden kann. Es kann für Cluster aktiviert werden, die nur Eigenschaftsdiagrammdaten enthalten, wenn `neptune_streams` es nicht aktiviert ist. Die Funktion wird automatisch deaktiviert, wenn sie aktiviert `neptune_streams` ist oder `neptune_streams` Daten noch nicht abgelaufen sind. Für die Aktivierung der Funktion ist ein Neustart der Writer-Instanz erforderlich. Diese Funktion ist ab Engine-Version 1.4.3.0 verfügbar.

Wenn diese Option aktiviert ist, werden die ungenutzten Glossareinträge durch einen Hintergrundjob bereinigt. Es reduziert nicht`VolumeBytesUsed`, sondern gibt Platz im Index für neue Einfügungen frei. Die Wachstumsrate von `VolumeBytesUsed` ist wahrscheinlich geringer, wenn die Wörterbuch-GC aktiviert ist, als wenn sie nicht aktiviert ist.

Die Wörterbuchbereinigung läuft im Hintergrund und scannt alle Diagramm- und Wörterbuchdaten, um Begriffe zu finden, die nicht verwendet werden. Ein neuer Lauf wird beim Start ausgelöst, sobald sich etwa 6% der Daten geändert haben. GC konkurriert mit Abfrage-Threads um Serverressourcen wie CPU, Puffercache, Undo-Log-Generierung und I/O Schreibvorgänge, wodurch der Abfragedurchsatz potenziell reduziert wird. Da GC Daten scannt, die nicht aktiv von Abfragen betroffen sind, kann sich dies auf den Puffercache auf dem Writer-Knoten auswirken. Wenn GC neue Löschungen durchführt, kann es im Cluster zu zusätzlichen I/O Schreibvorgängen kommen und es müssen mehr Undo-Logs gelöscht werden, was auch zu höheren Werten für die `UndoLogListSize` Metrik führen kann.

GC kann in zwei Modi ausgeführt werden, und`soft_delete`. `enabled` Bei der Ausführung im `soft_delete` Modus werden unbenutzte Glossareinträge als gelöscht markiert (soft_delete), aber nicht explizit gelöscht. Dieser Modus könnte auch verwendet werden, um die Leistungsmerkmale nach dem Einschalten des Hintergrundvorgangs zu verstehen. Wenn der `enabled` Modus verwendet wird, werden Einträge explizit gelöscht („hartes Löschen“). Es wird empfohlen, GC für eine gewisse Zeit im `soft_delete` Modus laufen zu lassen, bevor in den `enabled` Modus gewechselt wird.

Dictionary GC unterstützt eine maximale Parallelität von 16 (auf Computern mit 16 oder mehr Kernen). Es läuft standardmäßig mit einem einzigen Thread, kann aber auch mit höherer Parallelität ausgeführt werden, wenn es zum ersten Mal aktiviert wird. Dictionary-GC-Threads werden mit derselben Priorität wie die Abfrage-Threads ausgeführt, und sie haben auch mit Ressourcen auf dem Writer zu kämpfen.

Dictionary-GC kann über den `neptune_lab_mode` Parameter aktiviert werden, indem der `DictionaryGCMode` Schlüssel gesetzt wird. Es akzeptiert drei mögliche Werte: `disabled` (Standard)`soft_delete`, oder`enabled`. Im folgenden Codebeispiel würde beispielsweise der Wert `DictionaryGCMode` auf festgelegt werden`soft_delete`:

```
neptune_lab_mode = 'DictionaryGCMode=soft_delete'
```

Der Parallelitätsparameter,`DictionaryGCConcurrency`, ist optional und kann einen Wert zwischen 1 und 16 annehmen. Wenn er auf einen höheren Wert als das Minimum von 16 und die Anzahl der Kerne gesetzt ist, ist die Parallelität auf diesen Wert begrenzt.

```
neptune_lab_mode = 'DictionaryGCMode=soft_delete,DictionaryGCConcurrency=2'
```

Der Wörterbuch-GC-Job wird nach dem Serverstart im Hintergrund aktiviert, sobald einige Daten verfügbar sind. Der Engine-Status zeigt den aktuellen Status von Dictionary-GC an. Die unten

gezeigte Beispielausgabe zeigt, dass sich das Wörterbuch GC im `soft_delete` Modus befindet und mit einer Parallelität von 2 ausgeführt wird. Wenn der Hintergrundjob ausgeführt wird, sucht er möglicherweise aktiv nach unbenutzten Glossareinträgen und löscht sie oder er wartet auf neue Löschungen, um eine neue GC-Runde auszulösen.

```
{"status":"healthy",...,"labMode":
{"ObjectIndex":"disabled","DictionaryGC":"{Mode=enabled,Concurrency=2}"},...}
```

Dictionary GC wird angehalten, wenn eine der folgenden Bedingungen erfüllt ist:

- Aktiver Massenladevorgang.
- Der freie Speicherplatz beträgt weniger als 15 GB.
- `UndoLogListSize`ist höher als 1.000.000.

Neptune-integrierte servergenerierte Edge-ID

Neptune unterstützt Server-Generated Inline-Edge-IDs. Es kann über den Neptune-Konfigurationsparameter aktiviert werden, `neptune_enable_inline_server_generated_edge_id` wenn `neptune_streams` es nicht aktiviert ist. Diese Funktion ist ab der Engine-Version 1.4.3.0 für Gremlin-Abfragen verfügbar und wird in einer future Version für OpenCypher Abfragen verfügbar sein.

Die Edge-ID ist eine eindeutige Kennung für eine Kante. Beim Einfügen einer Kante kann eine Kanten-ID angegeben werden. Wenn keine ID angegeben wird, generiert der Server standardmäßig eine UUID-basierte ID und weist sie der Kante zu. Wie die benutzerdefinierte ID wird die vom UUID-based Server generierte ID im Wörterbuch gespeichert.

Wenn die `neptune_enable_inline_server_generated_edge_id` Funktion aktiviert ist, generiert der Server eine eindeutige Inline-ID, wenn in der Abfrage keine ID angegeben wird. Die Inline-Kanten-IDs werden nicht im Wörterbuch gespeichert, wodurch die Speichereffizienz verbessert wird. Die vom Server generierten Inline-IDs beginnen mit dem reservierten Präfix. `neptune_reserved`

> **Warning**
>
> Neptune reserviert das `'neptune_reserved'` Präfix für servergenerierte Inline-IDs. Bei Abfragen, bei denen versucht wird, Daten mit einer benutzerdefinierten ID einzufügen, die mit dem reservierten Präfix beginnt, wird ein Fehler angezeigt.

Die vom Server generierte Inline-Edge-ID-Funktion kann aktiviert werden, indem der Parameter auf Clusterebene auf gesetzt wird. `neptune_enable_inline_server_generated_edge_id` 1 Ein Neustart der Instanz ist erforderlich. Das folgende Beispiel aktiviert die servergenerierte Edge-ID-Funktion:

```
"ParameterName=neptune_enable_inline_server_generated_edge_id,ParameterValue=1,ApplyMethod=pend
reboot"
```

Um zu überprüfen, ob die Funktion aktiviert ist, können Sie die Funktionen im Engine-Status überprüfen. Diese Funktion wird automatisch deaktiviert, wenn sie aktiviert `neptune_streams` ist. Die folgende Beispielausgabe zeigt den Engine-Status für die aktivierte Funktion:

```
"features":{"InlineServerGeneratedEdgeId":"enabled"}
```

Das folgende Gremlin-Beispiel fügt eine Kante ohne benutzerdefinierte ID hinzu, wenn die vom Inline-Server generierte Edge-ID-Funktion aktiviert ist:

```
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": []
        }
    }
}
```

Abfragen eines Neptune–Diagramms

Neptune unterstützt die folgenden Diagrammabfragesprachen für den Zugriff auf ein Diagramm:

- Gremlin, von Apache TinkerPop für die Erstellung und Abfrage von Eigenschaftsgraphen definiert.

  Eine Abfrage in Gremlin ist eine Traversierung, die aus diskreten Schritten besteht. Jeder Schritt folgt einer Kante bis zu einem Knoten.

  Weitere Informationen zur Verwendung von Gremlin in Neptune finden Sie in Zugriff auf ein Neptune-Diagramm mit Gremlin. Spezifische Details zur Neptune-Implementierung von Gremlin finden Sie in Einhaltung der Gremlin-Standards in Amazon Neptune.
- openCypher ist eine deklarative Abfragesprache für Eigenschaftsdiagramme. Ursprünglich von Neo4j entwickelt, wurde sie 2015 als Open-Source-Software veröffentlicht und ist unter einer Apache-2-Open-Source-Lizenz für das openCypher-Projekt verfügbar. Die Syntax ist in der openCypher-Spezifikation dokumentiert.
- SPARQL ist eine deklarative Sprache zum Abfragen von RDF-Daten, die auf dem Musterabgleich von Diagrammen basiert. Sie wird vom World Wide Web Consortium unterstützt.

  Weitere Informationen zur Verwendung von SPARQL in Neptune finden Sie in Zugriff auf das Neptune-Diagramm mit SPARQL. Spezifische Details zur Neptune-Implementierung von SPARQL finden Sie in Einhaltung von SPARQL-Standards in Amazon Neptune.

> **Note**
>
> Sowohl Gremlin als auch openCypher können verwendet werden, um alle in Neptune gespeicherten Eigenschaftsdiagrammdaten abzufragen, unabhängig davon, wie sie geladen wurden.

Themen

- Abfragewarteschlangen in Amazon Neptune
- Plan-Cache in Amazon Neptune abfragen
- Einfügen einer benutzerdefinierten ID in eine Neptune-Gremlin- oder -SPARQL-Abfrage
- Zugriff auf ein Neptune-Diagramm mit Gremlin

- Zugriff auf das Neptun-Diagramm mit openCypher
- Zugriff auf das Neptune-Diagramm mit SPARQL

Abfragewarteschlangen in Amazon Neptune

Beim Entwickeln und Optimieren von Diagrammanwendungen kann es hilfreich sein, die Auswirkungen des Setzens von Datenbankabfragen in eine Warteschlange zu kennen. In Amazon Neptune werden Abfragen wie folgt in Warteschlangen eingereiht:

- Die maximale Anzahl von Abfragen, die pro Instance in eine Warteschlange gestellt werden können, beträgt unabhängig von der Größe der Instance 8.192. Alle über diese Zahl hinausgehenden Abfragen werden zurückgewiesen und schlagen mit einem `ThrottlingException` fehl.
- Die maximale Anzahl von Abfragen, die gleichzeitig ausgeführt werden können, hängt von der Anzahl der zugewiesenen Worker-Threads ab, die gewöhnlich auf die doppelte Anzahl der verfügbaren virtuellen CPU-Kerne (vCPUs) eingestellt ist.
- Die Abfragelatenz beinhaltet die Zeit, die eine Abfrage in der Warteschlange verbringt, sowie die zeit für das Netzwerk-Round-Tripping und die Zeit, die tatsächlich für die Ausführung benötigt wird.

Ermitteln der Anzahl der Abfragen, die sich zu einem bestimmten Zeitpunkt in Ihrer Warteschlange befinden

Die `MainRequestQueuePendingRequests` CloudWatch Metrik zeichnet die Anzahl der Anfragen, die in der Eingabewarteschlange warten, mit einer Genauigkeit von fünf Minuten auf (siehe). Neptun-Metriken CloudWatch

Sie können in Gremlin die aktuelle Anzahl von Abfragen in der Warteschlange mit dem `acceptedQueryCount`-Wert abrufen, der von Gremlin-Abfragestatus-API ausgegeben wird. Beachten Sie jedoch, dass der von SPARQL-Abfragestatus-API zurückgegebene `acceptedQueryCount`-Wert alle Abfragen enthält, die seit dem Start des Servers akzeptiert wurden, einschließlich abgeschlossener Abfragen.

Auswirkung von Abfragewarteschlangen auf Zeitüberschreitungen

Wie oben erwähnt, umfasst die Abfragelatenz die Zeit, die eine Abfrage in der Warteschlange verbringt, sowie die Zeit, die für die Ausführung benötigt wird.

Da der Zeitüberschreitungszeitraum einer Abfrage in der Regel ab dem Eintritt in die Warteschlange gemessen wird, kann eine sich langsam bewegende Warteschlange dazu führen, das viele Abfragen eine Zeitüberschreitung erleiden, sobald sie aus der Warteschlange genommen werden. Dies ist offensichtlich unerwünscht, daher ist es sinnvoll, eine große Anzahl von Abfragen in einer Warteschlange zu vermeiden, es sei denn, diese können schnell ausgeführt werden.

Plan-Cache in Amazon Neptune abfragen

Wenn eine Abfrage an Neptune gesendet wird, wird die Abfragezeichenfolge analysiert, optimiert und in einen Abfrageplan umgewandelt, der dann von der Engine ausgeführt wird. Anwendungen basieren häufig auf gängigen Abfragemustern, die mit unterschiedlichen Werten instanziiert werden. Der Abfrageplan-Cache kann die Gesamtlatenz reduzieren, indem er die Abfragepläne zwischenspeichert und so das Parsen und Optimieren für solche wiederholten Muster vermeidet.

Der Abfrageplan-Cache kann für OpenCypherAbfragen verwendet werden — sowohl für nicht parametrisierte als auch für parametrisierte Abfragen. Es ist für READ sowie für HTTP und Bolt aktiviert. Es wird für OC-Mutationsabfragen nicht unterstützt. Es wird für Gremlin- oder SPARQL-Abfragen nicht unterstützt.

Wie erzwingt man die Aktivierung oder Deaktivierung des Abfrageplan-Caches

Der Abfrageplan-Cache ist standardmäßig für parametrisierte Abfragen mit niedriger Latenz aktiviert. Ein Plan für eine parametrisierte Abfrage wird nur zwischengespeichert, wenn die Latenz unter dem Schwellenwert von 100 ms liegt. Dieses Verhalten kann pro Abfrage (parametrisiert oder nicht) durch den Abfragehinweis auf Abfrageebene außer Kraft gesetzt werden. `QUERY:PLANCACHE` Es muss zusammen mit der Klausel verwendet werden. `USING` Der Abfragehinweis akzeptiert `enabled` oder `disabled` als Wert.

AWS CLI

Erzwingen, dass der Plan zwischengespeichert oder wiederverwendet wird:

```
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"
```

Mit Parametern:

```
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}'
```

Erzwingen, dass der Plan weder zwischengespeichert noch wiederverwendet wird:

```
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"
```

Weitere Informationen finden Sie unter execute-open-cypher-query in der Befehlsreferenz. 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'])
```

AWS SDK-Beispiele in anderen Sprachen finden Sie unter. AWS SDK

awscurl

Erzwingen, dass der Plan zwischengespeichert oder wiederverwendet wird:

```
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**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

curl

Erzwingen, dass der Plan zwischengespeichert oder wiederverwendet wird:

```
curl https://your-neptune-endpoint:port/openCypher \
  -d "query=Using QUERY:PLANCACHE \"enabled\" MATCH(n) RETURN n LIMIT 1"
```

Mit Parametern:

```
curl https://your-neptune-endpoint:port/openCypher \
  -d "query=Using QUERY:PLANCACHE \"enabled\" RETURN \$arg" \
  -d "parameters={\"arg\": 123}"
```

Erzwingen, dass der Plan weder zwischengespeichert noch wiederverwendet wird:

```
curl https://your-neptune-endpoint:port/openCypher \
  -d "query=Using QUERY:PLANCACHE \"disabled\" MATCH(n) RETURN n LIMIT 1"
```

Wie kann man feststellen, ob ein Plan zwischengespeichert ist oder nicht

Wenn bei HTTP READ die Anfrage eingereicht und der Plan zwischengespeichert wurde, `explain` würden Details angezeigt, die für den Abfrageplan-Cache relevant sind.

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
```

Weitere Informationen finden Sie unter execute-open-cypher-explain-query in der Befehlsreferenz. 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'))
```

AWS SDK-Beispiele in anderen Sprachen finden Sie unter. AWS 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**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

curl

```
curl https://your-neptune-endpoint:port/openCypher \
```

```
  -d "query=Using QUERY:PLANCACHE \"enabled\" MATCH(n) RETURN n LIMIT 1" \
  -d "explain=details"
```

Wenn der Plan zwischengespeichert wurde, zeigt die `explain` Ausgabe:

```
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).
```

Bei Verwendung von Bolt wird die Erklärungsfunktion nicht unterstützt.

Räumung

Ein Abfrageplan wird aufgrund der Gültigkeitsdauer des Caches (TTL) gelöscht oder wenn eine maximale Anzahl von zwischengespeicherten Abfrageplänen erreicht wurde. Wenn der Abfrageplan erreicht wird, wird die TTL aktualisiert. Die Standardwerte sind:

- 1000 — Die maximale Anzahl von Plänen, die pro Instanz zwischengespeichert werden können.
- TTL — 300.000 Millisekunden oder 5 Minuten. Der Cache-Treffer startet die TTL neu und setzt sie auf 5 Minuten zurück.

Bedingungen, die dazu führen, dass der Plan nicht zwischengespeichert wird

Der Abfrageplan-Cache würde unter den folgenden Bedingungen nicht verwendet werden:

1. Wenn eine Abfrage mithilfe des Abfragehinweises gesendet wird`QUERY:PLANCACHE "disabled"`. Sie können die Abfrage erneut ausführen und entfernen, `QUERY:PLANCACHE "disabled"` um den Abfrageplan-Cache zu aktivieren.
2. Wenn es sich bei der übermittelten Abfrage nicht um eine parametrisierte Abfrage handelt und sie den Hinweis nicht enthält. `QUERY:PLANCACHE "enabled"`

3. Wenn die Abfrageauswertungszeit den Latenzschwellenwert überschreitet, wird die Abfrage nicht zwischengespeichert und als Abfrage mit langer Laufzeit betrachtet, für die der Abfrageplan-Cache nicht von Vorteil wäre.
4. Wenn die Abfrage ein Muster enthält, das keine Ergebnisse zurückgibt.
   - dh `MATCH (n:nonexistentLabel) return n` wenn es keine Knoten mit dem angegebenen Label gibt.
   - d.h. `MATCH (n {name: $param}) return n parameters={"param": "abcde"}` wenn es keine Knoten gibt, die enthaltenname=abcde.
5. Wenn der Abfrageparameter ein zusammengesetzter Typ ist, z. B. a `list` oder `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. Wenn der Abfrageparameter eine Zeichenfolge ist, die nicht Teil eines Datenlade- oder Dateneinfügevorgangs war. Wird `CREATE (n {name: "X"})` beispielsweise ausgeführt, um einzufügen`"X"`, dann `RETURN "X"` wird es zwischengespeichert, `RETURN "Y"` würde aber nicht zwischengespeichert, da es nicht eingefügt wurde und nicht in der Datenbank vorhanden `"Y"` ist.

Einfügen einer benutzerdefinierten ID in eine Neptune-Gremlin- oder -SPARQL-Abfrage

Standardmäßig weist Neptune jeder Abfrage einen eindeutigen `queryId`-Wert zu. Sie können diese ID verwenden, um Informationen zu einer laufenden Abfrage abzurufen (siehe Gremlin-Abfragestatus-API oder SPARQL-Abfragestatus-API) oder abzubrechen (siehe Gremlin-Abfrageabbruch oder SPARQL-Abfrageabbruch).

Mit Neptune können Sie auch Ihren eigenen `queryId`-Wert für eine Gremlin- oder SPARQL-Abfrage angeben, entweder im HTTP-Header oder (für eine SPARQL-Abfrage) mithilfe des `queryId`-Abfragehinweises. Die Zuweisung Ihrer eigenen `queryID` macht es einfach, eine Abfrage nachzuverfolgen, um ihren Status abzurufen oder sie abzubrechen.

Einfügen eines benutzerdefinierten **QueryID-Werts** mithilfe des HTTP-Headers

Sowohl für Gremlin als auch für SPARQL kann der HTTP-Header verwendet werden, um Ihren eigenen `queryId`-Wert in eine Abfrage einzufügen.

Gremlin-Beispiel

```
curl -XPOST https://your-neptune-endpoint:port \
    -d "{\"gremlin\": \
        \"g.V().limit(1).count()\" , \
        \"queryId\":\"4d5c4fae-aa30-41cf-9e1f-91e6b7dd6f47\"  }"
```

SPARQL-Beispiel

```
curl https://your-neptune-endpoint:port/sparql \
    -d "query=SELECT * WHERE { ?s ?p ?o } " \
       --data-urlencode \
       "queryId=4d5c4fae-aa30-41cf-9e1f-91e6b7dd6f47"
```

Einfügen eines benutzerdefinierten **QueryID-Werts** mithilfe eines SPARQL-Abfragehinweises

Hier ist ein Beispiel dafür, wie Sie den `queryId`-Abfragehinweis von SPARQL verwenden können, um einen benutzerdefinierten `queryId`-Wert in eine SPARQL-Abfrage einzufügen:

```
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}}"
```

Verwenden des **QueryID-Werts** zur Überprüfung des Abfragestatus

Gremlin-Beispiel

```
curl https://your-neptune-endpoint:port/gremlin/status \
    -d "queryId=4d5c4fae-aa30-41cf-9e1f-91e6b7dd6f47"
```

SPARQL-Beispiel

```
curl https://your-neptune-endpoint:port/sparql/status \
    -d "queryId=4d5c4fae-aa30-41cf-9e1f-91e6b7dd6f47"
```

Zugriff auf ein Neptune-Diagramm mit Gremlin

Amazon Neptune ist mit Apache TinkerPop und Gremlin kompatibel. Das bedeutet, dass Sie eine Verbindung zu einer Neptune-DB-Instance herstellen und die Gremlin Traversal Language verwenden können, um das Diagramm abzufragen (siehe The Graph in der Apache-Dokumentation). TinkerPop Informationen zu den Unterschieden bei der Neptune-Implementierung von Gremlin finden Sie unter Einhaltung von Gremlin-Standards.

Bei der Traversierung in Gremlin handelt es sich um eine Reihe verketteter Schritte. Begonnen wird an einem Vertex (oder Edge). Sie folgt dem Diagramm entlang der ausgehenden Edges der einzelnen Vertices und anschließend den ausgehenden Edges dieser Vertices. Jeder Schritt stellt in der Traversierung eine Operation dar. Weitere Informationen finden Sie in der Dokumentation unter The Traversal. TinkerPop

Unterschiedliche Neptune-Engine-Versionen unterstützen unterschiedliche Gremlin-Versionen. Sieh auf der Engine-Release-Seite der Neptune-Version nach, die du verwendest, um festzustellen, welche Gremlin-Version sie unterstützt, oder sieh in der folgenden Tabelle nach, in der die frühesten und neuesten Versionen von aufgeführt sind, die von verschiedenen Neptune-Engine-Versionen TinkerPop unterstützt werden:

Neptune-Engine-Version	TinkerPop Minimale Version	Maximale TinkerPop Version
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 Clients sind in der Regel innerhalb einer Serie abwärtskompatibel (`3.6.x`z. B. oder`3.7.x`), und obwohl sie diese Grenzen oft überschreiten können, werden in der obigen Tabelle die Versionskombinationen empfohlen, die verwendet werden sollten, um die bestmögliche Erfahrung und Kompatibilität zu erzielen. Sofern nicht anders empfohlen, ist es generell am besten, diese Richtlinien einzuhalten und die Client-Anwendungen so zu aktualisieren, dass sie der Version entsprechen, die TinkerPop Sie verwenden.

Bei der Aktualisierung TinkerPop von Versionen ist es immer wichtig, die Upgrade-Dokumentation zu TinkerPop lesen, die Ihnen hilft, neue Funktionen zu finden, die Sie nutzen können, aber auch Probleme, auf die Sie bei der Aktualisierung achten müssen. In der Regel sollten Sie davon ausgehen, dass vorhandene Abfragen und Funktionen nach dem Upgrade funktionieren, sofern nicht ein bestimmtes Problem als zu berücksichtigendes Problem bezeichnet wird. Schließlich ist es wichtig zu beachten, dass Sie eine Version, die Sie aktualisieren, um über eine neue Funktion zu verfügen, möglicherweise nicht verwenden können, wenn sie aus einer Version stammt, die später ist als die, die Neptune unterstützt.

Es gibt Gremlin-Sprachvarianten sowie eine Unterstützung für den Gremlin-Zugriff in verschiedenen Programmiersprachen. Weitere Informationen finden Sie in der Dokumentation unter Zu Gremlin-Sprachvarianten. TinkerPop

Diese Dokumentation beschreibt, wie Sie mit den folgenden Varianten und Programmiersprachen auf Neptune zugreifen können:

- Einrichten der Gremlin-Konsole zum Herstellen einer Verbindung mit einer Neptune-DB-Instance
- Herstellen einer Verbindung mit einer Neptune-DB-Instance über den HTTPS-REST-Endpunkt
- Java-based Gremlin-Clients zur Verwendung mit Amazon Neptune
- Herstellen einer Verbindung mit einer Neptune-DB-Instance über Python
- Herstellen einer Verbindung mit einer Neptune-DB-Instance über .NET
- Wird verwendet Node.js , um eine Verbindung zu einer Neptune-DB-Instance herzustellen
- Herstellen einer Verbindung mit einer Neptune-DB-Instance über Go

Wie unter beschriebenVerschlüsseln von Verbindungen zu Ihrer Amazon Neptune Neptune-Datenbank mit SSL/HTTPS, müssen Sie Transport Layer Security/Secure Sockets Layer (TLS/SSL) verwenden, wenn Sie in allen AWS Regionen eine Verbindung zu Neptune herstellen.

Sie benötigen Folgendes, um starten zu können:

- Eine Neptune-DB-Instance. Informationen zum Erstellen einer Neptune-DB-Instance finden Sie unter Einen Amazon Neptune Neptune-Cluster erstellen.
- Eine Amazon–EC2-Instance in derselben Virtual Private Cloud (VPC), in der sich auch Ihre Neptune-DB-Instance befindet.

Weitere Informationen zum Laden von Daten in Neptune, einschließlich Voraussetzungen, Formaten und Parametern finden Sie unter Laden von Daten in Amazon Neptune.

Themen

- Einrichten der Gremlin-Konsole zum Herstellen einer Verbindung mit einer Neptune-DB-Instance
- Herstellen einer Verbindung mit einer Neptune-DB-Instance über den HTTPS-REST-Endpunkt
- Java-based Gremlin-Clients zur Verwendung mit Amazon Neptune
- Herstellen einer Verbindung mit einer Neptune-DB-Instance über Python
- Herstellen einer Verbindung mit einer Neptune-DB-Instance über .NET
- Wird verwendet Node.js , um eine Verbindung zu einer Neptune-DB-Instance herzustellen
- Herstellen einer Verbindung mit einer Neptune-DB-Instance über Go
- Verwendung der AWS SDK zum Ausführen von Gremlin-Abfragen
- Gremlin-Abfragehinweise
- Gremlin-Abfragestatus-API
- Gremlin-Abfrageabbruch
- Unterstützung für skriptbasierte Gremlin-Sitzungen
- Gremlin-Transaktionen in Neptune
- Streaming-Abfrageergebnisse mit Gremlin
- Verwenden der Gremlin-API mit Amazon Neptune
- Zwischenspeichern von Abfrageergebnissen in Amazon Neptune Gremlin
- Mit den Schritten Gremlin mergeV () und mergeE () effiziente Upserts erstellen
- Effiziente Gremlin-Upserts mit fold () /coalesce () /unfold () erstellen
- Analysieren der Ausführung Neptune Neptun-Abfragen mit Gremlin Explain
- Verwenden von Gremlin mit der Neptune-DFE-Abfrage-Engine

Einrichten der Gremlin-Konsole zum Herstellen einer Verbindung mit einer Neptune-DB-Instance

Die Gremlin-Konsole ermöglicht es Ihnen, mit TinkerPop Graphen und Abfragen in einer REPL-Umgebung (Read-Eval-Print Loop) zu experimentieren.

Installieren der Gremlin-Konsole und Herstellen üblicher Verbindungen

Mit der Gremlin-Konsole können Sie eine Verbindung zu einer Remote-Graph-Datenbank herstellen. Der folgende Abschnitt führt Sie durch die Installation und Konfiguration der Gremlin-Konsole zum Herstellen einer Remote-Verbindung mit einer Neptune-DB-Instance. Sie müssen diese Anweisungen für eine Amazon-EC2-Instance befolgen, die sich in derselben Virtual Private Cloud (VPC) wie Ihre Neptune-DB-Instance befindet.

Hilfe zum Herstellen einer Verbindung mit Neptune mit SSL/TLS (was erforderlich ist) finden Sie unter. SSL/TLS Konfiguration

> **Note**
>
> Wenn für Ihr Neptune-DB-Cluster die IAM-Authentifizierung aktiviert ist, folgen Sie für die Installation der Gremlin-Konsole den Anweisungen in Herstellen einer Verbindung zu Amazon Neptune Neptune-Datenbanken mithilfe der IAM-Authentifizierung mit der Gremlin-Konsole und nicht den hier bereitgestellten Anweisungen.

Installieren der Gremlin-Konsole und Verbinden mit Neptune

1. Die Binärdateien der Gremlin-Konsole erfordern Java 8 oder Java 11. Diese Anweisungen setzen die Verwendung von Java 11 voraus. Sie können Java 11 wie folgt auf Ihrer EC2-Instance installieren:

   - Wenn Sie Amazon Linux 2 (AL2) verwenden:

     ```
     sudo amazon-linux-extras install java-openjdk11
     ```

   - Wenn Sie Amazon Linux 2023 (AL2023) verwenden:

     ```
     sudo yum install java-11-amazon-corretto-devel
     ```

   - Verwenden Sie für andere Distributionen die jeweils zutreffende Anweisung:

```
sudo yum install java-11-openjdk-devel
```

oder:

```
sudo apt-get install openjdk-11-jdk
```

2. Geben Sie Folgendes ein, um Java 11 als Standardlaufzeit für Ihre EC2-Instance festzulegen.

```
sudo /usr/sbin/alternatives --config java
```

Geben Sie die Nummer für Java 11 ein, wenn Sie aufgefordert werden.

3. Laden Sie die entsprechende Version der Gremlin Console von der Apache-Website herunter. Sie können das überprüfenZugriff auf ein Neptune-Diagramm mit Gremlin, um festzustellen, welche Gremlin-Version Ihre Version von Neptune unterstützt. Wenn Sie beispielsweise Version 3.7.2 benötigen, können Sie die Gremlin-Konsole von der Apache Tinkerpop-Website wie folgt auf Ihre EC2-Instance herunterladen:

```
wget https://archive.apache.org/dist/tinkerpop/3.7.2/apache-tinkerpop-gremlin-
console-3.7.2-bin.zip
```

4. Entpacken Sie die ZIP-Datei der Gremlin-Konsole.

```
unzip apache-tinkerpop-gremlin-console-3.7.2-bin.zip
```

5. Ändern Sie die Verzeichnisse im extrahierten Verzeichnis.

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

6. Erstellen Sie im Unterverzeichnis `conf` des extrahierten Verzeichnisses eine Datei namens `neptune-remote.yaml` mit dem folgenden Text. *`your-neptune-endpoint`*Ersetzen Sie durch den Hostnamen oder die IP-Adresse Ihrer Neptune-DB-Instance. Die eckigen Klammern (`[ ]`) sind erforderlich.

> **Note**
>
> Informationen zum Ermitteln des Hostnamens Ihrer Neptune-DB-Instance finden Sie im Abschnitt Verbinden mit Amazo-Neptune-Endpunkten.

```
hosts: [your-neptune-endpoint]
port: 8182
connectionPool: { enableSsl: true }
serializer: { className:
 org.apache.tinkerpop.gremlin.util.ser.GraphBinaryMessageSerializerV1,
              config: { serializeResultToString: true }}
```

> **Note**
>
> Serializer wurden in Version 3.7.0 vom `gremlin-driver` Modul auf das neue `gremlin-util` Modul verschoben. Das Paket wurde von org.apache.tinkerpop.gremlin.driver.ser zu org.apache.tinkerpop.gremlin.util.ser geändert.

7. Navigieren Sie in einem Terminalfenster zu dem Verzeichnis der Gremlin-Konsole (`apache-tinkerpop-gremlin-console-3.7.2`), und geben Sie den folgenden Befehl ein, um die Gremlin-Konsole auszuführen.

   ```
   bin/gremlin.sh
   ```

   Die Ausgabe sollte folgendermaßen aussehen:

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

   Sie sehen nun die `gremlin`>-Eingabeaufforderung. Die nächsten Schritte geben Sie über diese Eingabeaufforderung ein.

8. Geben Sie an der `gremlin`>-Eingabeaufforderung Folgendes ein, um eine Verbindung zur Neptune-DB-Instance herzustellen.

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

9. Geben Sie an der `gremlin>`-Eingabeaufforderung Folgendes ein, um in den Remote-Modus zu wechseln. Dadurch werden alle Gremlin-Abfragen an die Remote-Verbindung gesendet.

```
:remote console
```

10. Geben Sie Folgendes ein, um eine Abfrage an den Gremlin-Graph zu senden.

```
g.V().limit(1)
```

11. Wenn Sie fertig sind, geben Sie den folgenden Befehl ein, um die Gremlin-Konsole zu beenden.

```
:exit
```

> **Note**
>
> Verwenden Sie ein Semikolon (`;`) oder ein Zeilenumbruchzeichen (\n), um die Anweisungen voneinander abzutrennen.
> Jede Traversierung, die der letzten Traversierung vorausgeht, muss zum Ausführen mit `next()` enden. Es werden nur die Daten der letzten Traversierung zurückgegeben.

Weitere Informationen zur Neptune-Implementierung von Gremlin finden Sie unter the section called “Einhaltung von Gremlin-Standards”.

Alternative Möglichkeit für die Verbindung mit der Gremlin-Konsole

Nachteile des normalen Ansatzes für die Verbindung

Die häufigste Art der Verbindung mit der Gremlin-Konsole wird oben beschrieben, bei der an der `gremlin>`-Eingabeaufforderung Befehle wie diese verwendet werden:

```
gremlin> :remote connect tinkerpop.server conf/(file name).yaml
gremlin> :remote console
```

Das funktioniert gut und ermöglicht Ihnen das Senden von Abfragen an Neptune. Dabei wird jedoch die Groovy-Skript-Engine nicht berücksichtigt, sodass Neptune alle Abfragen als reines Gremlin behandelt. Das bedeutet, dass die folgenden Abfrageformulare fehlschlagen:

```
gremlin> 1 + 1
```

```
gremlin> x = g.V().count()
```

Sie kommen bei einer Verbindung dieser Art der Verwendung einer Variablen am nächsten, wenn Sie die von der Konsole verwaltete Variable `result` verwenden und die Abfrage wie folgt mit `:>` senden:

```
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]
```

Eine andere Art der Verbindung

Sie können auch auf eine andere Art eine Verbindung mit der Gremlin-Konsole herstellen, die Sie vielleicht nützlicher finden:

```
gremlin> g = traversal().withRemote('conf/neptune.properties')
```

Hier hat `neptune.properties` das folgende Format:

```
gremlin.remote.remoteConnectionClass=org.apache.tinkerpop.gremlin.driver.remote.DriverRemoteCon
gremlin.remote.driver.clusterFile=conf/my-cluster.yaml
gremlin.remote.driver.sourceName=g
```

Die Datei `my-cluster.yaml` sollte wie folgt aussehen:

```
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**
>
> Serializer wurden in Version 3.7.0 vom Modul auf das neue Modul `gremlin-driver` verschoben. `gremlin-util` Das Paket wurde von org.apache.tinkerpop.gremlin.driver.ser zu org.apache.tinkerpop.gremlin.util.ser geändert.

Wenn Sie die Verbindung der Gremlin-Konsole so konfigurieren, können Sie die folgenden Arten von Abfragen erfolgreich ausführen:

```
gremlin> 1+1
==>2

gremlin> x=g.V().count().next()
==>4249

gremlin> println("The answer was ${x}")
The answer was 4249
```

Sie können die Anzeige des Ergebnisses wie folgt vermeiden:

```
gremlin> x=g.V().count().next();[]
gremlin> println(x)
4249
```

Alle gewöhnlichen Abfragemethoden (ohne terminalen Schritt) funktionieren weiter. Beispiel:

```
gremlin> g.V().count()
==>4249
```

Sie können sogar den Schritt `g.io().read()` verwenden, um eine Datei über diese Art von Verbindung zu laden.

IAM-Authentifizierung

Neptune unterstützt die IAM-Authentifizierung, um den Zugriff auf Ihren DB-Cluster zu kontrollieren. Wenn Sie die IAM-Authentifizierung aktiviert haben, müssen Sie die Signature Version 4-Signatur verwenden, um Ihre Anfragen zu authentifizieren. Ausführliche Anweisungen und Codebeispiele für das Herstellen einer Verbindung von der Gremlin-Konsole aus finden Sie unter. Herstellen einer

Verbindung zu Amazon Neptune Neptune-Datenbanken mithilfe der IAM-Authentifizierung mit der Gremlin-Konsole

Herstellen einer Verbindung mit einer Neptune-DB-Instance über den HTTPS-REST-Endpunkt

Amazon Neptune stellt einen HTTPS-Endpunkt für Gremlin-Abfragen bereit. Die REST-Schnittstelle ist mit jeder Gremlin-Version kompatibel, die Ihr DB-Cluster verwendet. (Auf der Seite Engine-Version der von Ihnen ausgeführten Engine-Version finden Sie Informationen zur unterstützten Gremlin-Version.)

> **Note**
>
> Wie in Verschlüsseln von Verbindungen zu Ihrer Amazon Neptune Neptune-Datenbank mit SSL/HTTPS beschrieben, erforrdert Neptune jetzt, dass Sie die Verbindung über HTTPS und nicht über HTTP herstellen. Darüber hinaus unterstützt HTTP/2 Neptune derzeit keine REST-API-Anfragen. Clients müssen dies verwenden, HTTP/1.1 wenn sie eine Verbindung zu Endpunkten herstellen.

Die folgenden Anweisungen führen Sie durch das Herstellen einer Verbindung zum Gremlin-Endpunkt mittels des `curl`-Befehls und HTTPS. Sie müssen diese Anweisungen für eine Amazon-EC2-Instance befolgen, die sich in derselben Virtual Private Cloud (VPC) wie Ihre Neptune-DB-Instance befindet.

Der HTTPS-Endpunkt für Gremlin-Abfragen an eine Neptune-DB-Instance ist `https://`*`your-neptune-endpoint`*`:`*`port`*`/gremlin`.

> **Note**
>
> Informationen zum Ermitteln des Hostnamens Ihrer Neptune-DB-Instance finden Sie in Verbinden mit Amazo-Neptune-Endpunkten.

Herstellen einer Verbindung mit Neptune über den HTTP-REST-Endpunkt

Die folgenden Beispiele zeigen, wie eine Gremlin-Abfrage an den REST-Endpunkt gesendet wird. Sie können das AWS SDK, das AWS CLI oder verwenden. curl

AWS CLI

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

Weitere Informationen finden Sie unter execute-gremlin-query in der Befehlsreferenz. 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))
```

AWS SDK-Beispiele in anderen Sprachen wie Java, .NET und mehr finden Sie unter. AWS 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**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

Weitere Informationen zur Verwendung awscurl mit der IAM-Authentifizierung finden Sie unter. Verwenden von `awscurl` mit temporären Anmeldeinformationen, um eine sichere Verbindung zu einem DB-Cluster mit aktivierter IAM-Authentifizierung herzustellen

curl

Im folgenden Beispiel wird curl zum Übermitteln einer Gremlin-Abfrage über HTTP POST verwendet. Die Abfrage wird im JSON-Format im Text des Posts als `gremlin`-Eigenschaft übermittelt.

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

Auch wenn zum Senden von Gremlin-Abfragen HTTP-POST-Anforderungen empfohlen werden, können Sie auch HTTP-GET-Anforderungen verwenden:

```
curl -G "https://your-neptune-endpoint:port?gremlin=g.V().count()"
```

In diesen Beispielen wird der erste Scheitelpunkt im Diagramm mithilfe der `g.V().limit(1)` Traversierung zurückgegeben. Um etwas anderes abzufragen, ersetzen Sie diese durch eine andere Gremlin-Traversierung.

Important

Der REST-Endpunkt gibt standardmäßig alle Ergebnisse in einem einzelnen JSON-Ergebnissatz zurück. Wenn dieser Ergebnissatz zu groß ist, kann eine `OutOfMemoryError`-Ausnahme für die Neptune-DB-Instance auftreten.
Sie können dies vermeiden, indem Sie die Aufteilung von Antworten aktivieren (Rückgabe der Ergebnisse in mehreren getrennten Antworten). Siehe Verwenden optionaler nachgestellter HTTP-Header zum Aktivieren mehrteiliger Gremlin-Antworten.

Note

Neptune unterstützt die Eigenschaft `bindings` nicht.

Verwenden optionaler nachgestellter HTTP-Header zum Aktivieren mehrteiliger Gremlin-Antworten

Standardmäßig wird die HTTP-Antwort auf Gremlin-Abfragen als einzelner JSON-Ergebnissatz zurückgegeben. Bei einem sehr großen Ergebnissatz kann dies zu einer `OutOfMemoryError`-Ausnahme für die DB-Instance führen.

Sie können jedoch die Aufteilung von Antworten aktivieren (die Rückgabe von Antworten in mehreren getrennten Teilen). Hierzu fügen Sie Ihrer Anforderung einen nachgestellten Transfer-Encoding (TE)-Header (`te: trailers`) hinzu. Weitere Informationen zu TE-Headern finden Sie auf der MDN-Seite zu TE-Anforderungs-Headern.

Bei Rückgabe einer Antwort in mehreren Teilen kann die Diagnose eines Problems schwierig sein, das nach dem Erhalt des ersten Teils auftritt, da der erste Teil mit dem HTTP-Statuscode `200` (OK) eingeht. Ein nachfolgender Fehler führt in der Regel zu einer Meldung mit einer ungültigen Antwort, an die Neptune eine Fehlermeldung anfügt.

Um die Erkennung und Diagnose dieser Art von Fehler zu vereinfachen, fügt Neptune in die nachgestellten Header der einzelnen Antwortteile zwei neue Header-Felder ein:

- `X-Neptune-Status` – enthält den Antwortcode gefolgt von einem Kurznamen. Im Erfolgsfall wäre der nachgestellte Header beispielsweise: `X-Neptune-Status: 200 OK`. Im Fehlerfall wäre der Antwortcode ein Neptune-Engine-Fehlercode wie `X-Neptune-Status: 500 TimeLimitExceededException`.
- `X-Neptune-Detail` – ist bei erfolgreichen Anforderungen leer. Im Fehlerfall ist die JSON-Fehlermeldung enthalten. Da in HTTP-Header-Werten nur ASCII-Zeichen zulässig sind, ist die JSON-Zeichenfolge URL-codiert.

> **Note**
>
> Neptune unterstützt zurzeit die `gzip`-Komprimierung aufgeteilter Antworten nicht. Wenn der Client gleichzeitig eine Aufteilung und eine Komprimierung anfordert, überspringt Neptune die Komprimierung.

Java-based Gremlin-Clients zur Verwendung mit Amazon Neptune

Sie können einen von zwei Java-based Open-Source-Gremlin-Clients mit Amazon Neptune verwenden: den Apache TinkerPop Java Gremlin-Client oder den Gremlin-Client für Amazon Neptune.

Apache Java Gremlin-Client TinkerPop

Der Apache TinkerPop Java Gremlin-Treiber ist der standardmäßige, offizielle Gremlin-Client, der mit jeder Graphdatenbank funktioniert. TinkerPop-enabled Verwenden Sie diesen Client, wenn Sie maximale Kompatibilität mit dem breiteren TinkerPop Entwicklungsbereich benötigen, wenn Sie mit mehreren Graphdatenbanksystemen arbeiten oder wenn Sie die für Neptune spezifischen erweiterten Clustermanagement- und Lastausgleichsfunktionen nicht benötigen. Dieser Client eignet sich auch für einfache Anwendungen, die eine Verbindung zu einer einzelnen Neptune-Instanz herstellen, oder wenn Sie den Lastenausgleich lieber auf der Infrastrukturebene als innerhalb des Clients durchführen möchten.

> **Important**
>
> Die Auswahl der richtigen Apache TinkerPop Gremlin-Treiberversion ist entscheidend für die Kompatibilität mit Ihrer Neptune-Engine-Version. Die Verwendung einer inkompatiblen Version kann zu Verbindungsfehlern oder unerwartetem Verhalten führen. Ausführliche Informationen zur Versionskompatibilität finden Sie unterZugriff auf ein Neptune-Diagramm mit Gremlin.

> **Note**
>
> Die Tabelle, die Ihnen hilft, die richtige TinkerPop Apache-Version für Neptune zu ermitteln, wurde verschoben. the section called “Gremlin” Diese Tabelle befand sich zuvor viele Jahre auf dieser Seite und ist jetzt als Referenz für alle unterstützten Programmiersprachen zentralisiert. TinkerPop

Gremlin-Java-Client für Amazon Neptune

Der Gremlin-Client für Amazon Neptune ist ein Java-based Open-Source-Gremlin-Client, der als Drop-In-Ersatz für den Standard-Java-Client fungiert. TinkerPop

Der Neptune-Gremlin-Client ist für Neptune-Cluster optimiert. Er ermöglicht die Verwaltung der Verteilung des Datenverkehrs auf mehrere Instances in einem Cluster und passt sich an Änderungen der Cluster-Topologie an, wenn Sie ein Replikat hinzufügen oder entfernen. Sie können den Client sogar so konfigurieren, dass er Anforderungen über eine Teilmenge von Instances im Cluster verteilt, basierend auf Rolle, Instance-Typ, Availability Zone (AZ) oder Tags, die den Instances zugeordnet sind.

Die neueste Version des Neptune-Gremlin-Java-Clients ist in Maven Central verfügbar.

Weitere Informationen zum Neptune-Gremlin-ava-Client finden Sie in diesem Blogbeitrag. Codebeispiele und Demos finden Sie im Projekt des Kunden. GitHub

Bei der Auswahl der Version des Neptune Gremlin-Clients müssen Sie die zugrunde liegende TinkerPop Version im Verhältnis zu Ihrer Neptune-Engine-Version berücksichtigen. Ermitteln Sie anhand der Kompatibilitätstabelle unter the section called “Gremlin” die richtige TinkerPop Version für Ihre Neptune-Engine und wählen Sie dann anhand der folgenden Tabelle die entsprechende Neptune Gremlin-Client-Version aus:

Kompatibilität der Neptune Gremlin-Clientversionen

Neptune Gremlin-Clientversion	TinkerPop Version
3.x	3.7.x (AWS SDK for Java 2. x/1.x)
2.1.x	3.7.x (AWS SDK for Java 1.x)
2.0.x	3.6.x
1.12	3.5.x

Herstellen einer Verbindung zu einer Neptune-DB-Instance über einen Java-Client

Der folgende Abschnitt führt Sie durch die Ausführung eines vollständigen Java-Beispiels, das eine Verbindung zu einer Neptune-DB-Instance herstellt und mithilfe des Apache Gremlin-Clients eine Gremlin-Traversal durchführt. TinkerPop

Sie müssen diese Anweisungen für eine Amazon-EC2-Instance befolgen, die sich in derselben Virtual Private Cloud (VPC) wie Ihre Neptune-DB-Instance befindet.

Herstellen einer Verbindung mit Neptune über Java

1. Installieren Sie Apache Maven auf Ihrer EC2-Instance. Wenn Sie Amazon Linux 2023 (bevorzugt) verwenden, verwenden Sie:

   ```
   sudo dnf update -y
   sudo dnf install maven -y
   ```

   Wenn Sie Amazon Linux 2 verwenden, laden Sie die neueste Binärdatei herunter von 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. Installieren Sie Java. Die Gremlin-Bibliotheken erfordern Java 8 oder 11. Sie können Java 11 wie folgt installieren:

   - Wenn Sie Amazon Linux 2 (AL2) verwenden:

     ```
     sudo amazon-linux-extras install java-openjdk11
     ```

   - Wenn Sie Amazon Linux 2023 (AL2023) verwenden:

     ```
     sudo yum install java-11-amazon-corretto-devel
     ```

   - Verwenden Sie für andere Distributionen die jeweils zutreffende Anweisung:

     ```
     sudo yum install java-11-openjdk-devel
     ```

     oder:

     ```
     sudo apt-get install openjdk-11-jdk
     ```

3. Legen Sie Java 11 als Standardlaufzeit für Ihre EC2-Instance fest: Geben Sie Folgendes ein, um Java 11 als Standardlaufzeit für Ihre EC2-Instance festzulegen:

```
sudo /usr/sbin/alternatives --config java
```

Geben Sie die Nummer für Java 11 ein, wenn Sie aufgefordert werden.

4. Erstellen Sie ein neues Verzeichnis mit dem Namen **`gremlinjava`**:

```
mkdir gremlinjava
cd gremlinjava
```

5. Erstellen Sie im Verzeichnis `gremlinjava` eine `pom.xml`-Datei und öffnen Sie diese in einem Text-Editor:

```
nano pom.xml
```

6. Kopieren Sie Folgendes in die `pom.xml`-Datei und speichern Sie sie:

```
<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**
>
> Wenn Sie ein vorhandenes Maven-Projekt ändern, ist die erforderliche Abhängigkeit im vorhergehenden Code hervorgehoben.

7. Erstellen Sie Unterverzeichnisse für den Beispielquellcode (`src/main/java/com/amazonaws/`), indem Sie Folgendes in die Befehlszeile eingeben:

```
mkdir -p src/main/java/com/amazonaws/
```

8. Erstellen Sie im Verzeichnis `src/main/java/com/amazonaws/` eine Datei namens `App.java` und öffnen Sie diese dann in einem Text-Editor.

```
nano src/main/java/com/amazonaws/App.java
```

9. Kopieren Sie Folgendes in die `App.java`-Datei. *`your-neptune-endpoint`*Ersetzen Sie durch die Adresse Ihrer Neptune-DB-Instance. Sie dürfen das Präfix `https://` nicht in die Methode `addContactPoint` einfügen.

> **Note**
>
> Informationen zum Ermitteln des Hostnamens Ihrer Neptune-DB-Instance finden Sie in Verbinden mit Amazo-Neptune-Endpunkten.

```
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();
  }
}
```

Hilfe zum Herstellen einer Verbindung mit Neptune mit SSL/TLS (was erforderlich ist) finden Sie unter. SSL/TLS Konfiguration

10. Nutzen Sie folgenden Maven-Befehl, um das Beispiel zu kompilieren und auszuführen:

```
mvn compile exec:exec
```

Das vorherige Beispiel gibt über die `g.V().limit(3).elementMap()`-Traversierung eine Übersicht der Schlüssel und Werte der einzelnen Eigenschaften für die ersten beiden Knoten im Diagramm zurück. Um etwas anderes abzufragen, ersetzen Sie diese durch eine andere Gremlin-Traversierung mit einer der entsprechenden Ending-Methoden.

> **Note**
>
> Der letzte Teil der Gremlin-Abfrage, `.toList()`, ist für die Übermittlung der Traversierung zur Auswertung an den Server erforderlich. Wenn Sie diese oder eine gleichwertige Methode nicht einschließen, wird die Abfrage nicht an die Neptune-DB-Instance übermittelt.
> Sie müssen auch eine geeignete Endung anfügen, wenn Sie einen Eckpunkt oder eine Kante hinzufügen, wie beim Schritt `addV( )`.

Im Folgenden finden Sie Beispiele für Methoden, die die Abfrage an die Neptune-DB-Instance senden:

- `toList()`
- `toSet()`
- `next()`
- `iterate()`

Verwenden Sie diese Option, `iterate()` wenn Sie die Ergebnisse Ihrer Abfragen (z. B. Mutationen) nicht benötigen, da dies Serialisierungskosten spart.

Diese Terminalschritte verhalten sich unterschiedlich, je nachdem, ob Sie Abfragen als Zeichenketten (Skriptmodus) oder als Bytecode-Traversal-Objekte (Bytecode-Modus) einreichen. Die Unterschiede sind besonders wichtig für die Transaktionsverarbeitung. Details hierzu finden Sie unter Testen Sie den Gremlin-Code in dem Kontext, in dem Sie ihn einsetzen werden.

SSL/TLS Konfiguration für den Gremlin Java-Client

Neptune SSL/TLS muss standardmäßig aktiviert sein. Wenn der Java-Treiber mit `enableSsl(true)` konfiguriert ist, kann er in der Regel eine Verbindung mit Neptune herstellen, ohne einen `trustStore()` oder `keyStore()` mit der lokalen Kopie eines Zertifikats einrichten zu müssen.

Wenn die Instance, mit der Sie eine Verbindung herstellen, keine Internetverbindung hat, um ein öffentliches Zertifikat zu verifizieren, oder wenn das verwendete Zertifikat nicht öffentlich ist, können Sie zur Konfiguration einer lokalen Zertifikatkopie die folgenden Schritte ausführen:

Einrichtung einer lokalen Zertifikatkopie zur Aktivierung SSL/TLS

1. Laden Sie keytool von Oracle herunter und installieren Sie es. Dies vereinfacht die Einrichtung des lokalen Schlüsselspeichers erheblich.
2. Laden Sie das `SFSRootCAG2.pem`-CA-Zertifikat herunter (das Gremlin-Java-SDK erfordert ein Zertifikat, um das Remote-Zertifikat zu verifizieren):

   ```
   wget https://www.amazontrust.com/repository/SFSRootCAG2.pem
   ```

3. Erstellen Sie einen Schlüsselspeicher im JKS- oder PKCS12-Format. In diesem Beispiel wird JKS verwendet. Beantworten Sie die Fragen an der Eingabeaufforderung. Das hier erstellte Passwort wird später benötigt:

   ```
   keytool -genkey -alias (host name) -keyalg RSA -keystore server.jks
   ```

4. Importieren Sie die heruntergeladene Datei `SFSRootCAG2.pem` in den neu erstellten Schlüsselspeicher:

   ```
   keytool -import -keystore server.jks -file .pem
   ```

5. Konfigurieren Sie das `Cluster`-Objekt programmgesteuert:

   ```
   Cluster cluster = Cluster.build("(your neptune endpoint)")
                            .port(8182)
                            .enableSSL(true)
                            .keyStore('server.jks')
                            .keyStorePassword("(the password from step 2)")
                            .create();
   ```

   Sie können dies auch in einer Konfigurationsdatei ausführen, wenn Sie möchten, wie im Fall der Gremlin-Konsole:

   ```
   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 }}
   ```

IAM-Authentifizierung

Neptune unterstützt die IAM-Authentifizierung, um den Zugriff auf Ihren DB-Cluster zu kontrollieren. Wenn Sie die IAM-Authentifizierung aktiviert haben, müssen Sie die Signature Version 4-Signatur verwenden, um Ihre Anfragen zu authentifizieren. Ausführliche Anweisungen und Codebeispiele für die Verbindung von einem Java-Client aus finden Sie unter. Herstellen einer Verbindung zu Amazon Neptune Neptune-Datenbanken mithilfe von IAM mit Gremlin Java

Java-Beispiel für die Herstellung einer Verbindung mit einer Neptune-DB-Instance mit Wiederverbindungslogik

Das folgende Java-Beispiel zeigt, wie Sie eine Verbindung mit Wiederverbindungslogik zum Gremlin-Client herstellen, um die Verbindung nach einer unerwarteten Unterbrechung wiederherzustellen.

Ausführliche Anleitungen zur Entwicklung einer praktischen Wiederholungsstrategie, einschließlich exponentiellem Backoff mit Jitter, finden Sie unter. the section called “Ausnahmen und Wiederholungen”

Es gelten die folgenden Abhängigkeiten:

```
<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>
```

Hier finden Sie den Beispielcode:

> **Important**
>
> Der `CallExecutor` von Retry4j ist möglicherweise nicht threadsicher. Erwägen Sie, dass jeder Thread seine eigene `CallExecutor` Instanz oder eine andere Wiederholungsbibliothek verwendet.

> **Note**
>
> Das folgende Beispiel wurde aktualisiert und beinhaltet nun die Verwendung von requestInterceptor (). Dies wurde in 3.6.6 hinzugefügt. TinkerPop Vor TinkerPop Version 3.6.6 verwendete das Codebeispiel HandshakeInterceptor (), das in dieser Version veraltet war.

```
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;
  };
}
```

Herstellen einer Verbindung mit einer Neptune-DB-Instance über Python

> **Important**
>
> Die Auswahl der richtigen Apache TinkerPop Gremlin-Treiberversion ist entscheidend für die Kompatibilität mit Ihrer Neptune-Engine-Version. Die Verwendung einer inkompatiblen Version kann zu Verbindungsfehlern oder unerwartetem Verhalten führen. Ausführliche Informationen zur Versionskompatibilität finden Sie unterZugriff auf ein Neptune-Diagramm mit Gremlin.

Der folgende Abschnitt führt Sie durch die Ausführung eines Python-Beispiels mit Herstellung einer Verbindung zu einer Amazon-Neptune-DB-Instance und Ausführung einer Gremlin-Traversierung.

Sie müssen diese Anweisungen für eine Amazon-EC2-Instance befolgen, die sich in derselben Virtual Private Cloud (VPC) wie Ihre Neptune-DB-Instance befindet.

Bevor Sie beginnen, führen Sie die folgenden Schritte aus:

- Laden Sie Python 3.6 oder höher von der Python.org Website herunter und installieren Sie es.
- Überprüfen Sie, ob Sie pip installiert haben. Haben Sie pip nicht installiert oder sind sich nicht sicher, lesen Sie Do I need to install pip? in der pip-Dokumenation.
- Wenn Ihre Python-Installation nicht bereits darüber verfügt, ist der Download von `futures` wie folgt möglich: `pip install futures`

Herstellen einer Verbindung mit Neptune über Python

1. Geben Sie Folgendes ein, um das `gremlinpython`-Paket zu installieren:

   ```
   pip install --user gremlinpython
   ```

2. Erstellen Sie eine Datei namens `gremlinexample.py` und öffnen Sie diese dann in einem Text-Editor.

3. Kopieren Sie Folgendes in die `gremlinexample.py`-Datei. *`your-neptune-endpoint`*Ersetzen Sie durch die Adresse Ihres Neptune-DB-Clusters und *`your-neptune-port`* durch den Port Ihres Neptune-DB-Clusters (Standard:8182).

   Informationen zum Ermitteln der Adresse Ihrer Neptune-DB-Instance finden Sie im Abschnitt Verbinden mit Amazo-Neptune-Endpunkten.

   Das folgende Beispiel zeigt, wie man sich mit Gremlin Python verbindet.

```
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. Geben Sie den folgenden Befehl ein, um das Beispiel auszuführen:

```
python gremlinexample.py
```

   Die Gremlin-Abfrage am Ende dieses Beispiel gibt die Knoten (`g.V().limit(2)`) in einer Liste zurück. Diese Liste wird dann mit der Python-Standardfunktion `print` gedruckt.

> **Note**
>
> Der letzte Teil der Gremlin-Abfrage, `toList()`, ist für die Übermittlung der Traversierung zur Auswertung an den Server erforderlich. Wenn Sie diese oder eine

gleichwertige Methode nicht einschließen, wird die Abfrage nicht an die Neptune-DB-Instance übermittelt.

Im Folgenden finden Sie Beispiele für Methoden, die die Abfrage an die Neptune-DB-Instance senden:

- `toList()`
- `toSet()`
- `next()`
- `iterate()`

Diese Terminalschritte verhalten sich im Skriptmodus und im Bytecode-Modus unterschiedlich. Eine kanonische Liste der Terminalschritte und Einzelheiten darüber, wie sie sich auf Transaktionen auswirken, finden Sie unter. Testen Sie den Gremlin-Code in dem Kontext, in dem Sie ihn einsetzen werden

Verwenden Sie diese Option, `iterate()` wenn Sie die Ergebnisse Ihrer Abfragen (z. B. Mutationen) nicht benötigen, da dadurch Serialisierungskosten eingespart werden.

Das vorherige Beispiel gibt die ersten beiden Knoten im Diagramm über die `g.V().limit(2).toList()`-Traversierung zurück. Um etwas anderes abzufragen, ersetzen Sie diese durch eine andere Gremlin-Traversierung mit einer der entsprechenden Ending-Methoden.

IAM-Authentifizierung

Neptune unterstützt die IAM-Authentifizierung, um den Zugriff auf Ihren DB-Cluster zu kontrollieren. Wenn Sie die IAM-Authentifizierung aktiviert haben, müssen Sie die Signature Version 4-Signatur verwenden, um Ihre Anfragen zu authentifizieren. Ausführliche Anweisungen und Codebeispiele für die Verbindung von einem Python-Client aus finden Sie unterHerstellen einer Verbindung zu Amazon Neptune Neptune-Datenbanken mithilfe der IAM-Authentifizierung mit Gremlin Python.

Herstellen einer Verbindung mit einer Neptune-DB-Instance über .NET

> **Important**
>
> Die Auswahl der richtigen Apache TinkerPop Gremlin-Treiberversion ist entscheidend für die Kompatibilität mit Ihrer Neptune-Engine-Version. Die Verwendung einer inkompatiblen Version kann zu Verbindungsfehlern oder unerwartetem Verhalten führen. Ausführliche Informationen zur Versionskompatibilität finden Sie unterZugriff auf ein Neptune-Diagramm mit Gremlin.

Der folgende Abschnitt enthält ein in C# geschriebenes Codebeispiel, das eine Verbindung mit einer Neptune-DB-Instance herstellt und eine Gremlin-Traversierung ausführt.

Die Verbindungen mit Amazon Neptune müssen über eine Amazon-EC2-Instance erfolgen, die sich in derselben Virtual Private Cloud (VPC) wie Ihre Neptune-DB-Instance befindet. Dieser Beispielcode wurde auf einer Amazon-EC2-Instance getestet, auf der Ubuntu ausgeführt wird.

Bevor Sie beginnen, führen Sie die folgenden Schritte aus:

- Installieren Sie .NET auf der Amazon-EC2-Instance. Informationen zum Installieren von .NET auf verschiedenen Betriebssystemen einschließlich Windows, Linux und macOS finden Sie unter Get Started with .NET.
- Führen Sie Gremlin.NET die Installation aus, indem Sie Ihr Paket ausführen`dotnet add package gremlin.net`. Weitere Informationen finden Sie Gremlin.NETin der TinkerPop Dokumentation.

Um eine Verbindung zu Neptune herzustellen mit Gremlin.NET

1. Erstellen Sie ein neues .NET-Projekt.

   ```
   dotnet new console -o gremlinExample
   ```

2. Ändern Sie die Verzeichnisse in das neue Projektverzeichnis.

   ```
   cd gremlinExample
   ```

3. Kopieren Sie Folgendes in die `Program.cs`-Datei. *`your-neptune-endpoint`* Ersetzen Sie durch die Adresse Ihrer Neptune-DB-Instance.

   Informationen zum Ermitteln der Adresse Ihrer Neptune-DB-Instance finden Sie im Abschnitt Verbinden mit Amazo-Neptune-Endpunkten.

```
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. Geben Sie den folgenden Befehl ein, um das Beispiel auszuführen:

```
dotnet run
```

Die Gremlin-Abfrage am Ende dieses Beispiels gibt die Anzahl eines einzelnen Knotens zu Testzwecken zurück. Sie wird dann in der Konsole ausgegeben.

> **Note**
>
> Der letzte Teil der Gremlin-Abfrage, `Next()`, ist für die Übermittlung der Traversierung zur Auswertung an den Server erforderlich. Wenn Sie diese oder eine gleichwertige Methode nicht einschließen, wird die Abfrage nicht an die Neptune-DB-Instance übermittelt.

Im Folgenden finden Sie Beispiele für Methoden, die die Abfrage an die Neptune-DB-Instance senden:

- `ToList()`
- `ToSet()`
- `Next()`
- `Iterate()`

Diese Terminalschritte verhalten sich im Skriptmodus und im Bytecode-Modus unterschiedlich. Eine kanonische Liste der Terminalschritte und Einzelheiten darüber, wie sie sich auf Transaktionen auswirken, finden Sie unter. Testen Sie den Gremlin-Code in dem Kontext, in dem Sie ihn einsetzen werden

Verwenden Sie diese Option, `Iterate()` wenn Sie die Ergebnisse Ihrer Abfragen (z. B. Mutationen) nicht benötigen, da dadurch Serialisierungskosten eingespart werden.

Das vorherige Beispiel gibt eine Liste über die Traversierung `g.V().Limit(3).ToList()` zurück. Um etwas anderes abzufragen, ersetzen Sie diese durch eine andere Gremlin-Traversierung mit einer der entsprechenden Ending-Methoden.

IAM-Authentifizierung

Neptune unterstützt die IAM-Authentifizierung, um den Zugriff auf Ihren DB-Cluster zu kontrollieren. Wenn Sie die IAM-Authentifizierung aktiviert haben, müssen Sie die Signature Version 4-Signatur verwenden, um Ihre Anfragen zu authentifizieren. Ausführliche Anweisungen und Codebeispiele für die Verbindung von einem .NET-Client aus finden Sie unter. Herstellen einer Verbindung zu Amazon Neptune Neptune-Datenbanken mithilfe der IAM-Authentifizierung mit Gremlin .NET

Wird verwendet Node.js , um eine Verbindung zu einer Neptune-DB-Instance herzustellen

> **Important**
>
> Die Auswahl der richtigen Apache TinkerPop Gremlin-Treiberversion ist entscheidend für die Kompatibilität mit Ihrer Neptune-Engine-Version. Die Verwendung einer inkompatiblen Version kann zu Verbindungsfehlern oder unerwartetem Verhalten führen. Ausführliche Informationen zur Versionskompatibilität finden Sie unterZugriff auf ein Neptune-Diagramm mit Gremlin.

Der folgende Abschnitt führt Sie durch die Ausführung eines Node.js Beispiels, das eine Verbindung zu einer Amazon Neptune Neptune-DB-Instance herstellt und einen Gremlin-Traversal durchführt.

Sie müssen diese Anweisungen für eine Amazon-EC2-Instance befolgen, die sich in derselben Virtual Private Cloud (VPC) wie Ihre Neptune-DB-Instance befindet.

Bevor Sie beginnen, führen Sie die folgenden Schritte aus:

- Stellen Sie sicher, dass Node.js Version 8.11 oder höher installiert ist. Ist dies nicht der Fall, laden Sie es Node.js von der Nodejs.org Website herunter und installieren Sie es.

Um eine Verbindung zu Neptune herzustellen mit Node.js

1. Geben Sie Folgendes ein, um das `gremlin-javascript`-Paket zu installieren:

   ```
   npm install gremlin
   ```

2. Erstellen Sie eine Datei namens `gremlinexample.js` und öffnen Sie diese dann in einem Text-Editor.

3. Kopieren Sie Folgendes in die `gremlinexample.js`-Datei. *`your-neptune-endpoint`*Ersetzen Sie durch die Adresse Ihrer Neptune-DB-Instance.

   Informationen zum Ermitteln der Adresse Ihrer Neptune-DB-Instance finden Sie im Abschnitt Verbinden mit Amazo-Neptune-Endpunkten.

   ```
   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. Geben Sie den folgenden Befehl ein, um das Beispiel auszuführen:

   ```
   node gremlinexample.js
   ```

Das vorherige Beispiel gibt über die `g.V().limit(1).count().next()`-Traversierung die Anzahl eines einzelnen Vertex im Diagramm zurück. Um etwas anderes abzufragen, ersetzen Sie diese durch eine andere Gremlin-Traversierung mit einer der entsprechenden Ending-Methoden.

> **Note**
>
> Der letzte Teil der Gremlin-Abfrage, `next()`, ist für die Übermittlung der Traversierung zur Auswertung an den Server erforderlich. Wenn Sie diese oder eine gleichwertige Methode nicht einschließen, wird die Abfrage nicht an die Neptune-DB-Instance übermittelt.

Im Folgenden finden Sie Beispiele für Methoden, die die Abfrage an die Neptune-DB-Instance senden:

- `toList()`
- `toSet()`
- `next()`
- `iterate()`

Diese Terminalschritte verhalten sich im Skriptmodus und im Bytecode-Modus unterschiedlich. Eine kanonische Liste der Terminalschritte und Einzelheiten darüber, wie sie sich auf Transaktionen auswirken, finden Sie unter. Testen Sie den Gremlin-Code in dem Kontext, in dem Sie ihn einsetzen werden

Verwenden Sie diese Option, `iterate()` wenn Sie die Ergebnisse Ihrer Abfragen (z. B. Mutationen) nicht benötigen, da dadurch Serialisierungskosten eingespart werden.

> **Important**
>
> Dies ist ein eigenständiges Node.js Beispiel. Wenn Sie planen, Code wie diesen in einer AWS Lambda Funktion auszuführen, finden Sie weitere Informationen Beispiele für Lambda-Funktionen zur JavaScript effizienten Verwendung in einer Neptune-Lambda-Funktion.

IAM-Authentifizierung

Neptune unterstützt die IAM-Authentifizierung, um den Zugriff auf Ihren DB-Cluster zu kontrollieren. Wenn Sie die IAM-Authentifizierung aktiviert haben, müssen Sie die Signature Version 4-Signatur verwenden, um Ihre Anfragen zu authentifizieren. Ausführliche Anweisungen und Codebeispiele für das Herstellen einer Verbindung von einem JavaScript Client aus finden Sie unter. Herstellen einer Verbindung zu Amazon Neptune Neptune-Datenbanken mithilfe der IAM-Authentifizierung mit Gremlin JavaScript

Herstellen einer Verbindung mit einer Neptune-DB-Instance über Go

> **Important**
>
> Die Auswahl der richtigen Apache TinkerPop Gremlin-Treiberversion ist entscheidend für die Kompatibilität mit Ihrer Neptune-Engine-Version. Die Verwendung einer inkompatiblen

Version kann zu Verbindungsfehlern oder unerwartetem Verhalten führen. Ausführliche Informationen zur Versionskompatibilität finden Sie unterZugriff auf ein Neptune-Diagramm mit Gremlin.

> **Note**
>
> Die Gremlingo 3.5.x-Versionen sind abwärtskompatibel mit den TinkerPop 3.4.x-Versionen, solange Sie in den von Ihnen geschriebenen Gremlin-Abfragen nur 3.4.x-Funktionen verwenden.

Der folgende Abschnitt führt Sie durch die Ausführung eines Go-Beispiels mit Herstellung einer Verbindung zu einer Amazon-Neptune-DB-Instance und Ausführung einer Gremlin-Traversierung.

Sie müssen diese Anweisungen für eine Amazon-EC2-Instance befolgen, die sich in derselben Virtual Private Cloud (VPC) wie Ihre Neptune-DB-Instance befindet.

Bevor Sie beginnen, führen Sie die folgenden Schritte aus:

- Laden Sie Go 1.17 oder höher von der Website go.dev herunter und installieren Sie diese Version.

Herstellen einer Verbindung mit Neptune über Go

1. Initalisieren Sie ausgehend von einem leeren Verzeichnis ein neues Go-Modul:

   ```
   go mod init example.com/gremlinExample
   ```

2. Fügen Sie gremlin-go als Abhängigkeit Ihres neuen Moduls hinzu:

   ```
   go get github.com/apache/tinkerpop/gremlin-go/v3/driver
   ```

3. Erstellen Sie eine Datei namens `gremlinExample.go` und öffnen Sie diese dann in einem Text-Editor.
4. Kopieren Sie Folgendes in die Datei `gremlinExample.go` und ersetzen Sie *`(your neptune endpoint)`* durch die Adresse Ihrer Neptune-DB-Instance:

   ```
   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

Das Neptune-TLS-Zertifikatformat wird zurzeit für Go 1.18+ mit macOS nicht unterstützt und kann beim Versuch, eine Verbindung herzustellen, zu einem 509-Fehler führen. Für lokale Tests kann dies übersprungen werden, indem "" zu den Importen hinzugefügt und die Einstellungen wie folgt geändert werden: crypto/tls `DriverRemoteConnection`

```
// 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. Geben Sie den folgenden Befehl ein, um das Beispiel auszuführen:

```
go run gremlinExample.go
```

Die Gremlin-Abfrage am Ende dieses Beispiels gibt die Eckpunkte (`g.V().Limit(2)`) in einem Slice zurück. Dieser Slice wird dann iteriert und mit der Standardfunktion `fmt.Println` gedruckt.

> **Note**
>
> Der letzte Teil der Gremlin-Abfrage, `ToList()`, ist für die Übermittlung der Traversierung zur Auswertung an den Server erforderlich. Wenn Sie diese oder eine gleichwertige Methode nicht einschließen, wird die Abfrage nicht an die Neptune-DB-Instance übermittelt.

Im Folgenden finden Sie Beispiele für Methoden, die die Abfrage an die Neptune-DB-Instance senden:

- `ToList()`
- `ToSet()`
- `Next()`
- `Iterate()`

Diese Terminalschritte verhalten sich im Skriptmodus und im Bytecode-Modus unterschiedlich. Eine kanonische Liste der Terminalschritte und Einzelheiten darüber, wie sie sich auf Transaktionen auswirken, finden Sie unter. Testen Sie den Gremlin-Code in dem Kontext, in dem Sie ihn einsetzen werden

Verwenden Sie diese Option, `Iterate()` wenn Sie die Ergebnisse Ihrer Abfragen (z. B. Mutationen) nicht benötigen, da dadurch Serialisierungskosten eingespart werden.

Das vorherige Beispiel gibt die ersten beiden Knoten im Diagramm über die `g.V().Limit(2).ToList()`-Traversierung zurück. Um etwas anderes abzufragen, ersetzen Sie diese durch eine andere Gremlin-Traversierung mit einer der entsprechenden Ending-Methoden.

IAM-Authentifizierung

Neptune unterstützt die IAM-Authentifizierung, um den Zugriff auf Ihren DB-Cluster zu kontrollieren. Wenn Sie die IAM-Authentifizierung aktiviert haben, müssen Sie die Signature Version 4-Signatur verwenden, um Ihre Anfragen zu authentifizieren. Ausführliche Anweisungen und Codebeispiele für die Verbindung von einem Go-Client aus finden Sie unter. Herstellen einer Verbindung zu Amazon Neptune Neptune-Datenbanken mithilfe der IAM-Authentifizierung mit Gremlin Go

Verwendung der AWS SDK zum Ausführen von Gremlin-Abfragen

Mit dem AWS SDK können Sie Gremlin-Abfragen für Ihren Neptune-Graphen mit einer Programmiersprache Ihrer Wahl ausführen. Das Neptune Data API SDK (Dienstname`neptunedata`) bietet die ExecuteGremlinQueryAktion zum Senden von Gremlin-Abfragen.

Sie müssen diese Beispiele von einer Amazon EC2 EC2-Instance in derselben Virtual Private Cloud (VPC) wie Ihr Neptune-DB-Cluster oder von einem Standort aus ausführen, der über eine Netzwerkverbindung zu Ihrem Cluster-Endpunkt verfügt.

Direkte Links zur API-Referenzdokumentation für den `neptunedata` Service in jeder SDK-Sprache finden Sie unten:

Programmiersprache	Neptunedata API-Referenz
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

Programmiersprache	Neptunedata API-Referenz
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
Ruby	https://docs.aws.amazon.com/sdk-for-ruby/v3/api/Aws/Neptunedata.html
Rust	https://crates.io/crates/aws-sdk-neptunedata
CLI	https://docs.aws.amazon.com/cli/latest/reference/neptunedata/

Gremlin AWS SDK-Beispiele

Die folgenden Beispiele zeigen, wie Sie einen `neptunedata` Client einrichten, eine Gremlin-Abfrage ausführen und die Ergebnisse drucken. Ersetzen Sie *`YOUR_NEPTUNE_HOST`* und *`YOUR_NEPTUNE_PORT`* durch den Endpunkt und Port Ihres Neptune-DB-Clusters.

⚠ Client-side Timeout und Konfiguration erneut versuchen

Das SDK-Client-Timeout steuert, wie lange der Client auf eine Antwort wartet. Es steuert nicht, wie lange die Abfrage auf dem Server ausgeführt wird. Wenn der Client das Timeout überschreitet, bevor der Server fertig ist, wird die Abfrage möglicherweise weiterhin auf Neptune ausgeführt, während der Client keine Möglichkeit hat, die Ergebnisse abzurufen.

Wir empfehlen, das clientseitige Lese-Timeout auf `0` (kein Timeout) oder auf einen Wert zu setzen, der mindestens ein paar Sekunden länger ist als die serverseitige neptune_query_timeout-Einstellung auf Ihrem Neptune-DB-Cluster. Auf diese Weise kann Neptune kontrollieren, wann das Timeout bei Abfragen abläuft.
Wir empfehlen außerdem, die maximale Anzahl an Wiederholungsversuchen auf 1 (keine Wiederholungsversuche) festzulegen. Wenn das SDK eine Abfrage wiederholt, die noch auf dem Server ausgeführt wird, kann dies zu doppelten Vorgängen führen. Dies ist besonders wichtig bei Mutationsabfragen, bei denen ein erneuter Versuch zu unbeabsichtigten doppelten Schreibvorgängen führen kann.

Python

1. Folgen Sie den Installationsanweisungen, um Boto3 zu installieren.
2. Erstellen Sie eine Datei mit dem Namen `gremlinExample.py` und fügen Sie den folgenden Code ein:

```
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. Führen Sie das Beispiel aus: `python gremlinExample.py`

Java

1. Folgen Sie den Installationsanweisungen, um das AWS SDK for Java einzurichten.
2. Verwenden Sie den folgenden Code, um eine Gremlin-Abfrage einzurichten`NeptunedataClient`, eine Gremlin-Abfrage auszuführen und das Ergebnis auszudrucken:

```
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. Folgen Sie den Installationsanweisungen, um das AWS SDK für einzurichten. JavaScript Installieren Sie das Neptunedata-Client-Paket:. `npm install @aws-sdk/client-neptunedata`
2. Erstellen Sie eine Datei mit dem Namen `gremlinExample.js` und fügen Sie den folgenden Code ein:

```
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. Führen Sie das Beispiel aus: `node gremlinExample.js`

Gremlin-Abfragehinweise

Sie können Abfragehinweise verwenden, um Optimierungs- und Evaluierungsstrategien für eine bestimmte Gremlin-Abfrage in Amazon Neptune anzugeben.

Die meisten Abfragehinweise werden angegeben, indem der Abfrage ein `withSideEffect` Schritt mit der folgenden Syntax hinzugefügt wird.

```
g.withSideEffect(hint, value)
```

- hint – Identifiziert die Art des anzuwendenden Hinweises.
- value – Legt das Verhalten des jeweiligen Systemaspekts fest.

Das folgende Beispiel illustriert das Einfügen eines `repeatMode`-Hinweises in eine Gremlin-Traversierung.

> **Note**
>
> Abfragehinweise, die ich verwende, `withSideEffect` haben immer das Präfix. `Neptune#`

```
g.withSideEffect('Neptune#repeatMode',
 'DFS').V("3").repeat(out()).times(10).limit(1).path()
```

Die vorherige Abfrage weist die Neptune-Engine an, das Diagramm nach Depth First (`DFS`) und nicht nach der Neptune-Standardeinstellung Breadth First (`BFS`) zu traversieren.

> **Important**
>
> Einige Abfragehinweise verwenden den `with` Schritt anstelle von. `withSideEffect` Die beiden sind nicht austauschbar. Jeder Abschnitt mit Abfragehinweisen auf dieser Seite zeigt die richtige zu verwendende Syntax.

In den folgenden Abschnitten finden Sie weitere Informationen zu den verfügbaren Abfragehinweisen und ihre Nutzung.

Themen

- Gremlin-Abfragehinweis repeatMode
- Gremlin-Abfragehinweis noReordering
- Gremlin-Abfragehinweis typePromotion

- Gremlin-Abfragehinweis useDFE
- Hinweis zur Gremlin-Evaluierungsabfrage TimeoutBehavior
- Gremlin-Abfragehinweise für die Verwendung des Ergebniscaches

Gremlin-Abfragehinweis repeatMode

Der Neptune-Abfragehinweis `repeatMode` gibt an, wie die Neptune-Engine den Schritt `repeat()` in einer Gremlin-Traversierung evaluiert: Breadth First, Depth First oder Chunked Depth.

Der Evaluierungsmodus des `repeat()`-Schritts ist wichtig, wenn er verwendet wird, um einen Pfad zu finden oder ihm zu folgen, anstatt einfach einen Schritt für eine bestimmte Zahl von Malen zu wiederholen.

Syntax

Der `repeatMode`-Abfragehinweis wird durch Hinzufügen eines `withSideEffect`-Schritts zu der Abfrage angegeben.

```
g.withSideEffect('Neptune#repeatMode', 'mode').gremlin-traversal
```

> **Note**
>
> Alle Gremlin-Abfragehinweis-Nebeneffekte haben das Präfix `Neptune#`.

Verfügbare Modi

- `BFS`

  Breadth-First Suchen

  Standard-Ausführungsmodus für den `repeat()`-Schritt. Dies erfasst alle verwandten Knoten auf derselben Ebene, bevor der Pfad tiefer weiterverfolgt wird.

  Diese Version ist speicherintensiv, und die Grenzen können sehr groß werden. Es besteht ein höheres Risiko, dass der Arbeitsspeicher für die Abfrage nicht ausreicht und sie von der Neptune-Engine abgebrochen wird. Dies entspricht weitestgehend anderen Gremlin-Implementierungen.
- `DFS`

Depth-First Suchen

Folgt jedem Pfad bis zur maximalen Tiefe, bevor zur nächsten Lösung weitergegangen wird.

Dies verbraucht weniger Speicher. Es kann eine bessere Leistung in Situationen wie beim Suchen nach einem einzelnen Pfad ab dem Startpunkt aus mehreren Hops bieten.

- `CHUNKED_DFS`

  Gebündelte Suche Depth-First

  Ein hybrides Verfahren, bei dem die Graph-Tiefe zuerst in Gruppen von 1 000 Knoten (und nicht nach einzelnen Knoten) (`DFS`) oder alle Knoten (`BFS`) durchsucht werden.

  Die Neptune-Engine erfasst bis zu 1 000 Knoten auf jeder Ebene, bevor der Pfad tiefer weiterverfolgt wird.

  Diese Vorgehensweise sorgt für einen besseren Ausgleich zwischen Geschwindigkeit und Speichernutzung.

  Dies ist auch nützlich, wenn Sie `BFS` verwenden möchten, die Abfrage verwendet jedoch zu viel Speicherplatz.

Beispiel

Der folgende Abschnitt beschreibt die Auswirkung des Repeat-Modus auf eine Gremlin-Traversierung.

In Neptune wird für den `repeat()`-Schritt standardmäßig eine Breadth-First-Ausführungsstrategie (`BFS`) für alle Traversierungen ausgeführt.

In den meisten Fällen verwendet die TinkerGraph Implementierung dieselbe Ausführungsstrategie, in einigen Fällen ändert sie jedoch die Ausführung einer Durchquerung.

Die TinkerGraph Implementierung ändert beispielsweise die folgende Abfrage.

```
g.V("3").repeat(out()).times(10).limit(1).path()
```

Der `repeat()`-Schritt in dieser Traversierung wird zur folgenden Traversierung „ausgerollt“, was zu einer Depth First-Strategie (`DFS`) führt.

```
g.V(<id>).out().out().out().out().out().out().out().out().out().out().limit(1).path()
```

> **Important**
>
> Die Neptune-Abfrage-Engine führt dies nicht automatisch aus.

Breadth-first (BFS) ist die Standardausführungsstrategie und ähnelt TinkerGraph in den meisten Fällen. Es gibt jedoch einige Fälle, in denen Depth-First (DFS)-Strategien vorzuziehen sind.

BFS (Standard)

Breadth-first (BFS) ist die Standardausführungsstrategie für den `repeat()` Operator.

```
g.V("3").repeat(out()).times(10).limit(1).path()
```

Die Neptune-Engine durchsucht die ersten Neun-Hop-Grenzen vollständig, bevor nach einer Lösung ab dem zehnten Hop gesucht wird. Dies ist in vielen Fällen effektiv, etwa bei Abfragen nach dem kürzesten Pfad.

Im vorangehenden Beispiel wäre die Traversierung mittels des Depth-First-Modus (DFS) für den Operator `repeat()` jedoch sehr viel schneller.

DFS

Die folgende Abfrage verwendet den Depth-First-`repeat()`-Modus für den DFS-Operator.

```
g.withSideEffect("Neptune#repeatMode", "DFS").V("3").repeat(out()).times(10).limit(1)
```

Dies folgt jeder einzelnen Lösung bis zur maximalen Tiefe, bevor die nächste Lösung untersucht wird.

Gremlin-Abfragehinweis noReordering

Beim Senden einer Gremlin-Traversierung untersucht die Neptune-Abfrage-Engine die Struktur der Traversierung und ordnet Teile der Abfrage neu an, um den Evaluierungsaufwand und die Antwortzeit für die Abfrage zu minimieren. Beispiel: Eine Traversierung mit mehreren Einschränkungen, wie z. B.mit mehreren `has()`-Schritten, wird in der Regel nicht in der angegebenen Reihenfolge

ausgewertet. Stattdessen wird ihre Anordnung geändert, nachdem die Abfrage mit statischen Analysen überprüft wurde.

Die Neptune-Abfrage-Engine versucht, die selektivere Einschränkung zu identifizieren, und führt diese zuerst aus. Dies führt häufig zu einer besseren Leistung. Die Reihenfolge der Evaluierung der Abfrage durch Neptune ist jedoch möglicherweise nicht immer optimal.

Wenn Sie die genauen Merkmale der Daten kennen und die Reihenfolge der Abfrageausführung manuell festlegen möchten, können Sie mit dem Neptune-Abfragehinweis `noReordering` angeben, dass die Traversierung in der angegebenen Reihenfolge evaluiert werden soll.

Syntax

Der `noReordering`-Abfragehinweis wird durch Hinzufügen eines `withSideEffect`-Schritts zu der Abfrage angegeben.

```
g.withSideEffect('Neptune#noReordering', true or false).gremlin-traversal
```

> **Note**
>
> Alle Gremlin-Abfragehinweis-Nebeneffekte haben das Präfix `Neptune#`.

Verfügbare Werte

- `true`
- `false`

Gremlin-Abfragehinweis typePromotion

Wenn Sie eine Gremlin-Traversierung absenden, die nach einem numerischen Wert oder Bereich filtert, muss die Neptune-Abfrage-Engine normalerweise den Typ heraufstufen, wenn sie die Abfrage ausführt. Das bedeutet, dass sie die Werte aller Typen untersuchen muss, die den Wert enthalten könnten, nach dem Sie filtern.

Wenn Sie beispielsweise nach Werten filtern, die gleich 55 sind, muss die Engine nach Ganzzahlen gleich 55, langen Ganzzahlen gleich 55L, Gleitkommazahlen gleich 55,0 usw. suchen. Jede Typheraufstufung erfordert eine zusätzliche Suche im Speicher. Dies kann dazu führen, dass eine scheinbar einfache Abfrage unerwartet lange dauert, bis sie abgeschlossen ist.

Angenommen, Sie suchen nach allen Eckpunkten, deren Eigenschaft „Kundenalter“ größer als 5 ist:

```
g.V().has('customerAge', gt(5))
```

Um diese Traversierung gründlich auszuführen, muss Neptune die Abfrage auf jeden numerischen Typ erweitern, zu dem der Wert, den Sie abfragen, heraufgestuft werden könnte. In diesem Fall muss der Filter `gt` auf jede Ganzzahl über 5, jede lange Ganzzahl über 5L, jede Gleitkommazahl über 5,0 und jede Doppelzahl über 5,0 angewendet werden. Da für jede dieser Typheraufstufungen eine zusätzliche Suche im Speicherplatz erforderlich ist, werden Ihnen pro numerischem Filter mehrere Filter angezeigt, wenn Sie Gremlin-`profile`-API für diese Abfrage ausführen. Daher dauert der Vorgang deutlich länger als erwartet.

Häufig ist eine Heraufstufung des Typs nicht notwendig, da Sie bereits wissen, dass Sie nur Werte eines bestimmten Typs suchen müssen. In diesem Fall können Sie Ihre Abfragen erheblich beschleunigen, indem Sie die Typheraufstufung mit dem Abfragehinweis `typePromotion` deaktivieren.

Syntax

Der `typePromotion`-Abfragehinweis wird durch Hinzufügen eines `withSideEffect`-Schritts zu der Abfrage angegeben.

```
g.withSideEffect('Neptune#typePromotion', true or false).gremlin-traversal
```

> **Note**
>
> Alle Gremlin-Abfragehinweis-Nebeneffekte haben das Präfix `Neptune#`.

Verfügbare Werte

- `true`
- `false`

Um die Typheraufstugung für Abfrage oben zu deaktivieren, würden Sie Folgendes verwenden:

```
g.withSideEffect('Neptune#typePromotion', false).V().has('customerAge', gt(5))
```

Gremlin-Abfragehinweis useDFE

Mit diesem Abfragehinweis können Sie die Verwendung der DFE zur Ausführung der Abfrage aktivieren. Standardmäßig verwendet Neptune die DFE nicht, wenn dieser Abfragehinweis nicht auf `true` festgelegt ist, da der Instance-Parameter neptune_dfe_query_engine standardmäßig auf `viaQueryHint` festgelegt ist. Wenn Sie diesen Instance-Parameter auf `enabled` festlegen, wird die DFE-Engine für alle Abfragen verwendet, außer Abfragen, bei denen der Abfragehinweis `useDFE` auf `false` festgelegt ist.

Beispiel für die Aktivierung der DFE für eine Abfrage:

```
g.withSideEffect('Neptune#useDFE', true).V().out()
```

Hinweis zur Gremlin-Evaluierungsabfrage TimeoutBehavior

Standardmäßig bricht Neptune die Abfrage ab und gibt einen Fehler zurück, wenn eine Gremlin-Abfrage das Abfragetimeout überschreitet. Der `evaluationTimeoutBehavior` Abfragehinweis ändert dieses Verhalten, sodass Neptune alle Teilergebnisse zurückstreamt, die vor dem Timeout berechnet wurden, anstatt einen Fehler zurückzugeben.

> **Note**
>
> Dieser Abfragehinweis verwendet den `with` Schritt`withSideEffect`, nicht und verwendet auch nicht das Präfix. `Neptune#`

Syntax

```
g.with('evaluationTimeoutBehavior', 'value').gremlin-traversal
```

Verfügbare Werte

- `partialResults`— Wenn bei der Abfrage ein Timeout auftritt, gibt Neptune alle Ergebnisse zurück, die vor dem Timeout berechnet wurden.
- `exception`— Wenn bei der Abfrage ein Timeout auftritt, gibt Neptune einen Fehler zurück. Dies ist das Standardverhalten.

> **Important**
>
> Dieser Abfragehinweis kann nur bei schreibgeschützten Abfragen verwendet werden. Wenn Sie diesen Hinweis mit einer mutierenden Abfrage oder mit aktiviertem DFE verwenden (entweder über den `useDFE` Abfragehinweis oder den `neptune_dfe_query_engine` Instanzparameter), gibt Neptune einen Fehler zurück.

Beispiel

Die folgende Abfrage gibt Teilergebnisse zurück, wenn sie das Abfrage-Timeout überschreitet:

```
g.with('evaluationTimeoutBehavior', 'partialResults').V().out().valueMap()
```

Gremlin-Abfragehinweise für die Verwendung des Ergebniscaches

Die folgenden Abfragehinweise können verwendet werden, wenn der Abfrage-Ergebniscache aktiviert ist.

`Gremlin: Abfragehinweis aktivieren ResultCache`

Der Abfragehinweis `enableResultCache` mit dem Wert `true` bewirkt die Rückgabe von Abfrageergebnissen aus dem Cache, wenn sie sich bereits im Cache befinden. Andernfalls werden neue Ergebnisse zurückgegeben und zwischengespeichert, bis sie aus dem Cache gelöscht werden. Beispiel:

```
g.with('Neptune#enableResultCache', true)
 .V().has('genre','drama').in('likes')
```

Später können Sie auf die zwischengespeicherten Ergebnisse zugreifen, indem Sie genau dieselbe Abfrage erneut ausführen.

Wenn der Wert dieses Abfragehinweises `false` ist oder nicht vorhanden ist, werden die Abfrageergebnisse nicht zwischengespeichert. Wenn Sie den Wert auf `false` festlegen, werden vorhandene zwischengespeicherte Ergebnisse jedoch nicht gelöscht. Verwenden Sie die Hinweise `invalidateResultCache` oder `invalidateResultCachekey`, um zwischengespeicherte Ergebnisse zu löschen.

Gremlin: **`Abfragehinweis aktivieren ResultCacheWithTTL`**

Der Abfragehinweis `enableResultCacheWithTTL` gibt auch zwischengespeicherte Ergebnisse zurück, wenn vorhanden, ohne dass sich dies auf die TTL der Ergebnisse im Cache auswirkt. Wenn es aktuell keine zwischengespeicherten Ergebnisse gibt, gibt die Abfrage neue Ergebnisse zurück und speichert sie für die im Abfragehinweis `enableResultCacheWithTTL` angegebene Dauer (Time To Live, TTL) im Cache. Diese Dauer wird in Sekunden angegeben. Die folgende Abfrage gibt beispielsweise eine Dauer von sechzig Sekunden an:

```
g.with('Neptune#enableResultCacheWithTTL', 60)
 .V().has('genre','drama').in('likes')
```

Vor Ablauf der Dauer von 60 Sekunden können Sie dieselbe Abfrage (hier `g.V().has('genre','drama').in('likes')`) mit den Abfragehinweisen `enableResultCache` oder `enableResultCacheWithTTL` verwenden, um auf die zwischengespeicherten Ergebnisse zuzugreifen.

> **Note**
>
> Die mit `enableResultCacheWithTTL` angegebene Dauer wirkt sich nicht auf bereits zwischengespeicherte Ergebnisse aus.
>
> - Wenn Ergebnisse zuvor mit `enableResultCache` zwischengespeichert wurden, muss der Cache zunächst explizit geleert werden, bevor `enableResultCacheWithTTL` neue Ergebnisse generiert und für die angegebene TTL zwischenspeichert.
> - Wenn Ergebnisse zuvor mit `enableResultCachewithTTL` zwischengespeichert wurden, muss die vorherige TTL zunächst ablaufen, bevor `enableResultCacheWithTTL` neue Ergebnisse generiert und für die angegebene TTL zwischenspeichert.

Nach Ablauf der Dauer werden die zwischengespeicherten Ergebnisse für die Abfrage gelöscht und eine folgende Instance derselben Abfrage gibt neue Ergebnisse zurück. Wenn `enableResultCacheWithTTL` an diese folgende Abfrage angefügt ist, werden die neuen Ergebnisse mit der angegebenen TTL zwischengespeichert.

Gremlin-Abfragehinweis **`ungültig machen ResultCacheKey`**

Der Abfragehinweis `invalidateResultCacheKey` kann den Wert `true` oder `false` annehmen. Der Wert `true` bewirkt, dass die zwischengespeicherten Ergebnisse für

die Abfrage gelöscht werden, an die `invalidateResultCacheKey` angefügt ist.
Das folgende Beispiel führt beispielsweise dazu, dass die für den Abfrageschlüssel `g.V().has('genre','drama').in('likes')` zwischengespeicherten Ergebnisse gelöscht werden:

```
g.with('Neptune#invalidateResultCacheKey', true)
 .V().has('genre','drama').in('likes')
```

Die Beispielabfrage oben bewirkt keine Zwischenspeicherung der neuen Ergebnisse. Sie können `enableResultCache` (oder`enableResultCacheWithTTL`) in dieselbe Abfrage einfügen, wenn Sie die neuen Ergebnisse nach dem Löschen der vorhandenen zwischengespeicherten Ergebnisse zwischenspeichern möchten:

```
g.with('Neptune#enableResultCache', true)
 .with('Neptune#invalidateResultCacheKey', true)
 .V().has('genre','drama').in('likes')
```

Gremlin macht den Abfragehinweis ungültig ResultCache

Der Abfragehinweis `invalidateResultCache` kann den Wert `true` oder `false` annehmen. Der Wert `true` bewirkt die Löschung aller Ergebnisse im Ergebniscache. Beispiel:

```
g.with('Neptune#invalidateResultCache', true)
 .V().has('genre','drama').in('likes')
```

Die Beispielabfrage oben bewirkt keine Zwischenspeicherung der Ergebnisse. Sie können `enableResultCache` (oder`enableResultCacheWithTTL`) in dieselbe Abfrage einfügen, wenn Sie nach dem vollständigen Leeren des vorhandenen Cache neue Ergebnisse zwischenspeichern möchten:

```
g.with('Neptune#enableResultCache', true)
 .with('Neptune#invalidateResultCache', true)
 .V().has('genre','drama').in('likes')
```

Hinweis zur Gremlin-Nummer-Abfrage ResultsCached

Der Abfragehinweis `numResultsCached` kann nur für Abfragen verwendet werden, die `iterate()` enthalten. Er gibt die maximale Anzahl von Ergebnissen an, die für die Abfrage

zwischengespeichert werden sollen, an die er angefügt ist. Die Ergebnisse, die zwischengespeichert werden, wenn `numResultsCached` vorhanden sind, werden nicht zurückgegeben, sondern lediglich zwischengespeichert.

Die folgende Abfrage gibt beispielsweise an, dass bis zu 100 ihrer Ergebnisse zwischengespeichert werden sollen, aber keines dieser zwischengespeicherten Ergebnisse zurückgegeben werden soll:

```
g.with('Neptune#enableResultCache', true)
 .with('Neptune#numResultsCached', 100)
 .V().has('genre','drama').in('likes').iterate()
```

Sie können dann eine Abfrage wie die folgende verwenden, um einen Bereich der zwischengespeicherten Ergebnisse (hier die ersten zehn) abzurufen:

```
g.with('Neptune#enableResultCache', true)
 .with('Neptune#numResultsCached', 100)
 .V().has('genre','drama').in('likes').range(0, 10)
```

Gremlin **`kein CacheExceptions`** Abfragehinweis

Der Abfragehinweis `noCacheExceptions` kann den Wert `true` oder `false` annehmen. Der Wert `true` bewirkt, dass alle Ausnahmen für den Ergebniscache unterdrückt werden. Beispiel:

```
g.with('Neptune#enableResultCache', true)
 .with('Neptune#noCacheExceptions', true)
 .V().has('genre','drama').in('likes')
```

Insbesondere unterdrückt dies die Ausnahme`QueryLimitExceededException`, die ausgelöst wird, wenn die Ergebnisse einer Abfrage zu groß für den Ergebniscache sind.

Gremlin-Abfragestatus-API

Sie können alle aktiven Gremlin-Abfragen auflisten oder den Status einer bestimmten Abfrage abrufen. Der zugrunde liegende HTTP-Endpunkt für beide Operationen ist. `https://`*`your-neptune-endpoint`*`:`*`port`*`/gremlin/status`

Liste der aktiven Gremlin-Abfragen

Um alle aktiven Gremlin-Abfragen aufzulisten, rufen Sie den Endpunkt ohne `queryId` Parameter auf.

Anforderungsparameter

- includeWaiting (optional) — Wenn auf gesetztTRUE, umfasst die Antwort neben laufenden Abfragen auch wartende Abfragen.

Antwortsyntax

```
{
  "acceptedQueryCount": integer,
  "runningQueryCount": integer,
  "queries": [
    {
      "queryId": "guid",
      "queryEvalStats": {
        "waited": integer,
        "elapsed": integer,
        "cancelled": boolean
      },
      "queryString": "string"
    }
  ]
}
```

- akzeptiert QueryCount — Die Anzahl der Abfragen, die akzeptiert, aber noch nicht abgeschlossen wurden, einschließlich Abfragen in der Warteschlange.
- laufend QueryCount — Die Anzahl der aktuell laufenden Gremlin-Abfragen.
- queries – Eine Liste der aktuellen Gremlin-Abfragen.

Beispiel

AWS CLI

```
aws neptunedata list-gremlin-queries \
  --endpoint-url https://your-neptune-endpoint:port
```

Weitere Informationen finden Sie unter list-gremlin-queries in der Befehlsreferenz. 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)
```

AWS SDK-Beispiele in anderen Sprachen wie Java, .NET und mehr finden Sie unter. AWS SDK

awscurl

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

> **Note**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

Weitere Informationen zur Verwendung awscurl mit der IAM-Authentifizierung finden Sie unter. Verwenden von `awscurl` mit temporären Anmeldeinformationen, um eine sichere Verbindung zu einem DB-Cluster mit aktivierter IAM-Authentifizierung herzustellen

curl

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

Die folgende Ausgabe zeigt eine einzelne laufende Abfrage.

```
{
  "acceptedQueryCount": 9,
  "runningQueryCount": 1,
```

```
  "queries": [
    {
      "queryId": "fb34cd3e-f37c-4d12-9cf2-03bb741bf54f",
      "queryEvalStats": {
        "waited": 0,
        "elapsed": 23,
        "cancelled": false
      },
      "queryString": "g.V().out().count()"
    }
  ]
}
```

Den Status einer bestimmten Gremlin-Abfrage abrufen

Um den Status einer bestimmten Gremlin-Abfrage abzurufen, geben Sie den Parameter an`queryId`.

Anforderungsparameter

- queryId (erforderlich) — Die ID der Gremlin-Abfrage. Neptune weist diesen ID-Wert automatisch jeder Abfrage zu, oder Sie können Ihre eigene ID zuweisen (siehe). Einfügen einer benutzerdefinierten ID in eine Neptune-Gremlin- oder -SPARQL-Abfrage

Antwortsyntax

```
{
  "queryId": "guid",
  "queryString": "string",
  "queryEvalStats": {
    "waited": integer,
    "elapsed": integer,
    "cancelled": boolean,
    "subqueries": document
  }
}
```

- queryId — Die ID der Abfrage.
- queryString – Die übermittelte Abfrage. Die Abfrage wird nach 1024 Zeichen abgeschnitten, wenn sie länger ist.

- query EvalStats — Statistiken für die Abfrage, einschließlich `waited` (Wartezeit in Millisekunden), `elapsed` (Laufzeit in Millisekunden), (ob die Abfrage abgebrochen wurde) und `cancelled` (Anzahl der Unterabfragen). `subqueries`

Beispiel

AWS CLI

```
aws neptunedata get-gremlin-query-status \
  --endpoint-url https://your-neptune-endpoint:port \
  --query-id "fb34cd3e-f37c-4d12-9cf2-03bb741bf54f"
```

Weitere Informationen finden Sie unter get-gremlin-query-status in der Befehlsreferenz. 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)
```

AWS SDK-Beispiele in anderen Sprachen wie Java, .NET und mehr finden Sie unter. AWS SDK

awscurl

```
awscurl https://your-neptune-endpoint:port/gremlin/status/fb34cd3e-
f37c-4d12-9cf2-03bb741bf54f \
  --region us-east-1 \
  --service neptune-db
```

> **Note**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

Weitere Informationen zur Verwendung awscurl mit der IAM-Authentifizierung finden Sie unter. Verwenden von awscurl mit temporären Anmeldeinformationen, um eine sichere Verbindung zu einem DB-Cluster mit aktivierter IAM-Authentifizierung herzustellen

curl

```
curl https://your-neptune-endpoint:port/gremlin/status/fb34cd3e-
f37c-4d12-9cf2-03bb741bf54f
```

Nachfolgend finden Sie eine Beispielantwort.

```
{
  "queryId": "fb34cd3e-f37c-4d12-9cf2-03bb741bf54f",
  "queryString": "g.V().out().count()",
  "queryEvalStats": {
    "waited": 0,
    "elapsed": 23,
    "cancelled": false
  }
}
```

Gremlin-Abfrageabbruch

Zum Abrufen des Status von Gremlin-Abfragen verwenden Sie HTTP `GET` oder `POST`, um eine Anforderung an den `https://your-neptune-endpoint:port/gremlin/status`-Endpunkt zu senden.

Gremlin-Abfrageabbruch-Anforderungsparameter

- cancelQuery – Erforderlich für den Abbruch. Für diesen Parameter gibt es keinen entsprechenden Wert.
- queryId – Die ID der ausgeführten Gremlin-Abfrage, die abgebrochen werden soll.

Gremlin-Abfrageabbruch-Beispiel

Im Folgenden finden Sie ein Beispiel für das Abbrechen einer Abfrage.

AWS CLI

```
aws neptunedata cancel-gremlin-query \
  --endpoint-url https://your-neptune-endpoint:port \
  --query-id "fb34cd3e-f37c-4d12-9cf2-03bb741bf54f"
```

Weitere Informationen finden Sie unter cancel-gremlin-query in der Befehlsreferenz. 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)
```

AWS SDK-Beispiele in anderen Sprachen wie Java, .NET und mehr finden Sie unter. AWS 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**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

Weitere Informationen zur Verwendung awscurl mit der IAM-Authentifizierung finden Sie unter. Verwenden von awscurl mit temporären Anmeldeinformationen, um eine sichere Verbindung zu einem DB-Cluster mit aktivierter IAM-Authentifizierung herzustellen

curl

```
curl https://your-neptune-endpoint:port/gremlin/status \
  --data-urlencode "cancelQuery" \
  --data-urlencode "queryId=fb34cd3e-f37c-4d12-9cf2-03bb741bf54f"
```

Wenn der Abbruch erfolgreich ist, gibt HTTP 200 OK zurück.

Unterstützung für skriptbasierte Gremlin-Sitzungen

Sie können Gremlin-Sitzungen mit impliziten Transaktionen in Amazon Neptune verwenden. Informationen zu Gremlin-Sitzungen finden Sie in der Apache-Dokumentation unter Considering Sessions. TinkerPop Die folgenden Abschnitte beschreiben die Verwendung von Gremlin-Sitzungen mit Java.

> **Important**
>
> Zurzeit kann Neptune eine skriptbasierte Sitzung maximal 10 Minuten geöffnet lassen. Wenn Sie eine Sitzung nicht eher schließen, findet eine Zeitüberschreitung der Sitzung statt und alle ihre Einstellungen werden zurückgesetzt.

Themen

- Gremlin-Sitzungen in der Gremlin-Konsole
- Gremlin-Sitzungen in der Gremlin Language Variant

Gremlin-Sitzungen in der Gremlin-Konsole

Wenn Sie eine Remote-Verbindung auf der Gremlin-Konsole ohne den `session`-Parameter erstellen, wird die Remote-Verbindung im sitzungslosen Modus erstellt. In diesem Modus wird jede Anforderung, die an den Server gesendet wird, als eine vollständige Transaktion an sich behandelt, und zwischen den Anforderungen wird kein Status gespeichert. Wenn eine Anforderung fehlschlägt, wird nur die betreffende Anforderung zurückgesetzt.

Wenn Sie eine Remote-Verbindung mit dem `session`-Parameter erstellen, dann erstellen Sie eine skriptbasierte Sitzung, die solange dauert, bis Sie die Remote-Verbindung beenden. Jede Sitzung wird durch eine eindeutige UUID identifiziert, die die Konsole generiert und an Sie zurückgibt.

Es folgt ein Beispiel für einen Konsolenaufruf, der eine Sitzung erstellt. Nachdem Abfragen übermittelt wurden, schließt ein anderer Aufruf die Sitzung und führt ein Commit für die Abfragen durch.

> **Note**
>
> Der Gremlin-Client muss immer geschlossen sein, um serverseitige Ressourcen freigeben zu können.

```
gremlin> :remote connect tinkerpop.server conf/neptune-remote.yaml session
  . . .
  . . .
gremlin> :remote close
```

Weitere Informationen und Beispiele finden Sie in der Dokumentation unter Sessions. TinkerPop

Alle Abfragen, die Sie während einer Sitzung ausführen, bilden eine einzelne Transaktion, für die erst ein Commit durchgeführt wird, wenn alle Abfragen erfolgreich sind und Sie die Remote-Verbindung schließen. Wenn eine Abfrage fehlschlägt oder wenn Sie die Verbindung nicht innerhalb der von Neptune maximal unterstützten Sitzungszeit schließen, wird kein Commit der Sitzungstransaktion durchgeführt und alle enthaltenen Abfragen werden zurückgesetzt.

Gremlin-Sitzungen in der Gremlin Language Variant

Sie müssen in der Gremlin Language Variant (GLV) ein `SessionedClient`-Objekt erstellen, damit in einer einzelnen Transaktion mehrere Abfragen ausgegeben werden, vgl. das folgende Beispiel.

```
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();
}
```

Zeile 3 im Beispiel oben erstellt das `SessionedClient`-Objekt entsprechend den Konfigurationsoptionen, die für den betreffenden Cluster festgelegt wurden. Die *`sessionName`* Zeichenfolge, die Sie an die Connect-Methode übergeben, wird zum eindeutigen Namen der Sitzung. Um Kollisionen zu vermeiden, verwenden Sie eine UUID für den Namen.

Der Client startet eine Sitzungstransaktion, wenn er initialisiert wird. Für alle Abfragen, die Sie im Rahmen des Sitzungsformulars ausführen, werden nur Commits durchgeführt, wenn Sie `client.close( )` aufrufen. Nochmals: Wenn eine einzelne Abfrage fehlschlägt oder wenn Sie die Verbindung nicht innerhalb der von Neptune maximal unterstützten Sitzungszeit schließen, schlägt die Sitzungstransaktion fehl und alle enthaltenen Abfragen werden zurückgesetzt.

> **Note**
>
> Der Gremlin-Client muss immer geschlossen sein, um serverseitige Ressourcen freigeben zu können.

```
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();
    }
}
```

Gremlin-Transaktionen in Neptune

Es gibt verschiedene Kontexte, in denen Gremlin-Transaktionen ausgeführt werden. Bei der Arbeit mit Gremlin ist es wichtig, den Kontext, in dem Sie arbeiten, und dessen Auswirkungen zu verstehen:

- **`Script-based`** – Anforderungen werden mit textbasierten Gremlin-Zeichenfolgen gesendet, z. B.:
  - Verwenden von Java-Treiber und `Client.submit(`*`string`*`)`.
  - Verwenden von Gremlin-Konsole und `:remote connect`.
  - Verwenden der HTTP-API.
- **`Bytecode-based`** – Anforderungen werden mit einem serialisierten Gremlin-Bytecode gesendet, der für Gremlin Language Variants (GLV) typisch ist.

  Zum Beispiel mit dem Java-Treiber, `g = traversal().withRemote(...)`.

Für jeden der oben genannten Kontexte gibt es zusätzlichen Kontext – ob die Anforderung als sitzungslos oder sitzungsgebunden gesendet wird.

> **Note**
>
> Für Gremlin-Transaktionen muss stets entweder ein Commit oder ein Rollback ausgeführt werden, damit serverseitige Ressourcen freigegeben werden können. Im Falle eines Fehlers während der Transaktion ist es wichtig, die gesamte Transaktion erneut zu versuchen und nicht nur die einzelne Anfrage, die fehlgeschlagen ist.

Anforderungen, die nicht an eine Sitzung gebunden sind

Wenn eine Anforderung sitzungslos ist, entspricht eine Anforderung einer einzelnen Transaktion.

Im Fall von Skripts bedeutet dies, dass eine oder mehrere Gremlin-Anweisungen, die in einer einzelnen Anforderung gesendet werden, das Commit oder Rollback als einzelne Transaktion ausführen. Beispiel:

```
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();
```

Im Fall von Bytecode wird für jede Traversierung, die über g ausgelöst und ausgeführt wird, eine sitzungslose Anforderung gesendet.

```
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();
```

Anforderungen, die an eine Sitzung gebunden sind

Wenn Anforderungen an eine Sitzung gebunden sind, können mehrere Anforderungen im Kontext einer einzelnen Transaktion gesendet werden.

Im Fall von Skripts bedeutet dies, dass Diagrammoperationen nicht alle zu einem einzelnen eingebetteten Zeichenfolgenwert verkettet werden müssen:

```
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();
}
```

Bei skriptbasierten Sitzungen wird beim Schließen des Clients mit die Transaktion `client.close()` festgeschrieben. In skriptbasierten Sitzungen ist kein expliziter Rollback-Befehl verfügbar. Um ein Rollback zu erzwingen, können Sie die Transaktion zum Scheitern bringen, indem Sie beispielsweise `g.inject(0).fail('rollback')` vor dem Schließen des Clients eine Abfrage ausgeben.

> **Note**
>
> Eine Abfrage wieg`.inject(0).fail('rollback')`, die verwendet wird, um absichtlich einen Fehler auszulösen, um ein Rollback zu erzwingen, erzeugt eine Ausnahme auf dem Client. Fangen Sie die resultierende Ausnahme ab und verwerfen Sie sie, bevor Sie den Client schließen.

Bei Bytecode kann die Transaktion explizit gesteuert und die Sitzung transparent verwaltet werden. Gremlin Language Variants (GLV) unterstützen die Gremlin-Syntax `tx()` für den `commit()` oder `rollback()` einer Transaktion wie folgt:

```
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();
    }
}
```

Obwohl das obige Beispiel in Java geschrieben ist, können Sie diese `tx()` Syntax auch in anderen Sprachen verwenden. Eine sprachspezifische Transaktionssyntax finden Sie im Abschnitt Transaktionen der TinkerPop Apache-Dokumentation für Java, Python, Javascript, .NET und Go.

> **Warning**
>
> Sitzungslose schreibgeschützte Abfragen werden unter SNAPSHOT-Isolation ausgeführt. Schreibgeschützte Abfragen innerhalb einer expliziten Transaktion werden unter SERIALIZABLE-Isolation ausgeführt. Die schreibgeschützten Abfragen unter `SERIALIZABLE`-Isolation führen zu einem höheren Overhead und können im Gegensatz zu isolierten Abfragen unter `SNAPSHOT` gleichzeitige Schreibvorgänge blockieren oder von diesen blockiert werden.

Timeout-Verhalten bei Bytecode-Commit und -Rollback

Wenn Sie Bytecode-basierte Transaktionen mit dieser `tx()` Syntax verwenden, unterliegen die `rollback()` Operationen `commit()` und nicht den Timeout-Einstellungen für Abfragen. Weder der globale `neptune_query_timeout` Parameter noch die Timeout-Werte pro Abfrage, die durch festgelegt wurden, gelten für diese Operationen. `evaluationTimeout` Auf dem Server `commit()` und ohne Zeitlimit `rollback()` ausführen, bis sie abgeschlossen sind oder ein Fehler auftritt.

Auf der Client-Seite werden die Gremlin-Treiber `tx.commit()` und die `tx.rollback()` Aufrufe erst abgeschlossen, wenn der Server antwortet. Je nach Sprache kann sich dies als blockierender Anruf oder als ungelöster asynchroner Vorgang äußern. Kein Treiber bietet eine integrierte Timeout-Einstellung, die diese Aufrufe begrenzt. Einzelheiten zum Parallelitätsverhalten im Zusammenhang mit diesen Transaktionsfunktionen finden Sie in der API-Dokumentation für Ihre spezielle Gremlin-Sprachvariante.

> **Important**
>
> Wenn ein `commit()` `rollback()` OR-Aufruf länger als erwartet dauert, wird er möglicherweise durch einen Sperrkonflikt bei einer gleichzeitigen Transaktion blockiert. Weitere Hinweise zu Sperrkonflikten finden Sie unter. Konfliktlösung mithilfe von Timeouts Lock-Wait

Wenn Sie die Zeit, in der Ihre Anwendung auf ein `commit()` Oder wartet, einschränken müssen`rollback()`, können Sie die Parallelitätsfunktionen Ihrer Sprache verwenden, um ein clientseitiges Timeout festzulegen. Wenn das clientseitige Timeout ausgelöst wird, setzt der Server die Verarbeitung des Vorgangs fort. Der serverseitige Vorgang enthält einen Worker-Thread, bis er abgeschlossen ist. Schließen Sie nach einem clientseitigen Timeout die Verbindung und erstellen Sie eine neue, anstatt die bestehende Verbindung wiederzuverwenden, da der Transaktionsstatus unbestimmt ist.

Server-side Bereinigung der Transaktion

Wenn ein Client eine Transaktion unterbricht oder abbricht, ohne einen Commit oder Rollback vorzunehmen, verfügt Neptune über serverseitige Mechanismen, die die verwaiste Transaktion irgendwann bereinigen:

- Sitzungs-Timeout — Bytecode-based Sitzungen, die länger als die maximale Sitzungsdauer (10 Minuten) inaktiv bleiben, werden geschlossen, und jede offene Transaktion wird zurückgesetzt.
- Timeout bei Verbindungsinaktivität — Neptune schließt WebSocket Verbindungen, die etwa 20 Minuten lang inaktiv waren. Wenn die Verbindung geschlossen wird, macht der Server alle offenen Transaktionen, die mit dieser Verbindung verknüpft sind, rückgängig.

Diese Säuberungsmechanismen sind Sicherheitsnetze. Wir empfehlen Ihnen, Transaktionen immer dann explizit festzuschreiben oder rückgängig zu machen, wenn Sie mit ihnen fertig sind.

Streaming-Abfrageergebnisse mit Gremlin

Wenn Sie eine Gremlin-Traversal ausführen, die eine große Anzahl von Ergebnissen zurückgibt, streamt Neptune diese stapelweise über die Verbindung zurück an den Client. WebSocket Neptune sendet Ergebnisstapel, sobald sie produziert werden, ohne darauf zu warten, dass der Kunde mehr anfordert. Dies kann von Vorteil sein, wenn Sie Ergebnisse so verarbeiten möchten, wie sie vom Server zurückgegeben werden, erfordert jedoch die Verwendung von verzögerten Iterationsmustern, um zu vermeiden, dass die gesamte Ergebnismenge im Speicher gesammelt wird.

Neptune sendet standardmäßig Ergebnisse in Batches von 64 pro WebSocket Frame. Sie können diese serverseitige Standardeinstellung nicht ändern, aber die Batchgröße kann pro Anfrage vom Client mithilfe der Request-Option (aufgerufen `Tokens.ARGS_BATCH_SIZE` im Java-Treiber oder `batchSize`als Standard auf Treiberebene) außer Kraft gesetzt werden. `connectionPool.resultIterationBatchSize`

Einzelheiten zur Konfiguration von Treibern `batchSize` in anderen Sprachen finden Sie im Abschnitt Konfiguration für jeden Treiber in der Dokumentation zu Apache TinkerPop Gremlin Drivers and Variants.

Da der Server die Ergebnisse automatisch weiterleitet, wird der clientseitige Gegendruck implizit über TCP und Flusssteuerung abgewickelt. WebSocket Wenn der Client langsam aus dem Socket liest, werden die Schreibvorgänge des Servers irgendwann blockiert, bis der Client aufholt.

> **Important**
>
> Streaming ist am effektivsten bei Durchläufen, bei denen schrittweise Ergebnisse erzielt werden können. Durchläufe, die`order()`,,`groupCount()`, oder andere Schritte beinhalten `group()``dedup()`, bei denen die vollständige Durchquerung abgeschlossen sein muss, bevor Ergebnisse ausgegeben werden, veranlassen Neptune, die gesamte Ergebnismenge im Speicher zu materialisieren, bevor das Streaming beginnt. In diesen Fällen reduziert die Batchverarbeitung immer noch den Aufwand für die Serialisierung pro Frame, reduziert jedoch nicht die serverseitige Speichernutzung.

Schrittweise Nutzung der Ergebnisse

Um Ergebnisse sofort zu verarbeiten, iterieren Sie langsam mithilfe von`hasNext()`/`next()`oder gleichwertigen APIs, anstatt alle Ergebnisse in einer Liste zu sammeln. Sie können sie verwenden`next(batchSize)`, um Ergebnisse in Batches auf Anwendungsebene abzurufen, sodass Sie Zwischenarbeiten zwischen den Batches durchführen können, während der Server weiterhin Ergebnisse produziert.

Example Java (GLV-Bytecode)

```
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`);
}
```

Eifriger Konsum versus schrittweiser Konsum

Streaming ermöglicht es Ihnen, Ergebnisse inkrementell zu verarbeiten, wenn zusätzliche Daten abgerufen und zurückgegeben werden. Die folgenden Methoden blockieren, bis die gesamte Ergebnismenge im Speicher erfasst ist. Dadurch wird verhindert, dass Ihre Anwendung auf die Ergebnisse reagiert, sobald sie eintreffen:

- Java: `toList()` oder `toSet()`
- Python: `toList()` oder `toSet()`
- Gehe zu:`ToList()`,`ToSet()`, oder `GetResultSet().GetAll()`
- .NET: `ToList()` oder `Promise()`
- Node.js: `toList()`

> **Note**
>
> Daten fließen immer noch inkrementell über die WebSocket Verbindung, selbst wenn diese Methoden verwendet werden. Der Unterschied besteht darin, dass Ihre Anwendung einzelne Ergebnisse erst verarbeiten kann, wenn die gesamte Erfassung abgeschlossen ist. Verwenden Sie die in den obigen Beispielen gezeigten Lazy-Iterations- oder Batch-Muster, um die Ergebnisse sofort zu verarbeiten.

Verwenden der Gremlin-API mit Amazon Neptune

> **Note**
>
> Amazon Neptune unterstützt die Eigenschaft `bindings` nicht.

Alle HTTPS-Anforderungen von Gremlin verwenden einen einzelnen Endpunkt: `https://`*`your-neptune-endpoint`*`:`*`port`*`/gremlin`. Alle Neptune-Verbindungen müssen HTTPS verwenden.

Du kannst die Gremlin Console direkt über mit einem Neptun-Graphen verbinden. WebSockets

Weitere Informationen zum Herstellen einer Verbindung zum Gremlin-Endpunkt finden Sie unter Zugriff auf ein Neptune-Diagramm mit Gremlin.

Die Amazon-Neptune-Implementierung von Gremlin umfasst spezifische Details und Unterschiede, die Sie berücksichtigen müssen. Weitere Informationen finden Sie unter Einhaltung der Gremlin-Standards in Amazon Neptune.

Informationen zur Gremlin-Sprache und zu Traversals finden Sie in der Apache-Dokumentation unter The Traversal. TinkerPop

Zwischenspeichern von Abfrageergebnissen in Amazon Neptune Gremlin

Amazon Neptune unterstützt einen Ergebnis-Cache für Gremlin-Abfragen.

Sie können den Abfrageergebnis-Cache aktivieren und dann einen Abfragehinweis verwenden, um die Ergebnisse einer schreibgeschützten Gremlin-Abfrage zwischenzuspeichern.

Bei jeder erneuten Ausführung der Abfrage werden die zwischengespeicherten Ergebnisse mit geringer Latenz und ohne I/O Kosten abgerufen, sofern sie sich noch im Cache befinden. Dies funktioniert für Abfragen, die über HTTP-Endpunkte oder Websockets als Bytecode oder Zeichenfolgen gesendet werden.

> **Note**
>
> An den Profilendpunkt gesendete Abfragen werden nicht zwischengespeichert, auch wenn der Abfrage-Cache aktiviert ist.

Sie können das Verhalten des Neptune-Abfrageergebnis-Cache auf verschiedene Arten steuern. Beispiel:

- Sie können zwischengespeicherte Ergebnisse in Blöcken paginiert abrufen.
- Sie können die Dauer (Time-to-Live, TTL) für bestimmte Abfragen angeben.
- Sie können den Cache für bestimmte Abfragen leeren.
- Sie können den gesamten Cache löschen.
- Sie können Benachrichtigungen für den Fall einrichten, dass die Ergebnisse die Cachegröße überschreiten.

Der Cache wird mithilfe einer Least-Recently-Used (LRU)-Richtlinie verwaltet. Sobald der dem Cache zugewiesene Speicherplatz voll ist, werden daher die ältesten verwendeten Ergebnisse entfernt, um Platz für die Zwischenspeicherung neuer Ergebnisse zu schaffen.

> **Important**
>
> Der Abfrageergebnis-Cache ist für die Instance-Typen `t3.medium` oder `t4.medium` nicht verfügbar.

Aktivieren des Abfrageergebnis-Cache in Neptune

Der Cache für Abfrageergebnisse kann für alle Instanzen in einem Cluster oder pro Instanz aktiviert werden. Um den Ergebnis-Cache für alle Instances in einem Cluster zu aktivieren, setzen Sie den `neptune_result_cache` Parameter im Cluster `cluster-parameter-group` auf`1`. Um dies für eine bestimmte Instance zu aktivieren, setzen Sie den `neptune_result_cache` Parameter in der Instanz `instance-parameter-group` auf`1`. Die Einstellung für die Cluster-Parametergruppe überschreibt den Wert der Instanzparametergruppe.

Auf allen betroffenen Instances ist ein Neustart erforderlich, damit die Parametereinstellungen für den Ergebnis-Cache angewendet werden. Sie können den Ergebnis-Cache zwar über für alle Instances in einem Cluster aktivieren`cluster-parameter-group`, aber jede Instanz verwaltet ihren eigenen Cache. Bei der Cache-Funktion für Abfrageergebnisse handelt es sich nicht um einen clusterweiten Cache.

Nach Aktivierung des Ergebnis-Caches reserviert Neptune einen Teil des aktuellen Arbeitsspeichers für das Zwischenspeichern von Abfrageergebnissen. Je größer der verwendete Instance-Typ und je größer der verfügbare Speicher, desto mehr Speicher reserviert Neptune für den Cache.

Wenn der für den Ergebnis-Cache reservierte Arbeitsspeicher gefüllt wird, löscht Neptune automatisch die ältesten verwendeten zwischengespeicherten Ergebnisse, um Platz für neue zu schaffen.

Sie können den aktuellen Status des Ergebniscaches mithilfe des Befehls Instance-Status überprüfen.

Verwenden von Hinweisen zum Zwischenspeichern von Abfrageergebnissen

Nach der Aktivierung des Abfrageergebnis-Caches können Sie das Zwischenspeichern von Abfragen mittels Abfragehinweisen steuern. Alle folgenden Beispiele beziehen sich auf diese Abfragetraversierung:

```
g.V().has('genre','drama').in('likes')
```

Verwenden von „Aktivieren“ ResultCache

Nach der Aktivierung des Abfrageergebnis-Caches können Sie die Ergebnisse einer Gremlin-Abfrage über den Abfragehinweis `enableResultCache` wie folgt zwischenspeichern:

```
g.with('Neptune#enableResultCache', true)
 .V().has('genre','drama').in('likes')
```

Neptune gibt anschließend die Abfrageergebnisse zurück und speichert diese auch im Cache. Später können Sie auf die zwischengespeicherten Ergebnisse zugreifen, indem Sie genau dieselbe Abfrage erneut ausführen.

```
g.with('Neptune#enableResultCache', true)
 .V().has('genre','drama').in('likes')
```

Der Cache-Schlüssel, der die zwischengespeicherten Ergebnisse identifiziert, ist die Abfragezeichenfolge selbst:

```
g.V().has('genre','drama').in('likes')
```

Verwenden von `enable ResultCacheWithTTL`

Über den Abfragehinweis `enableResultCacheWithTTL` können Sie angeben, wie lange die Abfrageergebnisse zwischengespeichert werden sollen. Die folgende Abfrage gibt beispielsweise an, dass die Abfrageergebnisse nach 120 Sekunden ablaufen sollen:

```
g.with('Neptune#enableResultCacheWithTTL', 120)
 .V().has('genre','drama').in('likes')
```

Der Cache-Schlüssel, der die zwischengespeicherten Ergebnisse identifiziert, ist auch hier die Abfragezeichenfolge selbst:

```
g.V().has('genre','drama').in('likes')
```

Und auch hier können Sie auf die zwischengespeicherten Ergebnisse über diese Abfragezeichenfolge und den Abfragehinweis `enableResultCache` zugreifen:

```
g.with('Neptune#enableResultCache', true)
 .V().has('genre','drama').in('likes')
```

Wenn seit dem Zwischenspeichern der Ergebnisse 120 oder mehr Sekunden vergangen sind, gibt diese Abfrage neue Ergebnisse zurück und speichert sie ohne Angabe einer Dauer im Zwischenspeicher.

Sie können auf die zwischengespeicherten Ergebnisse auch über dieselbe Abfrage mit dem Abfragehinweis `enableResultCacheWithTTL` zugreifen. Beispiel:

```
g.with('Neptune#enableResultCacheWithTTL', 140)
 .V().has('genre','drama').in('likes')
```

Bis zum Ablauf von 120 Sekunden (d. h. der aktuell gültigen TTL) gibt diese neue Abfrage unter Verwendung des Abfragehinweises `enableResultCacheWithTTL` die zwischengespeicherten Ergebnisse zurück. Nach 120 Sekunden würde sie neue Ergebnisse zurückgeben und diese für die Dauer von 140 Sekunden zwischenspeichern.

> **Note**
>
> Wenn die Ergebnisse für einen Abfrageschlüssel bereits zwischengespeichert sind, generiert derselbe Abfrageschlüssel mit `enableResultCacheWithTTL` keine neuen Ergebnisse

und hat keine Auswirkung auf die Dauer der Speicherung der aktuell zwischengespeicherten Ergebnisse.

- Wenn bereits Ergebnisse mit `enableResultCache` zwischengespeichert wurden, muss der Cache zunächst geleert werden, bevor `enableResultCacheWithTTL` neue Ergebnisse generiert und für die angegebene TTL zwischenspeichert.
- Wenn Ergebnisse zuvor mit `enableResultCachewithTTL` zwischengespeichert wurden, muss die vorherige TTL zunächst ablaufen, bevor `enableResultCacheWithTTL` neue Ergebnisse generiert und für die angegebene TTL zwischenspeichert.

Verwenden von **`invalidate ResultCacheKey`**

Sie können mittels des Abfragehinweises `invalidateResultCacheKey` zwischengespeicherte Ergebnisse für eine bestimmte Abfrage löschen. Beispiel:

```
g.with('Neptune#invalidateResultCacheKey', true)
 .V().has('genre','drama').in('likes')
```

Diese Abfrage löscht den Cache für den Abfrageschlüssel `g.V().has('genre','drama').in('likes')` und gibt neue Ergebnisse für diese Abfrage zurück.

Sie können `invalidateResultCacheKey` auch mit `enableResultCache` oder `enableResultCacheWithTTL` kombinieren. Die folgende Abfrage löscht beispielsweise die aktuellen zwischengespeicherten Ergebnisse, speichert neue Ergebnisse im Cache und gibt diese zurück:

```
g.with('Neptune#enableResultCache', true)
 .with('Neptune#invalidateResultCacheKey', true)
 .V().has('genre','drama').in('likes')
```

Verwenden von invalidate ResultCache

Sie können mittels des Abfragehinweises `invalidateResultCache` alle zwischengespeicherten Ergebnisse im Abfrageergebnis-Cache löschen. Beispiel:

```
g.with('Neptune#invalidateResultCache', true)
```

```
  .V().has('genre','drama').in('likes')
```

Diese Abfrage löscht den gesamten Ergebnis-Cache und gibt neue Ergebnisse für die Abfrage zurück.

Sie können `invalidateResultCache` auch mit `enableResultCache` oder `enableResultCacheWithTTL` kombinieren. Die folgende Abfrage löscht beispielsweise den gesamten aktuellen Ergebnis-Cache, speichert neue Ergebnisse für diese Abfrage im Cache und gibt diese zurück:

```
g.with('Neptune#enableResultCache', true)
 .with('Neptune#invalidateResultCache', true)
 .V().has('genre','drama').in('likes')
```

Paginieren zwischengespeicherter Abfrageergebnisse

Angenommen, Sie haben bereits eine große Anzahl von Ergebnissen wie folgt zwischengespeichert:

```
g.with('Neptune#enableResultCache', true)
 .V().has('genre','drama').in('likes')
```

Nehmen Sie nun an, Sie geben die folgende Bereichsabfrage aus:

```
g.with('Neptune#enableResultCache', true)
 .V().has('genre','drama').in('likes').range(0,10)
```

Neptune sucht zunächst nach dem vollständigen Cache-Schlüssel, `g.V().has('genre','drama').in('likes').range(0,10)`. Wenn dieser Schlüssel nicht vorhanden ist, sucht Neptune als Nächstes nach einem Schlüssel für diese Abfragezeichenfolge ohne Bereich (`g.V().has('genre','drama').in('likes')`). Wenn dieser Schlüssel vorhanden ist, ruft Neptune die ersten zehn Ergebnisse aus dem Cache ab wie vom Bereich angegeben.

> **Note**
>
> Wenn Sie den Abfragehinweis `invalidateResultCacheKey` mit einer Abfrage mit Bereich am Ende verwenden, löscht Neptune den Cache für eine Abfrage ohne diesen Bereich, wenn es keine exakte Übereinstimmung für die Abfrage mit Bereich gibt.

Num ResultsCached mit **.iterate** () verwenden

Mittels des Abfragehinweises `numResultsCached` können Sie den Ergebniscache auffüllen, ohne alle zwischengespeicherten Ergebnisse zurückzugeben. Dies kann nützlich sein, wenn Sie eine große Anzahl von Ergebnissen paginieren möchten.

Der Abfragehinweis `numResultsCached` funktioniert nur bei Abfragen, die mit `iterate()` enden.

Wenn Sie beispielsweise die ersten 50 Ergebnisse der Beispielabfrage zwischenspeichern möchten:

```
g.with("Neptune#enableResultCache", true)
 .with("Neptune#numResultsCached", 50)
 .V().has('genre','drama').in('likes').iterate()
```

In diesem Fall ist der Abfrageschlüssel im Cache: `g.with("Neptune#numResultsCached", 50).V().has('genre','drama').in('likes')`. Sie können jetzt die ersten zehn der zwischengespeicherten Ergebnisse mit dieser Abfrage abrufen:

```
g.with("Neptune#enableResultCache", true)
 .with("Neptune#numResultsCached", 50)
 .V().has('genre','drama').in('likes').range(0, 10)
```

Sie können die nächsten zehn Ergebnisse aus der Abfrage wie folgt abrufen:

```
g.with("Neptune#enableResultCache", true)
 .with("Neptune#numResultsCached", 50)
 .V().has('genre','drama').in('likes').range(10, 20)
```

Vergessen Sie nicht, den Abfragehinweis `numResultsCached` einzufügen! Dieser ist ein wesentlicher Bestandteil des Abfrageschlüssels und muss daher vorhanden sein, um auf die zwischengespeicherten Ergebnisse zugreifen zu können.

Einige Dinge, die Sie bei der Verwendung von **Num** beachten sollten ResultsCached

- Die Zahl, die Sie mit **numResultsCached** angeben, wird am Ende der Abfrage angewendet. Das bedeutet zum Beispiel, dass die folgende Abfrage tatsächlich Ergebnisse im Bereich `(1000, 1500)` zwischenspeichert:

  ```
  g.with("Neptune#enableResultCache", true)
  ```

```
 .with("Neptune#numResultsCached", 500)
 .V().range(1000, 2000).iterate()
```

- Die Zahl, die Sie mit **`numResultsCached`** angeben, gibt die maximale Anzahl von Ergebnissen an, die zwischengespeichert werden sollen. Das bedeutet zum Beispiel, dass die folgende Abfrage tatsächlich Ergebnisse im Bereich `(1000, 2000)` zwischenspeichert:

  ```
  g.with("Neptune#enableResultCache", true)
   .with("Neptune#numResultsCached", 100000)
   .V().range(1000, 2000).iterate()
  ```

- Ergebnisse, die von Abfragen zwischengespeichert werden, die mit **`.range().iterate()`** enden, haben einen eigenen Bereich. Angenommen, Sie speichern Ergebnisse mithilfe einer Abfrage wie der folgenden im Cache:

  ```
  g.with("Neptune#enableResultCache", true)
   .with("Neptune#numResultsCached", 500)
   .V().range(1000, 2000).iterate()
  ```

  Um die ersten 100 Ergebnisse aus dem Cache abzurufen, würden Sie eine Abfrage wie diese schreiben:

  ```
  g.with("Neptune#enableResultCache", true)
   .with("Neptune#numResultsCached", 500)
   .V().range(1000, 2000).range(0, 100)
  ```

  Diese hundert Ergebnisse würden den Ergebnissen der Basisabfrage im Bereich `(1000, 1100)` entsprechen.

Abfrage-Cache-Schlüssel für die Suche nach zwischengespeicherten Ergebnissen

Nach dem Zwischenspeichern der Ergebnisse einer Abfrage rufen nachfolgende Abfragen mit demselben Abfrage-Cache-Schlüssel Ergebnisse aus dem Cache ab, anstatt neue zu generieren. Der Abfrage-Cache-Schlüssel einer Abfrage wird wie folgt ausgewertet:

1. Alle Cache-bezogenen Abfragehinweise außer `numResultsCached` werden ignoriert.
2. Ein letzter `iterate()`-Schritt wird ignoriert.
3. Der Rest der Abfrage wird entsprechend ihrer Bytecode-Darstellung angeordnet.

Die resultierende Zeichenfolge wird mit dem Index der Abfrageergebnisse abgeglichen, die sich bereits im Cache befinden, um festzustellen, ob es für die Abfrage einen Cache-Treffer gibt.

Betrachten Sie beispielsweise diese Abfrage:

```
g.withSideEffect('Neptune#typePromotion', false).with("Neptune#enableResultCache",
 true)
 .with("Neptune#numResultsCached", 50)
 .V().has('genre','drama').in('likes').iterate()
```

Sie wird als Bytecode-Version hiervon gespeichert:

```
g.withSideEffect('Neptune#typePromotion', false)
 .with("Neptune#numResultsCached", 50)
 .V().has('genre','drama').in('likes')
```

Ausnahmen für den Ergebnis-Cache

Wenn die Ergebnisse einer Abfrage, die Sie zwischenspeichern möchten, zu groß für den Cache sind, auch wenn alle zuvor zwischengespeicherten Daten entfernt wurden, gibt Neptune den Fehler `QueryLimitExceededException` aus. Es werden keine Ergebnisse zurückgegeben und die Ausnahme generiert die folgende Fehlermeldung:

```
The result size is larger than the allocated cache,
      please refer to results cache best practices for options to rerun the query.
```

Sie können diese Meldung mittels des Abfragehinweises `noCacheExceptions` wie folgt unterdrücken:

```
g.with('Neptune#enableResultCache', true)
 .with('Neptune#noCacheExceptions', true)
 .V().has('genre','drama').in('likes')
```

Mit den Schritten Gremlin mergeV () und mergeE () effiziente Upserts erstellen

Ein Upsert (bzw. eine bedingte Einfügung) verwendet Eckpunkte oder Kanten, wenn bereits vorhanden, oder erstellt sie andernfalls. Effiziente Upserts können die Leistung von Gremlin-Abfragen deutlich verbessern.

Mit Upserts können Sie idempotente Einfügeoperationen schreiben: Das Ergebnis bleibt gleich, unabhängig davon, wie häufig eine Operation ausgeführt wird. Dies ist in hoch gleichzeitigen Schreibszenarien nützlich, in denen gleichzeitige Änderungen für denselben Teil eines Diagramms zumden Rollback einer oder mehrerer Transaktionen mit einer `ConcurrentModificationException` erzwingen können, was Wiederholungen erforderlich macht.

> **Note**
>
> Die `mergeE()` Schritte `mergeV()` und erfordern die Neptune-Engine-Version 1.2.1.0 oder höher, mit der 3.6-Unterstützung eingeführt wurde. TinkerPop Wenn auf Ihrem Cluster eine ältere Engine-Version ausgeführt wird, finden Sie stattdessen weitere Informationen. Effiziente Upserts vor 3.6.x Sie können Ihre aktuelle Engine-Version anhand des Instanzstatus-Endpunkts überprüfen.

Die folgende Abfrage führt beispielsweise einen Upsert für einen Eckpunkt mittels der bereitgestellten `Map` aus, um zunächst zu versuchen, einen Eckpunkt mit der `T.id "v-1"` zu finden. Wenn dieser Eckpunkt gefunden wird, wird er zurückgegeben. Wenn er nicht gefunden wird, wird durch die `onCreate`-Klausel ein Eckpunkt mit dieser `id` und der Eigenschaft erstellt.

```
g.mergeV([(id):'v-1']).
  option(onCreate, [(label): 'PERSON', 'email': 'person-1@example.org'])
```

Batching von Upserts zur Verbesserung des Durchsatzes

In Schreibszenarien mit einem hohen Durchsatz können Sie `mergeV()`- und `mergeE()`-Schritte verketten, um Upserts für Eckpunkte und Kanten batchweise auszuführen. Batching reduziert den Transaktionsaufwand, der entsteht, wenn für eine große Anzahl von Eckpunkten und Kanten Upserts ausgeführt werden. Sie können den Durchsatz weiter verbessern, indem Sie Batch-Anforderungen parallel über mehrere Clients ausführen.

Als Faustregel gilt, dass pro Batch-Anforderung Upserts für ungefähr 200 Datensätze ausgeführt werden sollten. Ein Datensatz ist eine einzelne Eckpunkt- oder Kantenbezeichnung oder -eigenschaft. Ein Eckpunkt mit einer einzelnen Bezeichnung und 4 Eigenschaften generiert beispielsweise 5 Datensätze. Eine Kante mit einer Bezeichnung und einer einzelnen Eigenschaft generiert 2 Datensätze. Wenn Sie Upserts für Batches von Eckpunkten ausführen möchten, die

jeweils eine einzelne Bezeichnung und 4 Eigenschaften besitzen, sollten Sie mit einer Batchgröße von 40 beginnen, da `200 / (1 + 4) = 40`.

Sie können mit der Batchgröße experimentieren. 200 Datensätze pro Batch sind ein guter Ausgangspunkt. Die ideale Batchgröße kann je nach Workload jedoch höher oder niedriger sein. Beachten Sie jedoch, dass Neptune die Gesamtzahl der Gremlin-Schritte pro Anforderung einschränken kann. Diese Einschränkung ist nicht dokumentiert. Um jedoch auf der sicheren Seite zu sein, sollten Ihre Anforderungen nicht mehr als 1 500 Gremlin-Schritte enthalten. Neptune lehnt große Batchanforderungen mit mehr als 1 500 Schritten möglicherweise ab.

Um den Durchsatz zu erhöhen, können Sie Upserts für Batches parallel über mehrere Clients ausführen (siehe Erstellen von effizienten Multi-Thread-Gremlin-Schreibvorgängen). Die Anzahl der Clients sollte der Anzahl der Worker-Threads in Ihrer Neptune-Writer-Instance entsprechen, in der Regel das 2-Fache der Anzahl der vCPUs auf dem Server. Eine `r5.8xlarge`-Instance hat beispielsweise 32 vCPUs und 64 Worker-Threads. Für Schreibszenarien mit einem hohen Durchsatz über eine `r5.8xlarge` würden Sie daher 64 Clients verwenden, die Batch-Upserts parallel in Neptune schreiben.

Jeder Client sollte eine Batch-Anforderung einreichen und mit der Einreichung einer weiteren Anforderung bis zum Abschluss der Anforderung warten. Obwohl diese mehreren Clients parallel ausgeführt werden, senden die einzelnen Clients die Anforderungen seriell. Dies stellt sicher, dass der Server kontinuierlich Anforderungen erhält, die alle Worker-Threads auslasten, ohne dass die serverseitige Anforderungswarteschlange überlastet wird (siehe Dimensionieren von DB-Instances in einem Neptune-DB-Cluster).

Vermeiden von Schritten, die mehrere Traverser generieren

Wenn ein Gremlin-Step ausgeführt wird, nimmt er einen eingehenden Traverser auf und gibt einen oder mehrere Ausgabe-Traverser aus. Die Anzahl der von einem Schritt ausgegebenen Traverser bestimmt, wie häufig der nächste Schritt ausgeführt wird.

In der Regel soll jede Batchoperation, z. B. ein Upsert von Eckpunkt A, nur einmal ausgeführt werden, sodass die Reihenfolge der Operationen wie folgt aussieht: Upsert von Eckpunkt A, dann Upsert von Eckpunkt B, dann Upsert von Eckpunkt C und so weiter. Solange ein Schritt nur ein Element erstellt oder ändert, gibt er nur einen Traverser aus, und die Schritte, die die nächste Operation darstellen, werden nur einmal ausgeführt. Wenn eine Operation hingegen mehr als ein Element erstellt oder ändert, gibt sie mehrere Traverser aus, sodass die nachfolgenden Schritte mehrfach ausgeführt werden, jeweils einmal pro ausgegebenem Traverser. Dies kann dazu führen,

dass die Datenbank unnötige zusätzliche Arbeit leistet. In einigen Fällen kann es zur Erstellung unerwünschter zusätzlicher Eckpunkte, Kanten oder Eigenschaftswerte kommen.

Ein Beispiel für Fehler dieser Art ist eine Abfrage wie `g.V().addV()`. Diese einfache Abfrage fügt für jeden im Diagramm gefundenen Eckpunkt einen Eckpunkt hinzu, da `V()` für jeden Eckpunkt im Diagramm einen Traverser ausgibt und jeder Traverser den Aufruf von `addV()` auslöst.

Informationen zur Behandlung von Operationen, die mehrere Traverser auslösen können, finden Sie unter Kombinieren von Upserts und Einfügungen.

Upserts für Eckpunkte

Der `mergeV()`-Schritt ist speziell für Upserts von Eckpunkten vorgesehen. Er verwendet `Map` als Argument. Dieses Argument repräsentiert Elemente, die mit vorhandenen Eckpunkten im Diagramm abgeglichen werden sollen. Wenn ein Element nicht gefunden wird, wird diese `Map` zur Erstellung eines neuen Eckpunkts verwendet. Mit diesem Schritt können Sie auch das Verhalten bei einer Erstellung oder Übereinstimmung ändern, da der Modulator `option()` zusammen mit den Token `Merge.onCreate` und `Merge.onMatch` zur Steuerung des Verhaltens verwendet werden kann. Weitere Informationen zur Verwendung dieses Schritts finden Sie in der TinkerPop Referenzdokumentation.

Sie können eine Eckpunkt-ID verwenden, um festzustellen, ob ein bestimmter Eckpunkt vorhanden ist. Dies ist der bevorzugte Ansatz, da Neptune Upserts für hoch gleichzeitige Anwendungsfälle für IDs optimiert. Die folgende Abfrage erstellt beispielsweise einen Eckpunkt mit einer bestimmten Eckpunkt-ID, wenn noch nicht vorhanden, oder verwendet ihn erneut, wenn vorhanden:

```
g.mergeV([(T.id): 'v-1']).
    option(onCreate, [(T.label): 'PERSON', email: 'person-1@example.org', age: 21]).
    option(onMatch, [age: 22]).
  id()
```

Beachten Sie, dass diese Abfrage mit dem `id()`-Schritt endet. Der `id()`-Schritt am Ende einer Upsert-Abfrage ist nicht unbedingt erforderlich, stellt jedoch sicher, dass der Server nicht alle Eckpunkteigenschaften zurück zum Client serialisiert, was den Sperraufwand der Abfrage reduziert.

Alternativ können Sie zur Identifizierung eines Eckpunkts auch eine Eckpunkteigenschaft verwenden:

```
g.mergeV([email: 'person-1@example.org']).
    option(onCreate, [(T.label): 'PERSON', age: 21]).
    option(onMatch, [age: 22]).
```

```
  id()
```

Verwenden Sie zur Erstellung von Eckpunkten nach Möglichkeit die von Ihnen bereitgestellten IDs und verwenden Sie während Upsert-Operationen diese IDs, um zu ermitteln, ob ein bestimmter Eckpunkt vorhanden ist. So kann Neptune die Upserts optimieren. Ein ID-based Upsert kann erheblich effizienter sein als ein eigenschaftsbasierter Upsert, wenn gleichzeitige Änderungen üblich sind.

Verketten von Eckpunkt-Upserts

Sie können Eckpunkt-Upserts verketten, um sie in ein Batch einzufügen:

```
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()
```

Alternativ können Sie auch diese `mergeV()`-Syntax verwenden:

```
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'])
```

Da dieses Abfrageformat jedoch Elemente in den Suchkriterien enthält, die für die einfache Suche nach `id` nicht benötigt werden, ist sie nicht so effizient wie die vorherige Abfrage.

Upserts für Kanten

Der `mergeE()`-Schritt ist speziell für Upserts von Kanten vorgesehen. Er verwendet `Map` als Argument. Dieses Argument repräsentiert Elemente, die mit vorhandenen Kanten im Diagramm

abgeglichen werden sollen. Wenn ein Element nicht gefunden wird, wird diese `Map` zur Erstellung einer neuen Kante verwendet. Mit diesem Schritt können Sie auch das Verhalten bei einer Erstellung oder Übereinstimmung ändern, da der Modulator `option()` zusammen mit den Token `Merge.onCreate` und `Merge.onMatch` zur Steuerung des Verhaltens verwendet werden kann. Weitere Informationen zur TinkerPop Verwendung dieses Schritts finden Sie in der Referenzdokumentation.

Sie können mittels Kanten-IDs Upserts für Kanten auf die gleiche Weise wie für Eckpunkte mittels benutzerdefinierter Eckpunkt-IDs ausführen. Dies ist auch hier der bevorzugte Ansatz, da Neptune so die Abfrage optimieren kann. Die folgende Abfrage erstellt beispielsweise eine Kante auf Grundlage der Kanten-ID, wenn noch nicht vorhanden, oder verwendet sie erneut, wenn vorhanden. Die Abfrage verwendet auch die IDs der Eckpunkte `Direction.from` und `Direction.to`, wenn eine neue Kante erstellt werden muss:

```
g.mergeE([(T.id): 'e-1']).
    option(onCreate, [(from): 'v-1', (to): 'v-2', weight: 1.0]).
    option(onMatch, [weight: 0.5]).
  id()
```

Beachten Sie, dass diese Abfrage mit dem `id()`-Schritt endet. Der `id()`-Schritt am Ende der Upsert-Abfrage für die Kante ist nicht unbedingt erforderlich, stellt jedoch sicher, dass der Server nicht alle Kanteneigenschaften zurück zum Client serialisiert, was den Sperraufwand der Abfrage reduziert.

Viele Anwendungen verwenden benutzerdefinierte Eckpunkt-IDs, überlassen Neptune jedoch die Generierung von Kanten-IDs. Wenn Sie die ID einer Kante nicht kennen, aber die IDs der Eckpunkte `from` und `to` kennen, können Sie diese Art von Abfrage für Kanten-Upserts verwenden:

```
g.mergeE([(from): 'v-1', (to): 'v-2', (T.label): 'KNOWS']).
  id()
```

Alle Eckpunkte, auf die `mergeE()` verweist, müssen vorhanden sein, damit der Schritt die Kante erstellen kann.

Verketten von Kanten-Upserts

Wie bei Eckpunkt-Upserts können auch hier `mergeE()`-Schritte für Batch-Anforderungen verkettet werden:

```
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()
```

Kombinieren von Eckpunkt- und Kanten-Upserts

Manchmal möchten Sie vielleicht Upserts für Eckpunkte und die Kanten ausführen, die sie verbinden. Sie können die hier gezeigten Batch-Beispiele kombinieren. Das folgende Beispiel zeigt ein Upsert für 3 Eckpunkte und 2 Kanten:

```
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()
```

Kombinieren von Upserts und Einfügungen

Manchmal möchten Sie vielleicht Upserts für Eckpunkte und die Kanten ausführen, die sie verbinden. Sie können die hier gezeigten Batch-Beispiele kombinieren. Das folgende Beispiel zeigt ein Upsert für 3 Eckpunkte und 2 Kanten:

In der Regel werden Upserts Element für Element ausgeführt. Wenn Sie die hier gezeigten Upsert-Muster befolgen, gibt jede Upsert-Operation einen einzelnen Traverser aus, sodass die nachfolgende Operation jeweils nur einmal ausgeführt wird.

Manchmal möchten Sie jedoch vielleicht Upserts und Einfügungen kombinieren. Dies kann beispielsweise der Fall sein, wenn Sie Kanten verwenden, um Instances von Aktionen oder Ereignissen darzustellen. Möglicherweise verwendet eine Anforderung Upserts, um sicherzustellen, dass alle notwendigen Eckpunkte vorhanden sind, und dann Einfügungen, um Kanten hinzuzufügen. Achten Sie bei Anforderungen dieser Art auf die potenzielle Anzahl von Traversern, die bei jeder Operation ausgegeben werden.

Im folgenden Beispiel werden Upserts und Einfügungen kombiniert, um dem Diagramm Kanten hinzuzufügen, die Ereignisse repräsentieren:

```
// 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()
```

Die Abfrage sollte 5 Kanten einfügen: 2 FOLGE-Kanten und 3 AUFGERUFENE Kanten. Die geschriebene Abfrage fügt jedoch 8 Kanten ein: 2 FOLGE-Kanten und 6 AUFGERUFENE Kanten. Der Grund hierfür ist, dass die Operation, die die 2 FOLGE-Kanten einfügt, 2 Traverser ausgibt. Daher wird die nachfolgende Einfügeoperation, die 3 Kanten einfügt, zweimal ausgeführt.

Die Lösung besteht darin, nach jeder Operation, die mehr als einen Traverser ausgeben könnte, einen `fold()`-Schritt hinzuzufügen:

```
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()
```

Hier wurde im Anschluss an die Operation, die FOLGE-Kanten einfügt, ein `fold()`-Schritt eingefügt. Das Ergebnis ist ein einzelner Traverser, sodass die nachfolgende Operation nur einmal ausgeführt wird.

Dieser Ansatz hat den Nachteil, dass die Abfrage jetzt nicht vollständig optimiert ist, da `fold()` nicht optimiert ist. Die Einfügeoperation nach `fold()` wird jetzt ebenfalls nicht optimiert.

Wenn Sie die `fold()` verwenden müssen, um die Zahl der Traverser für nachfolgende Schritte zu reduzieren, ordnen Sie die Operationen so an, dass die kostengünstigsten Operationen den nicht optimierten Teil der Abfrage einnehmen.

Kardinalität einstellen

Die Standardkardinalität für Scheitelpunkteigenschaften in Neptune ist festgelegt, was bedeutet, dass bei Verwendung von mergeV () allen in der Map angegebenen Werten diese Kardinalität zugewiesen wird. Um die einfache Kardinalität zu verwenden, müssen Sie ihre Verwendung explizit angeben. Ab TinkerPop Version 3.7.0 gibt es eine neue Syntax, die es ermöglicht, die Kardinalität als Teil der Map anzugeben, wie im folgenden Beispiel gezeigt:

```
g.mergeV([(T.id): '1234']).
  option(onMatch, ['age': single(20), 'name': single('alice'), 'city': set('miami')])
```

Alternativ können Sie die Kardinalität dafür wie folgt als Standard festlegen: `option`

```
// 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)
```

`mergeV()`Vor Version 3.7.0 gibt es weniger Optionen zum Einstellen der Kardinalität. Der allgemeine Ansatz besteht darin, auf den folgenden `property()` Schritt zurückzugreifen:

```
g.mergeV([(T.id): '1234']).
  option(onMatch, sideEffect(property(single,'age', 20).
  property(set,'city','miami')).constant([:]))
```

> **Note**
>
> Dieser Ansatz funktioniert nur, `mergeV()` wenn er mit einem Startschritt verwendet wird. Sie könnten daher keine Kette `mergeV()` innerhalb einer einzigen Traversierung erstellen, da der erste Schritt `mergeV()` nach dem Startschritt, der diese Syntax verwendet, einen Fehler erzeugt, falls es sich bei dem eingehenden Traverser um ein Graphelement handelt. In diesem Fall sollten Sie Ihre `mergeV()` Aufrufe in mehrere Anfragen aufteilen, von denen jede ein Startschritt sein kann.

Effiziente Gremlin-Upserts mit `fold () /coalesce () /unfold` () erstellen

Ein Upsert (bzw. eine bedingte Einfügung) verwendet Eckpunkte oder Kanten, wenn bereits vorhanden, oder erstellt sie andernfalls. Effiziente Upserts können die Leistung von Gremlin-Abfragen deutlich verbessern.

Diese Seite zeigt die Verwendung des Gremlin-Musters `fold()/coalesce()/unfold()` zur Erstellung effizienter Upserts. Mit der Veröffentlichung der TinkerPop Version 3.6.x, die in Neptune in der Engine-Version 1.2.1.0 eingeführt wurde, sind die neuen `mergeE()` Schritte `mergeV()` und in den meisten Fällen jedoch vorzuziehen. Das hier beschriebene Muster `fold()/coalesce()/unfold()` ist in einigen komplexen Situationen möglicherweise weiter nützlich. Grundsätzlich sollten Sie jedoch `mergeV()` und `mergeE()` verwenden, wie in Mit den Schritten Gremlin mergeV () und mergeE () effiziente Upserts erstellen beschrieben.

Mit Upserts können Sie idempotente Einfügeoperationen schreiben: Das Ergebnis bleibt gleich, unabhängig davon, wie häufig eine Operation ausgeführt wird. Dies ist in hoch gleichzeitigen Schreibszenarien nützlich, in denen gleichzeitige Änderungen für denselben Teil eines Diagramms zumden Rollback einer oder mehrerer Transaktionen mit einer `ConcurrentModificationException` erzwingen können, was eine Wiederholung erforderlich macht.

Die folgende Abfrage führt beispielsweise ein Eckpunkt-Upsert aus, indem sie zunächst im Datensatz nach dem angegebenen Eckpunkt sucht und dann die Ergebnisse zu einer Liste zusammenfasst. Bei der ersten Traversierung für den `coalesce()`-Schritt wird diese Liste dann von der Abfrage geöffnet. Wenn die geöffnete Liste nicht leer ist, werden die Ergebnisse von `coalesce()` ausgegeben. Wenn `unfold()` jedoch eine leere Sammlung zurückgibt, da der Eckpunkt zurzeit nicht vorhanden ist, evaluiert `coalesce()` die zweite bereitgestellte Traversierung. Bei dieser zweiten Traversierung erstellt die Abfrage den fehlenden Eckpunkt.

```
g.V('v-1').fold()
          .coalesce(
             unfold(),
             addV('Person').property(id, 'v-1')
                           .property('email', 'person-1@example.org')
           )
```

Verwenden Sie eine optimierte Form von coalesce () für Upserts

Neptune kann den Ausdruck `fold().coalesce(unfold(), ...)` optimieren, um Aktualisierungen mit einem hohen Durchsatz zu erstellen. Dies funktioniert jedoch nur, wenn beide Teile von `coalesce()` entweder einen Eckpunkt oder eine Kante zurückgeben, aber sonst nichts. Wenn Sie versuchen, aus einem Teil von `coalesce()` etwas anderes zurückzugeben, z. B. eine Eigenschaft, erfolgt die Neptune-Optimierung nicht. Die Abfrage kann zwar erfolgreich sein, funktioniert jedoch nicht so gut wie eine optimierte Version, insbesondere bei großen Datensätzen.

Da nicht optimierte Upsert-Abfragen die Ausführungszeiten verlängern und den Durchsatz reduzieren, lohnt sich die Verwendung des Gremlin-Endpunkts `explain`, um festzustellen, ob eine Upsert-Abfrage vollständig optimiert ist. Achten Sie bei der Überprüfung von `explain`-Plänen auf Zeilen, die mit `+ not converted into Neptune steps` und `WARNING: >>` beginnen. Beispiel:

```
+ not converted into Neptune steps: [FoldStep, CoalesceStep([[UnfoldStep],
 [AddEdgeSte...
WARNING: >> FoldStep << is not supported natively yet
```

Diese Warnungen können Ihnen helfen, die Teile einer Abfrage zu identifizieren, die ihre vollständige Optimierung verhindern.

Manchmal ist es nicht möglich, eine Abfrage vollständig zu optimieren. In diesen Situationen sollten Sie versuchen, die Schritte, die nicht optimiert werden können, an das Ende der Abfrage zu setzen, sodass die Engine so viele Schritte wie möglich optimieren kann. Diese Technik wird in einigen der Beispiele für Batch-Upserts verwendet, bei denen alle optimierten Upserts für einen Satz von Eckpunkten oder Kanten ausgeführt werden, bevor auf dieselben Eckpunkte oder Kanten zusätzliche, möglicherweise nicht optimierte Änderungen angewendet werden.

Batching von Upserts zur Verbesserung des Durchsatzes

In Schreibszenarien mit einem hohen Durchsatz können Sie Upsert-Schritte verketten, um Upserts für Eckpunkte und Kanten batchweise auszuführen. Batching reduziert den Transaktionsaufwand, der entsteht, wenn für eine große Anzahl von Eckpunkten und Kanten Upserts ausgeführt werden. Sie können den Durchsatz weiter verbessern, indem Sie Batch-Anforderungen parallel über mehrere Clients ausführen.

Als Faustregel gilt, dass pro Batch-Anforderung Upserts für ungefähr 200 Datensätze ausgeführt werden sollten. Ein Datensatz ist eine einzelne Eckpunkt- oder Kantenbezeichnung oder -

eigenschaft. Ein Eckpunkt mit einer einzelnen Bezeichnung und 4 Eigenschaften generiert beispielsweise 5 Datensätze. Eine Kante mit einer Bezeichnung und einer einzelnen Eigenschaft generiert 2 Datensätze. Wenn Sie Upserts für Batches von Eckpunkten ausführen möchten, die jeweils eine einzelne Bezeichnung und 4 Eigenschaften besitzen, sollten Sie mit einer Batchgröße von 40 beginnen, da `200 / (1 + 4) = 40`.

Sie können mit der Batchgröße experimentieren. 200 Datensätze pro Batch sind ein guter Ausgangspunkt. Die ideale Batchgröße kann je nach Workload jedoch höher oder niedriger sein. Beachten Sie jedoch, dass Neptune die Gesamtzahl der Gremlin-Schritte pro Anforderung einschränken kann. Diese Einschränkung ist nicht dokumentiert. Um jedoch auf der sicheren Seite zu sein, sollten Ihre Anforderungen nicht mehr als 1 500 Gremlin-Schritte enthalten. Neptune lehnt große Batchanforderungen mit mehr als 1 500 Schritten möglicherweise ab.

Um den Durchsatz zu erhöhen, können Sie Upserts für Batches parallel über mehrere Clients ausführen (siehe Erstellen von effizienten Multi-Thread-Gremlin-Schreibvorgängen). Die Anzahl der Clients sollte der Anzahl der Worker-Threads in Ihrer Neptune-Writer-Instance entsprechen, in der Regel das 2-Fache der Anzahl der vCPUs auf dem Server. Eine `r5.8xlarge`-Instance hat beispielsweise 32 vCPUs und 64 Worker-Threads. Für Schreibszenarien mit einem hohen Durchsatz über eine `r5.8xlarge` würden Sie daher 64 Clients verwenden, die Batch-Upserts parallel in Neptune schreiben.

Jeder Client sollte eine Batch-Anforderung einreichen und mit der Einreichung einer weiteren Anforderung bis zum Abschluss der Anforderung warten. Obwohl diese mehreren Clients parallel ausgeführt werden, senden die einzelnen Clients die Anforderungen seriell. Dies stellt sicher, dass der Server kontinuierlich Anforderungen erhält, die alle Worker-Threads auslasten, ohne dass die serverseitige Anforderungswarteschlange überlastet wird (siehe Dimensionieren von DB-Instances in einem Neptune-DB-Cluster).

Vermeiden von Schritten, die mehrere Traverser generieren

Wenn ein Gremlin-Step ausgeführt wird, nimmt er einen eingehenden Traverser auf und gibt einen oder mehrere Ausgabe-Traverser aus. Die Anzahl der von einem Schritt ausgegebenen Traverser bestimmt, wie häufig der nächste Schritt ausgeführt wird.

In der Regel soll jede Batchoperation, z. B. ein Upsert von Eckpunkt A, nur einmal ausgeführt werden, sodass die Reihenfolge der Operationen wie folgt aussieht: Upsert von Eckpunkt A, dann Upsert von Eckpunkt B, dann Upsert von Eckpunkt C und so weiter. Solange ein Schritt nur ein Element erstellt oder ändert, gibt er nur einen Traverser aus, und die Schritte, die die nächste Operation darstellen, werden nur einmal ausgeführt. Wenn eine Operation hingegen mehr als ein

Element erstellt oder ändert, gibt sie mehrere Traverser aus, sodass die nachfolgenden Schritte mehrfach ausgeführt werden, jeweils einmal pro ausgegebenem Traverser. Dies kann dazu führen, dass die Datenbank unnötige zusätzliche Arbeit leistet. In einigen Fällen kann es zur Erstellung unerwünschter zusätzlicher Eckpunkte, Kanten oder Eigenschaftswerte kommen.

Ein Beispiel für Fehler dieser Art ist eine Abfrage wie `g.V().addV()`. Diese einfache Abfrage fügt für jeden im Diagramm gefundenen Eckpunkt einen Eckpunkt hinzu, da `V()` für jeden Eckpunkt im Diagramm einen Traverser ausgibt und jeder Traverser den Aufruf von `addV()` auslöst.

Informationen zur Behandlung von Operationen, die mehrere Traverser auslösen können, finden Sie unter Kombinieren von Upserts und Einfügungen.

Upserts für Eckpunkte

Sie können mittels einer Eckpunkt-ID feststellen, ob ein entsprechender Eckpunkt vorhanden ist. Dies ist der bevorzugte Ansatz, da Neptune Upserts für hoch gleichzeitige Anwendungsfälle für IDs optimiert. Die folgende Abfrage erstellt beispielsweise einen Eckpunkt mit einer bestimmten Eckpunkt-ID, wenn noch nicht vorhanden, oder verwendet ihn erneut, wenn vorhanden:

```
g.V('v-1')
 .fold()
  .coalesce(unfold(),
            addV('Person').property(id, 'v-1')
                          .property('email', 'person-1@example.org'))
  .id()
```

Beachten Sie, dass diese Abfrage mit dem `id()`-Schritt endet. Der `id()`-Schritt am Ende einer Upsert-Abfrage ist nicht unbedingt erforderlich, stellt jedoch sicher, dass der Server nicht alle Eckpunkteigenschaften zurück zum Client serialisiert, was den Sperraufwand der Abfrage reduziert.

Alternativ können Sie mittels einer Eckpunkteigenschaft ermitteln, ob der Eckpunkt vorhanden ist:

```
g.V()
 .hasLabel('Person')
 .has('email', 'person-1@example.org')
 .fold()
 .coalesce(unfold(),
           addV('Person').property('email', 'person-1@example.org'))
 .id()
```

Verwenden Sie zur Erstellung von Eckpunkten nach Möglichkeit die von Ihnen bereitgestellten IDs und verwenden Sie während Upsert-Operationen diese IDs, um zu ermitteln, ob ein bestimmter Eckpunkt vorhanden ist. So kann Neptune Upserts für die IDs optimieren. In stark ID-based parallelen Änderungsszenarien kann ein Upsert deutlich effizienter sein als ein eigenschaftsbasierter Upsert.

Verketten von Eckpunkt-Upserts

Sie können Eckpunkt-Upserts verketten, um sie in ein Batch einzufügen:

```
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()
```

Upserts für Kanten

Sie können mittels Kanten-IDs Upserts für Kanten auf die gleiche Weise wie für Eckpunkte mittels benutzerdefinierter Eckpunkt-IDs ausführen. Dies ist auch hier der bevorzugte Ansatz, da Neptune so die Abfrage optimieren kann. Die folgende Abfrage erstellt beispielsweise eine Kante auf Grundlage der Kanten-ID, wenn noch nicht vorhanden, oder verwendet sie erneut, wenn vorhanden. Die Abfrage verwendet auch die IDs der Eckpunkte `from` und `to`, wenn eine neue Kante erstellt werden muss.

```
g.E('e-1')
 .fold()
 .coalesce(unfold(),
           addE('KNOWS').from(V('v-1'))
                        .to(V('v-2'))
                        .property(id, 'e-1'))
 .id()
```

Viele Anwendungen verwenden benutzerdefinierte Eckpunkt-IDs, überlassen Neptune jedoch die Generierung von Kanten-IDs. Wenn Sie die ID einer Kante nicht kennen, aber die IDs der Eckpunkte `from` und `to` kennen, können Sie diese Art von Formulierung für Kanten-Upserts verwenden:

```
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()
```

Dr Eckpunktschritt in der `where()`-Klausel sollte `inV()` sein (oder `outV()`, wenn Sie `inE()` für die Suche nach der Kante verwendet haben), nicht `otherV()`. Verwenden Sie hier nicht `otherV()`, da die Abfrage dann nicht optimiert wird und die Leistung beeinträchtigt wird. Neptune würde beispielsweise die folgende Abfrage nicht optimieren:

```
// 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()
```

Wenn Sie die Kanten- oder Eckpunkt-IDs nicht bereits kennen, können Sie Upserts mittels Eckpunkteigenschaften ausführen:

```
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()
```

Wie bei Scheitelpunkt-Upserts ist es vorzuziehen, Kanten-Upserts mit entweder einer ID-based Kanten-ID oder `from` und `to` Scheitelpunkt-IDs zu verwenden, anstatt eigenschaftsbasierte Hochstellungen zu verwenden, damit Neptune den Upsert vollständig optimieren kann.

Es wird geprüft, ob ein Scheitelpunkt von und zu einem Scheitelpunkt existiert

Beachten Sie die Konstruktion der Schritte zur Erstellung einer neuen Kante: `addE().from().to()`. Diese Konstruktion stellt sicher, dass die Abfrage das Vorhandensein sowohl des `from`- als auch des `to`-Eckpunkts prüft. Wenn keiner dieser Eckpunkte vorhanden ist, gibt die Abfrage einen Fehler wie folgt zurück:

```
{
  "detailedMessage": "Encountered a traverser that does not map to a value for child...
  "code": "IllegalArgumentException",
  "requestId": "..."
}
```

Wenn der `from`- oder der `to`-Eckpunkt möglicherweise nicht vorhanden ist, sollten Sie ein Upsert diesen Eckpunkt versuchen, bevor Sie ein Upsert für die Kante zwischen den beiden Eckpunkten ausführen. Siehe Kombinieren von Eckpunkt- und Kanten-Upserts.

Es gibt eine alternative Konstruktion zum Erstellen einer Kante, die Sie nicht verwenden sollten: `V().addE().to()`. Sie fügt nur dann eine Kante hinzu, wenn der `from`-Eckpunkt vorhanden ist. Wenn der `to`-Eckpunkt nicht vorhanden ist, generiert die Abfrage wie zuvor beschrieben einen Fehler. Wenn jedoch der `from`-Eckpunkt nicht vorhanden ist, schlägt das Einfügen einer Kante ohne Meldung fehl, ohne dass ein Fehler generiert wird. Die folgende Upsert-Operation wird beispielsweise ohne Kanten-Upsert abgeschlossen, wenn der `from`-Eckpunkt nicht vorhanden ist:

```
// 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()
```

Verketten von Kanten-Upserts

Wenn Sie Kanten-Upserts verketten möchten, um eine Batch-Anforderung zu erstellen, müssen Sie jeden Upsert mit einer Eckpunktsuche beginnen, auch wenn Sie die Kanten-IDs bereits kennen.

Wenn Sie die IDs der Kanten, für die Sie ein Upsert ausführen möchten, und die IDs der `from`- und `to`-Eckpunkte bereits kennen, können Sie diese Formulierung verwenden:

```
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()
```

Das vielleicht häufigste Szenario besteht darin, dass Sie die IDs der `from`- und `to`-Eckpunkte kennen, aber nicht die IDs der Kanten, für die Sie ein Upsert ausführen möchten. Verwenden Sie in diesem Fall die folgende Formulierung:

```
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()
```

Wenn Sie die IDs der Kanten kennen, für die Sie ein Upsert ausführen möchten, aber nicht die IDs der `from`- und `to`-Eckpunkte (was ungewöhnlich ist), können Sie diese Formulierung verwenden:

```
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()
```

Kombinieren von Eckpunkt- und Kanten-Upserts

Manchmal möchten Sie vielleicht Upserts für Eckpunkte und die Kanten ausführen, die sie verbinden. Sie können die hier gezeigten Batch-Beispiele kombinieren. Das folgende Beispiel zeigt ein Upsert für 3 Eckpunkte und 2 Kanten:

```
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()
```

Kombinieren von Upserts und Einfügungen

Manchmal möchten Sie vielleicht Upserts für Eckpunkte und die Kanten ausführen, die sie verbinden. Sie können die hier gezeigten Batch-Beispiele kombinieren. Das folgende Beispiel zeigt ein Upsert für 3 Eckpunkte und 2 Kanten:

In der Regel werden Upserts Element für Element ausgeführt. Wenn Sie die hier gezeigten Upsert-Muster befolgen, gibt jede Upsert-Operation einen einzelnen Traverser aus, sodass die nachfolgende Operation jeweils nur einmal ausgeführt wird.

Manchmal möchten Sie jedoch vielleicht Upserts und Einfügungen kombinieren. Dies kann beispielsweise der Fall sein, wenn Sie Kanten verwenden, um Instances von Aktionen oder Ereignissen darzustellen. Möglicherweise verwendet eine Anforderung Upserts, um sicherzustellen, dass alle notwendigen Eckpunkte vorhanden sind, und dann Einfügungen, um Kanten hinzuzufügen. Achten Sie bei Anforderungen dieser Art auf die potenzielle Anzahl von Traversern, die bei jeder Operation ausgegeben werden.

Im folgenden Beispiel werden Upserts und Einfügungen kombiniert, um dem Diagramm Kanten hinzuzufügen, die Ereignisse repräsentieren:

```
// 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()
```

Die Abfrage sollte 5 Kanten einfügen: 2 FOLGE-Kanten und 3 AUFGERUFENE Kanten. Die geschriebene Abfrage fügt jedoch 8 Kanten ein: 2 FOLGE-Kanten und 6 AUFGERUFENE Kanten. Der Grund hierfür ist, dass die Operation, die die 2 FOLGE-Kanten einfügt, 2 Traverser ausgibt. Daher wird die nachfolgende Einfügeoperation, die 3 Kanten einfügt, zweimal ausgeführt.

Die Lösung besteht darin, nach jeder Operation, die mehr als einen Traverser ausgeben könnte, einen `fold()`-Schritt hinzuzufügen:

```
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()
```

Hier wurde im Anschluss an die Operation, die FOLGE-Kanten einfügt, ein `fold()`-Schritt eingefügt. Das Ergebnis ist ein einzelner Traverser, sodass die nachfolgende Operation nur einmal ausgeführt wird.

Dieser Ansatz hat den Nachteil, dass die Abfrage jetzt nicht vollständig optimiert ist, da `fold()` nicht optimiert ist. Der folgende Einfügevorgang, der auf `fold()` folgt, wird jetzt nicht optimiert.

Wenn Sie die `fold()` verwenden müssen, um die Zahl der Traverser für nachfolgende Schritte zu reduzieren, ordnen Sie die Operationen so an, dass die kostengünstigsten Operationen den nicht optimierten Teil der Abfrage einnehmen.

Upserts mit Änderung vorhandener Eckpunkte und Kanten

Manchmal möchten Sie vielleicht noch nicht vorhandene Eckpunkte oder Kanten erstellen und dann eine Eigenschaft hinzufügen oder aktualisieren, unabhängig davon, ob es sich um neue oder vorhandene Eckpunkte oder Kanten handelt.

Um eine Eigenschaft hinzuzufügen oder zu ändern, verwenden Sie den Schritt `property()`. Verwenden Sie diesen Schritt außerhalb des Schritts `coalesce()`. Wenn Sie versuchen, eine Eigenschaft vorhandener Eckpunkte oder Kanten innerhalb des Schritts `coalesce()` zu ändern, wird die Abfrage möglicherweise nicht von der Neptune-Abfrage-Engine optimiert.

Die folgende Abfrage fügt jedem Upsert-Eckpunkt eine Zählereigenschaft hinzu oder aktualisiert diese. Jeder `property()`-Schritt besitzt eine einfache Kardinalität, um sicherzustellen, dass die

neuen Werte alle vorhandenen Werte ersetzen und nicht zu einer Gruppe vorhandener Werte hinzugefügt werden.

```
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()
```

Wenn es einen Eigenschaftswert gibt, z. B. den Zeitstempelwert `lastUpdated`, der für alle Upsert-Elemente gilt, können Sie ihn am Ende der Abfrage hinzufügen oder aktualisieren:

```
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()
```

Wenn es zusätzliche Bedingungen gibt, die bestimmen, ob ein Eckpunkt oder eine Kante weiter geändert werden sollte oder nicht, können Sie mit einem `has()`-Schritt die Elemente filtern, die geändert werden sollen. Das folgende Beispiel verwendet einen `has()`-Schritt, um Upsert-Eckpunkte basierend auf dem Wert ihrer `version`-Eigenschaft zu filtern. Die Abfrage aktualisiert dann für jeden Eckpunkt, dessen Wert für `version` kleiner als 3 ist, auf `version`:

```
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()
```

Analysieren der Ausführung Neptune Neptun-Abfragen mit Gremlin Explain

Amazon Neptune hat das Gremlin-Feature explain hinzugefügt. Dieses Feature ist ein Self-Service-Tool, um den Ausführungsansatz der Neptune-Engine zu verstehen. Sie rufen diese Funktion auf, indem Sie einem HTTP-Aufruf, der eine Gremlin-Abfrage sendet, einen `explain`-Parameter hinzufügen.

Die `explain`-Funktion bietet Informationen zur logischen Struktur von Abfrageausführungsplänen. So können Sie potenzielle Engpässe bei Bewertung und Ausführung identifizieren und Ihre Abfrage optimieren, wie in Optimieren von Gremline-Abfragen beschrieben. Anschließend können Sie mit Abfragehinweisen die Abfrageausführungspläne verbessern.

Themen

- Grundlagen der Funktionsweise von Gremlin-Abfragen in Neptune
- Verwendung der Gremlin Explain API in Neptune
- Gremlin-Profil-API in Neptune
- Optimieren von Gremlin-Abfragen mithilfe von Explain und Profile
- Native Unterstützung für Gremlin-Schritte in Amazon Neptune

Grundlagen der Funktionsweise von Gremlin-Abfragen in Neptune

Um die Gremlin-Berichte `explain` und `profile` in Amazon Neptune optimal nutzen zu können, sollten Sie einige Hintergrundinformationen zu Gremlin-Abfragen verstehen.

Themen

- Gremlin-Anweisungen in Neptune
- Verarbeitung von Gremlin-Abfragen mit Anweisungsindizes in Neptune
- Verarbeitung von Gremlin-Abfragen in Neptune

Gremlin-Anweisungen in Neptune

Die Eigenschaftsdiagrammdaten in Amazon Neptune bestehen aus Vier-Positions-Anweisungen (Quads). Jede dieser Anweisungen stellt eine einzelne atomare Einheit von Eigenschaftsdiagrammdaten dar. Weitere Informationen finden Sie unter Neptune-Diagrammdatenmodell. Ähnlich wie beim Resource Description Framework (RDF)-Datenmodell sind diese vier Positionen wie folgt:

- `subject (S)`
- `predicate (P)`
- `object (O)`
- `graph (G)`

Jede Anweisung ist eine Assertion über eine oder mehrere Ressourcen. Eine Anweisung kann beispielsweise das Vorhandensein einer Beziehung zwischen zwei Ressourcen bestätigen oder einer Ressource eine Eigenschaft (Schlüssel-Wert-Paar) anfügen.

Sie können sich das Prädikat als das Verb der Anweisung vorstellen, das den Typ der Beziehung oder Eigenschaft beschreibt. Das Objekt ist das Ziel der Beziehung oder der Wert der Eigenschaft. Die Diagrammposition ist optional und kann auf viele verschiedene Arten verwendet werden. Für Neptune-Eigenschaftsdiagrammdaten (PG-Daten) wird sie entweder nicht verwendet (Null-Diagramm) oder verwendet, um die ID für eine Kante darzustellen. Ein Satz von Anweisungen mit geteilten Ressourcen-IDs erstellt ein Diagramm.

Es gibt drei Klassen von Anweisungen im Neptune-Eigenschaftsdiagramm-Datenmodell:

Themen

- Gremlin-Eckpunktbezeichnungs-Anweisungen
- Gremlin Edge-Anweisungen
- Gremlin-Eigenschaftenanweisungen

Gremlin-Eckpunktbezeichnungs-Anweisungen

Eckpunktbezeichnungs-Anweisungen in Neptune dienen zwei Zwecken:

- Sie verfolgen die Bezeichnungen für einen Eckpunkt.
- Das Vorhandensein von mindestens einer dieser Anweisungen impliziert das Vorhandensein eines bestimmten Eckpunkts im Diagramm.

Das Subjekt dieser Anweisungen ist eine Eckpunkt-ID, und das Objekt ist eine Bezeichnung, die beide vom Benutzer angegeben werden. Sie verwenden ein spezielles festes Prädikat für diese Anweisungen, die als `<~label>` angezeigt werden, und eine Standard-Diagramm-ID (das Null-Diagramm), angezeigt als <~>.

Betrachten Sie beispielsweise die folgende `addV`-Transversale:

```
g.addV("Person").property(id, "v1")
```

Diese Transversale führt dazu, dass die folgende Anweisung zu dem Diagramm hinzugefügt wird.

```
StatementEvent[Added(<v1> <~label> <Person> <~>) .]
```

Gremlin Edge-Anweisungen

Eine Gremlin-Kanten-Anweisung impliziert das Vorhandensein einer Kante zwischen zwei Eckpunkten in einem Diagramm in Neptune. Das Subjekt (S) einer Edge-Anweisung ist der Quell-`from`-Eckpunkt. Das Prädikat (P) ist eine vom Benutzer bereitgestellte Grenze-Bezeichnung. Das Objekt (O) ist der `to`-Ziel-Eckpunkt. Das Diagramm (G) ist eine vom Benutzer bereitgestellte Edge-ID.

Betrachten Sie beispielsweise die folgende `addE`-Transversale:

```
g.addE("knows").from(V("v1")).to(V("v2")).property(id, "e1")
```

Die Transversale führt dazu, dass die folgende Anweisung zu dem Diagramm hinzugefügt wird.

```
StatementEvent[Added(<v1> <knows> <v2> <e1>) .]
```

Gremlin-Eigenschaftenanweisungen

Eine Gremlin-Eigenschaftsanweisung in Neptune bestätigt einen einzelnen Eigenschaftswert für einen Eckpunkt oder eine Kante. Das Subjekt ist eine vom Benutzer bereitgestellte Eckpunkt- oder Grenze-ID. Das Prädikat ist der Eigenschaftsname (Schlüssel), und das Objekt ist der einzelne Eigenschaftswert. Das Diagramm (G) ist wiederum die Standard-Diagramm-ID, das Null-Diagramm, das als angezeigt wird <~>.

Betrachten Sie das folgende Beispiel für eine Vertex-Eigenschaft.

```
g.V("v1").property("name", "John")
```

Diese Anweisung führt zu Folgendem.

```
StatementEvent[Added(<v1> <name> "John" <~>) .]
```

Eigenschaftenanweisungen unterscheiden sich dadurch, dass es sich bei ihrem Objekt um einen primitiven Wert handelt (`string`, `date`, `byte`, `short`, `int`, `long`, `float` oder `double`). Ihr Objekt ist keine Ressourcen-ID, die als Betreff einer anderen Assertion verwendet werden könnte.

Für Multi-Eigenschaften erhält jeder einzelne Eigenschaftswert in der Gruppe eine eigene Anweisung.

```
g.V("v1").property(set, "phone", "956-424-2563").property(set, "phone", "956-354-3692
 (tel:9563543692)")
```

Daraus resultiert Folgendes.

```
StatementEvent[Added(<v1> <phone> "956-424-2563" <~>) .]
StatementEvent[Added(<v1> <phone> "956-354-3692" <~>) .]
```

Kanteneigenschaften werden ähnlich wie Scheitelpunkteigenschaften behandelt, allerdings wird die Kantenkennung in der Position (S) verwendet. Beispiel: Hinzufügen einer Eigenschaft zu einer Kante:

```
g.E("e1").property("weight", 0.8)
```

Dies führt dazu, dass dem Diagramm die folgende Anweisung hinzugefügt wird.

```
StatementEvent[Added(<e1> <weight> 0.8 <~>) .]
```

Verarbeitung von Gremlin-Abfragen mit Anweisungsindizes in Neptune

Der Zugriff auf Anweisungen erfolgt in Amazon Neptune über drei Anweisungsindizes, wie in Indizierung von Anweisungen in Neptune beschrieben. Neptune extrahiert ein Anweisungsmuster aus einer Gremlin-Abfrage, in der einige Positionen bekannt sind. Die restlichen Positionen werden durch eine Indexsuche entdeckt.

Neptune geht davon aus, dass das Eigenschaftsdiagrammschema nicht groß ist. Das bedeutet, dass die Anzahl unterschiedlicher Kantenbezeichnungen und Eigenschaftsnamen vergleichsweise gering ist, was zu einer insgesamt geringen Anzahl unterschiedlicher Prädikate führt. Neptune verfolgt unterschiedliche Prädikate in einem separaten Index. Es verwendet diesen Cache von Prädikaten, um einen Union-Scan von `{ all P x POGS }` durchzuführen, anstatt einen OSGP-Index zu verwenden. Wenn Sie keinen OSGP-Index für die umgekehrte Transversale benötigen, sparen Sie sowohl Speicherplatz als auch Lastdurchsatz.

Mit der Neptune Gremlin Explain/Profile API können Sie die Anzahl der Prädikate in Ihrem Diagramm abrufen. Sie können anschließend feststellen, ob Ihre Anwendung die Neptune-Annahme aufhebt, dass Ihr Eigenschaftsdiagrammschema klein ist.

Die folgenden Beispiele zeigen, wie Neptune Indizes verwendet, um Gremlin-Abfragen zu verarbeiten.

Frage: Was sind die Bezeichnungen des Eckpunkts **v1**?

```
  Gremlin code:      g.V('v1').label()
```

```
Pattern:           (<v1>, <~label>, ?, ?)
Known positions:   SP
Lookup positions:  OG
Index:             SPOG
Key range:         <v1>:<~label>:*
```

Frage: Was sind die „weiß“-Out-Grenzen des Eckpunkts **v1**?

```
Gremlin code:      g.V('v1').out('knows')
Pattern:           (<v1>, <knows>, ?, ?)
Known positions:   SP
Lookup positions:  OG
Index:             SPOG
Key range:         <v1>:<knows>:*
```

Frage: Welche Eckpunkte haben eine **Person**-Eckpunktbezeichnung?

```
Gremlin code:      g.V().hasLabel('Person')
Pattern:           (?, <~label>, <Person>, <~>)
Known positions:   POG
Lookup positions:  S
Index:             POGS
Key range:         <~label>:<Person>:<~>:*
```

Frage: Was sind die from/to Eckpunkte einer bestimmten Kante? **e1**

```
Gremlin code:      g.E('e1').bothV()
Pattern:           (?, ?, ?, <e1>)
Known positions:   G
Lookup positions:  SPO
Index:             GPSO
Key range:         <e1>:*
```

Ein Anweisungsindex, den Neptune nicht besitzt, ist ein umgekehrter Traversierungs-OSGP-Index. Dieser Index könnte verwendet werden, um alle eingehenden Kanten für alle Kantenbezeichnungen zu sammeln, wie im folgenden Beispiel gezeigt.

Frage: Was sind die eingehenden benachbarten Eckpunkte **v1**?

```
Gremlin code:      g.V('v1').in()
Pattern:           (?, ?, <v1>, ?)
```

```
Known positions:   O
Lookup positions:  SPG
Index:             OSGP  // <-- Index does not exist
```

Verarbeitung von Gremlin-Abfragen in Neptune

In Amazon Neptune können komplexere Traversierungen durch Muster dargestellt werden, die basierend auf der Definition benannter Variablen eine Beziehung erstellen, die mit Mustern geteilt werden kann, um Joins zu erstellen. Dies wird im folgenden Beispiel veranschaulicht.

Frage: Was ist die Zwei-Hop-Nachbarschaft des Eckpunkts **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
```

Wenn Sie die ?2-Variable über die beiden Muster hinweg weitergeben (an der O-Position im ersten Muster und der S-Position des zweiten Musters), erstellen Sie einen Join zwischen den First-Hop-Nachbarn und den Second-Hop-Nachbarn. Jede Neptune-Lösung hat Bindungen für die drei genannten Variablen, die verwendet werden können, um einen TinkerPopTraverser (einschließlich Pfadinformationen) neu zu erstellen.

Der erste Schritt bei der Verarbeitung von Gremlin-Abfragen besteht darin, die Abfrage in ein TinkerPop Traversal-Objekt zu zerlegen, das aus einer Reihe von Schritten besteht. TinkerPop Bei diesen Schritten, die Teil des TinkerPop Open-Source-Apache-Projekts sind, handelt es sich sowohl um logische als auch um physische Operatoren, die eine Gremlin-Traversal in der Referenzimplementierung ausmachen. Sie werden beide verwendet, um das Modell der Abfrage darzustellen. Sie sind ausführbare Operatoren, die Lösungen gemäß der Semantik des Operators, den sie darstellen, erstellen können. Zum Beispiel `.V()` wird durch den sowohl repräsentiert als auch ausgeführt. TinkerPop GraphStep

Da diese TinkerPop Standardschritte ausführbar sind, kann ein solcher TinkerPop Traversal jede Gremlin-Abfrage ausführen und die richtige Antwort liefern. Wenn TinkerPop Schritte jedoch anhand

eines großen Graphen ausgeführt werden, können sie manchmal sehr ineffizient und langsam sein. Anstatt sie zu verwenden, versucht Neptune, die Traversierung in ein deklaratives Format zu konvertieren, das aus Gruppen von Mustern besteht, wie zuvor beschrieben.

Neptune unterstützt zurzeit nicht alle Gremlin-Operatoren (Schritte) in der nativen Abfrage-Engine. Daher wird versucht, so viele Schritte wie möglich in eine einzelne `NeptuneGraphQueryStep` zu reduzieren, die den deklarativen logischen Abfrageplan für alle konvertierten Schritte enthält. Idealerweise werden alle Schritte konvertiert. Wenn jedoch ein Schritt angetroffen wird, der nicht konvertiert werden kann, bricht Neptune die native Ausführung ab und verschiebt die gesamte Abfrageausführung von diesem Zeitpunkt an auf die Schritte. TinkerPop Es versucht kein In- oder Outweaving bezüglich der nativen Ausführung.

Nachdem die Schritte in einen logischen Abfrageplan umgesetzt wurden, führt Neptune eine Reihe von Abfrageoptimierern aus, die den Abfrageplan basierend auf statischen Analysen und geschätzter Kardinalität neu schreiben. Diese Optimierer erledigen Aufgaben wie die Neuanordnung von Operatoren basierend auf der Bereichsanzahl, das Reduzieren unnötiger oder redundanter Operatoren, das Neuanordnen von Filtern, das Verschieben von Operatoren in verschiedene Gruppen und so weiter.

Nach der Erstellung eines optimierten Abfrageplan erstellt Neptune eine Pipeline mit physischen Operatoren zur Ausführung der Abfrage. Dies umfasst das Lesen von Daten aus den Anweisungsindizes, das Durchführen von Joins verschiedener Typen, das Filtern, Sortieren usw. Die Pipeline erzeugt einen Lösungsstream, der dann wieder in einen Stream von TinkerPop Traverser-Objekten konvertiert wird.

Serialisierung von Abfrageergebnissen

Amazon Neptune verwendet derzeit die Serialisierer für TinkerPop Antwortnachrichten, um Abfrageergebnisse (TinkerPop Traverser) in serialisierte Daten umzuwandeln, die über das Kabel zurück an den Client gesendet werden. Diese Serialisierungsformate sind in der Regel recht ausführlich.

Um beispielsweise das Ergebnis einer Eckpunktabfrage wie `g.V().limit(1)` zu serialisieren, muss die Neptune-Abfrage-Engine eine einzelne Suche ausführen, um das Abfrageergebnis zu erzeugen. Der `GraphSON`-Serializer würde jedoch eine große Anzahl zusätzlicher Suchanfragen durchführen, um den Eckpunkt in das Serialisierungsformat zu verpacken. Er müsste eine Suche durchführen, um die Bezeichnung zu erhalten, eine, um die Eigenschaftsschlüssel abzurufen, und eine Suche pro Eigenschaftsschlüssel für den Eckpunkt, um alle Werte für jeden Schlüssel abzurufen.

Einige der Serialisierungsformate sind effizienter, aber alle erfordern zusätzliche Suchvorgänge. Darüber hinaus versuchen die TinkerPop Serialisierer nicht, doppelte Suchanfragen zu vermeiden, was oft dazu führt, dass viele Suchanfragen unnötig wiederholt werden.

Dies macht es sehr wichtig, Ihre Abfragen so zu schreiben, dass sie nur nach den benötigten Informationen fragen. Beispielsweise würde `g.V().limit(1).id()` nur die Eckpunkt-ID zurückgeben und alle zusätzlichen Serialisierer-Suchen entfernen. Mit Gremlin-Profil-API in Neptune können Sie sehen, wie viele Suchaufrufe während der Abfrageausführung und während der Serialisierung durchgeführt werden.

Verwendung der Gremlin **`Explain`** API in Neptune

Die Amazon-Neptune-Gremlin-`explain`-API gibt den Abfrageplan zurück, der ausgeführt würde, wenn eine bestimmte Abfrage ausgeführt würde. Da die API die Abfrage nicht tatsächlich ausführt, wird der Plan fast sofort zurückgegeben.

Er unterscheidet sich vom Schritt TinkerPop .explain (), um spezifische Informationen für die Neptune-Engine melden zu können.

Informationen, die in einem Gremlin Explain-Bericht enthalten sind

Der `explain`-Bericht enthält die folgenden Informationen:

- Die Abfragezeichenfolge, wie gewünscht.
- Die ursprüngliche Traversierung. Dies ist das TinkerPop Traversal-Objekt, das durch das Parsen der Abfragezeichenfolge in Schritten erzeugt wird. TinkerPop Es entspricht der ursprünglichen Abfrage, die erstellt wurde, indem die Abfrage gegen die `.explain()` ausgeführt wurde. TinkerPop TinkerGraph
- Die konvertierte Traversierung. Dies ist der Neptun-Traversal, der durch die Konvertierung der TinkerPop Traversal in die logische Neptun-Abfrageplandarstellung erzeugt wird. In vielen Fällen wird die gesamte TinkerPop Durchquerung in zwei Neptun-Schritte umgewandelt: einen, der die gesamte Abfrage ausführt (`NeptuneGraphQueryStep`) und einen, der die Ausgabe der Neptune-Abfrage-Engine wieder in Traversers () umwandelt. TinkerPop `NeptuneTraverserConverterStep`
- Die optimierte Traversierung. Dies ist die optimierte Version des Neptune-Abfrageplans, nachdem dieser von mehreren statischen, den Aufwand reduzierenden Optimierern verarbeitet wurde, die die Abfrage basierend auf statischen Analysen und geschätzten Kardinalitäten neu schreiben. Diese Optimierer erledigen Aufgaben wie die Neuanordnung von Operatoren basierend auf der

Bereichsanzahl, das Reduzieren unnötiger oder redundanter Operatoren, das Neuanordnen von Filtern, das Verschieben von Operatoren in verschiedene Gruppen und so weiter.

- Die Anzahl der Prädikate. Aufgrund der zuvor beschriebenen Neptune-Indizierungsstrategie können zahlreiche unterschiedliche Prädikate zu Leistungsproblemen führen. Dies gilt insbesondere für Abfragen, die Reverse-Transversal-Operatoren ohne Grenzbezeichnung (`.in` oder `.both`) verwenden. Wenn solche Operatoren verwendet werden und die Anzahl der Prädikate hoch genug ist, zeigt der `explain`-Bericht eine Warnmeldung an.
- DFE-Informationen. Wenn die alternative DFE-Engine aktiviert ist, können die folgenden Traversierungskomponenten in der optimierten Traversierung erscheinen:
  - **`DFEStep`** – Ein Neptune-optimierter DFE-Schritt in der Traversierung mit einem untergeordneten `DFENode`. `DFEStep` stellt den Teil des Abfrageplans dar, der in der DFE-Engine ausgeführt wird.
  - **`DFENode`** – Enthält die Zwischendarstellung als einen oder mehrere untergeordnete `DFEJoinGroupNodes`.
  - **`DFEJoinGroupNode`** – Stellt eine Verbindung von einem oder mehreren `DFENode`- oder `DFEJoinGroupNode`-Elementen dar.
  - **`NeptuneInterleavingStep`** – Ein Neptune-optimierter DFE-Schritt in der Traversierung mit einem untergeordneten `DFEStep`.

    Enthält außerdem ein `stepInfo`-Element mit Informationen zur Traversierung, z. B. das Grenzelement, die verwendeten Pfadelemente usw. Diese Informationen werden zur Verarbeitung des untergeordneten `DFEStep` verwendet.

  Eine einfache Art, festzustellen, ob Ihre Abfrage von DFE ausgewertet wird, besteht in der Überprüfung, ob die `explain`-Ausgabe einen `DFEStep` enthält. Jeder Teil der Durchquerung, der nicht Teil von ist, wird nicht von DFE ausgeführt, sondern von der `DFEStep` Engine ausgeführt. TinkerPop

  Einen Beispielbericht finden Sie unter Beispiel mit DFE-Aktivierung.

`Gremlin erklärt die Syntax`

Die Syntax der `explain` API ist dieselbe wie die der HTTP-API für Abfragen, außer dass sie anstelle von `/gremlin/explain` als Endpunkt verwendet wird`/gremlin`, wie in den folgenden Beispielen.

AWS CLI

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

Weitere Informationen finden Sie unter execute-gremlin-explain-query in der Befehlsreferenz. 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'])
```

AWS SDK-Beispiele in anderen Sprachen wie Java, .NET und mehr finden Sie unter. AWS 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**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

Weitere Informationen zur Verwendung awscurl mit der IAM-Authentifizierung finden Sie unter. Verwenden von `awscurl` mit temporären Anmeldeinformationen, um eine sichere Verbindung zu einem DB-Cluster mit aktivierter IAM-Authentifizierung herzustellen

curl

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

Die vorherige Abfrage würde die folgende Ausgabe erzeugen.

```
*******************************************************
                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
```

Nicht konvertierte Schritte TinkerPop

Im Idealfall werden alle TinkerPop Schritte einer Traversierung von einem eigenen Neptun-Operator abgedeckt. Wenn dies nicht der Fall ist, greift Neptune aufgrund von Lücken in der TinkerPop Bedienerabdeckung auf die Step-Ausführung zurück. Wenn eine Traversierung einen Schritt verwendet, für den Neptune noch keine native Abdeckung besitzt, zeigt der `explain`-Bericht eine Warnung an, die angibt, wo die Lücke aufgetreten ist.

Wenn ein Schritt ohne einen entsprechenden nativen Neptun-Operator angetroffen wird, wird die gesamte Traversierung von diesem Punkt an mithilfe von Schritten ausgeführt, auch wenn nachfolgende TinkerPop Schritte native Neptun-Operatoren haben.

Eine Ausnahme bilden Aufrufe von Neptune-Volltextsuchen. Der NeptuneSearchStep implementiert Schritte ohne systemeigene Entsprechungen als Volltextsuchschritte.

Beispiel für eine **`Explain-Ausgabe`**, bei der alle Schritte in einer Abfrage systemeigene Entsprechungen haben

Dies ist ein Beispielbericht für `explain` für eine Abfrage, in der es für alle Schritte native Entsprechungen gibt.

```
*******************************************************
                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
```

Beispiel, in dem es für einige Abfrageschritte keine nativen Entsprechungen gibt

Neptune verarbeitet `GraphStep` und `VertexStep` nativ. Wenn Sie jedoch `FoldStep` und `UnfoldStep` einführen, unterscheidet sich die `explain`-Ausgabe:

```
********************************************************
                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
```

In diesem Fall hebt `FoldStep` die native Ausführung auf. Aber auch die nachfolgende `VertexStep` wird nicht mehr nativ behandelt, da sie den `Fold`/`Unfold`-Schritten nachgelagert erscheint.

Um Leistung und Kosteneinsparungen zu erzielen, ist es wichtig, dass Sie versuchen, Traversals so zu formulieren, dass der größtmögliche Arbeitsaufwand nativ in der Neptune-Abfrage-Engine erledigt wird, anstatt schrittweise Implementierungen vorzunehmen. TinkerPop

Beispiel für eine Abfrage mit Verwendung der Neptune-Volltextsuche

Die folgende Abfrage verwendet die Neptune-Volltextsuche:

```
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*")
```

Der Teil `.has("name", "Neptune#fts mark*")` beschränkt die Suche auf Eckpunkte mit `name`, während `.has("Person", "name", "Neptune#fts mark*")` die Suche auf Eckpunkte mit `name` und der Beschriftung `Person` beschränkt. Dies führt zur folgenden Traversierung im `explain`-Bericht:

```
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}
    }
}]
```

Beispiel für die Verwendung von Explain bei aktiviertem DFE

Dies ist ein Beispiel für einen `explain`-Bericht bei Aktivierung der alternativen DFE-Abfrage-Engine:

```
********************************************************
                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=92233720368547758O7}]
        }, {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=92233720368547758O7}],
```

```
            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
```

Eine Beschreibung der DFE-specific Abschnitte im Bericht finden Informationen in `explain` Sie unter.

`Gremlin-Profil-API` in Neptune

Die Neptune-Gremlin-`profile`-API führt eine angegebene Gremlin-Traversierung aus, erfasst Metriken zur Ausführung und erstellt einen Profilbericht als Ausgabe.

Er unterscheidet sich vom Schritt TinkerPop .profile (), um spezifische Informationen für die Neptune-Engine melden zu können.

Der Profilbericht enthält die folgenden Informationen über den Abfrageplan:

- Die physische Operator-Pipeline
- Die Indexoperationen für die Abfrageausführung und Serialisierung
- Die Größe des Ergebnisses

Die `profile`-API verwendet eine erweiterte Version der HTTP-API-Syntax für Abfragen mit `/gremlin/profile` als Endpunkt anstelle von `/gremlin`.

Spezifische Parameter für das Neptun-Gremlin-Profil

- profile.results – `boolean`, zulässige Werte: `TRUE` und `FALSE`, Standardwert: `TRUE`.

  Falls true, werden die Abfrageergebnisse gesammelt und als Teil des `profile`-Berichts angezeigt. Falls false, wird nur die Ergebniszahl angezeigt.

- profile.chop – `int`, Standardwert: 250.

  Wenn der Wert nicht Null ist, wird die Ergebniszeichenfolge mit dieser Anzahl von Zeichen abgeschnitten. Dadurch werden nicht alle Ergebnisse erfasst. Es wird einfach die Größe der Zeichenfolge im Profilbericht begrenzt. Wenn auf Null gesetzt, enthält die Zeichenfolge alle Ergebnisse.

- profile.serializer – `string`, Standardwert: `<null>`.

  Wenn der Wert nicht null ist, werden die gesammelten Ergebnisse in einer serialisierten Antwortnachricht in dem von diesem Parameter angegebenen Format zurückgegeben. Die Anzahl der Indexoperationen, die zum Erstellen dieser Antwortnachricht erforderlich sind, wird zusammen mit der Größe in Bytes angegeben, die an den Client gesendet werden soll.

  Zulässige Werte sind `<null>` oder einer der gültigen Enumerationswerte des MIME-Typs oder TinkerPop -Treibers „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`, zulässige Werte: `TRUE` und `FALSE`, Standardwert: `FALSE`.

  Bei „true“ wird ein detaillierter Bericht aller Indexoperationen angezeigt, die während der Abfrageausführung und -serialisierung durchgeführt wurden. Warnung: Dieser Bericht kann sehr ausführlich sein.

Beispielausgabe des Neptune Gremlin-Profils

Das folgende Beispiel zeigt eine mögliche `profile`-Abfrage:

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"
```

Weitere Informationen finden Sie unter execute-gremlin-profile-query in der Befehlsreferenz. 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'])
```

AWS SDK-Beispiele in anderen Sprachen wie Java, .NET und mehr finden Sie unter. AWS 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**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

Weitere Informationen zur Verwendung awscurl mit der IAM-Authentifizierung finden Sie unter. Verwenden von `awscurl` mit temporären Anmeldeinformationen, um eine sichere Verbindung zu einem DB-Cluster mit aktivierter IAM-Authentifizierung herzustellen

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"}'
```

Diese Abfrage generiert den folgenden `profile`- Bericht, wenn sie auf dem Luftrouten-Beispieldiagramm aus dem Blogbeitrag Let Me Graph That For You – Part 1 – Air Routes ausgeführt wird.

```
*******************************************************
                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
```

Zusätzlich zu den Abfrageplänen, die durch einen Aufruf von Neptune `explain` zurückgegeben werden, enthalten die `profile`-Ergebnisse Laufzeitstatistiken zur Abfrageausführung. Jede Join-Operation wird mit der Zeit markiert, die für die Ausführung des Joins erforderlich war, sowie mit der tatsächlichen Anzahl der Lösungen, die sie durchlaufen haben.

Die `profile`-Ausgabe enthält die Zeit, die für der Core-Abfrageausführung erforderlich war, sowie die Serialisierungsphase, wenn die `profile.serializer`-Option angegeben wurde.

Die Aufschlüsselung der Indexoperationen, die während jeder Phase ausgeführt werden, ist ebenfalls unten in der `profile`-Ausgabe enthalten.

Beachten Sie, dass aufeinander folgende Ausführungen derselben Abfrage aufgrund des Cachings unterschiedliche Ergebnisse in Bezug auf Laufzeit- und Indexoperationen anzeigen können.

Für Abfragen, die den `repeat()`-Schritt verwenden, ist eine Aufschlüsselung der Grenze bei jeder Iteration verfügbar, wenn der `repeat()`-Schritt als Teil einer `NeptuneGraphQueryStep` heruntergeschoben wurde.

Unterschiede in den **`Profilberichten`**, wenn DFE aktiviert ist

Wenn die alternative Neptune-DFE-Abfrage-Engine aktiviert ist, unterscheidet sich die `profile`-Ausgabe etwas:

Optimierte Traversierung: Dieser Abschnitt ist dem Abschnitt in der `explain`-Ausgabe ähnlich, enthält jedoch zusätzliche Informationen. Dazu gehören die Art der DFE-Operatoren, die bei der Planung berücksichtigt wurden, und die Kostenschätzungen für den ungünstigsten und den günstigsten Fall.

Physische Pipeline: Dieser Abschnitt erfasst die Operatoren, die zur Ausführung der Abfrage verwendet werden. `DFESubQuery`-Elemente abstrahieren den physischen Plan, der von DFE verwendet wird, um den Teil des Plans auszuführen, für den sie jeweils verantwortlich sind. Die `DFESubQuery`-Elemente werden im folgenden Abschnitt mit der Auflistung von DFE-Statistiken angezeigt.

DFEQueryEngine Statistik: Dieser Abschnitt wird nur angezeigt, wenn mindestens ein Teil der Abfrage von DFE ausgeführt wird. Er beschreibt verschiedene, für DFE spezifische Laufzeitstatistiken und enthält eine detaillierte Analyse der Zeit nach `DFESubQuery`, die für die verschiedenen Teile der Abfrageausführung aufgewendet wurde.

Verschachtelte Unterabfragen in verschiedenen `DFESubQuery`-Elementen werden in diesem Abschnitt vereinfacht dargestellt. Eindeutige Bezeichner sind mit einem Header gekennzeichnet, der mit `subQuery=` beginnt.

Traversal-Metriken: In diesem Abschnitt werden Traversal-Metriken auf Schrittebene angezeigt. Wenn die DFE-Engine die Abfrage ganz oder teilweise ausführt, werden Metriken für angezeigt. `DFEStep` and/or `NeptuneInterleavingStep` Siehe [Optimieren von Gremlin-Abfragen mithilfe von Explain und Profile].

> **Note**
>
> Da es sich bei der DFE-Unterstützung für Gremlin um eine experimentelle Funktion handelt, kann sich das genaue Format der Ausgabe ändern. `profile`

`Beispielprofilausgabe`, wenn die Neptune Dataflow Engine (DFE) aktiviert ist

Wenn zur Ausführung von Gremlin-Abfragen die DFE-Engine verwendet wird, wird die Ausgabe von Gremlin-`profile`-API wie folgt formatiert.

Abfrage:

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()"
```

Weitere Informationen finden Sie unter execute-gremlin-profile-query in der Befehlsreferenz. 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'])
```

AWS SDK-Beispiele in anderen Sprachen wie Java, .NET und mehr finden Sie unter. AWS 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**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

Weitere Informationen zur Verwendung awscurl mit der IAM-Authentifizierung finden Sie unter. Verwenden von `awscurl` mit temporären Anmeldeinformationen, um eine sichere Verbindung zu einem DB-Cluster mit aktivierter IAM-Authentifizierung herzustellen

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=9223372036854775807})],
             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,o
                 alt=(type=PipelineScan,
cost=(exp=(in=1.00,out=27.76,io=Infinity,comp=0.00,mem=295147905179352830000.00),wc=(in=1.00,o
         } [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**
>
> Da es sich bei der DFE-Unterstützung für Gremlin um eine experimentelle Funktion handelt, kann sich das genaue Format der `profile` Ausgabe ändern.

Optimieren von Gremlin-Abfragen mithilfe von Explain und Profile

Sie können Ihre Gremlin-Abfragen in Amazon Neptune häufig zur Erzielung einer besseren Leistung optimieren, indem Sie die Informationen in den Berichten aus den Neptune-APIs explain und profile verwenden. Daher sollten Sie wissen, wie Neptune Gremlin-Traversierungen verarbeitet.

> **Important**
>
> In TinkerPop Version 3.4.11 wurde eine Änderung vorgenommen, die die Korrektheit der Verarbeitung von Abfragen verbessert, die Abfrageleistung jedoch vorerst ernsthaft beeinträchtigen kann.
> Zum Beispiel könnte eine Abfrage dieser Art deutlich langsamer ausgeführt werden:
>
> ```
> g.V().hasLabel('airport').
>   order().
>     by(out().count(),desc).
>   limit(10).
> ```

```
  out()
```

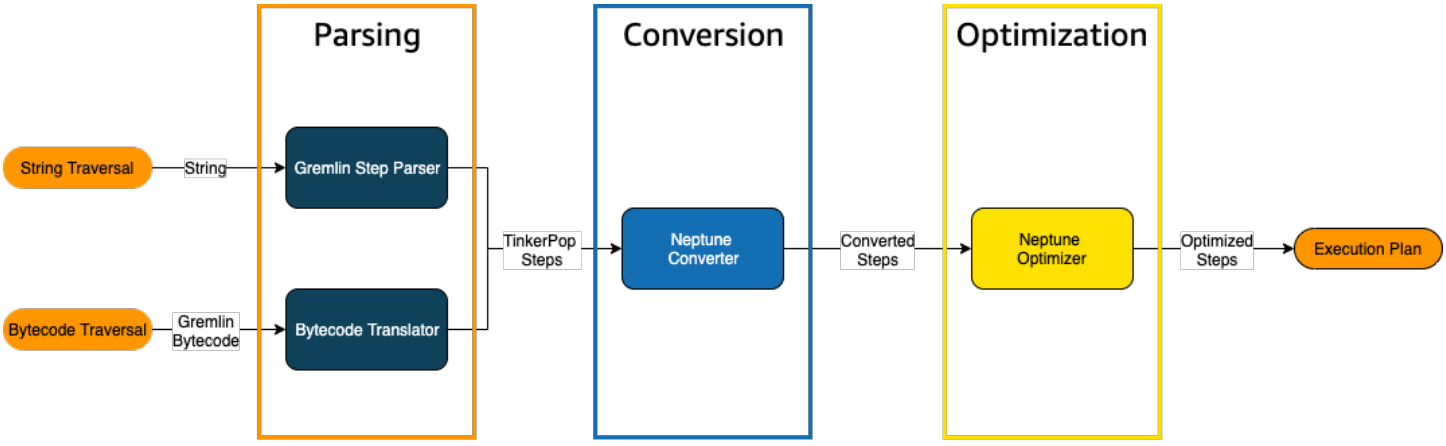
Die Scheitelpunkte nach dem Limit-Schritt werden jetzt aufgrund der Änderung in 3.4.11 nicht optimal abgerufen. TinkerPop Um dies zu vermeiden, können Sie die Abfrage ändern, indem Sie den Schritt barrier() an einer beliebigen Stelle nach `order().by()` hinzufügen. Beispiel:

```
g.V().hasLabel('airport').
  order().
    by(out().count(),desc).
  limit(10).
  barrier().
  out()
```

TinkerPop 3.4.11 wurde in der Neptune-Engine-Version 1.0.5.0 aktiviert.

Informationen zur Verarbeitung von Gremlin-Traversierungen in Neptune

Wenn eine Gremlin-Traversierung an Neptune gesendet wird, gibt es drei Hauptprozesse für die Transformierung der Traversierung in einen zugrunde liegenden Ausführungsplan zur Ausführung durch die Engine. Diese Schritte sind Parsing, Konvertierung und Optimierung:



Parsing von Traversierungen

Der erste Schritt bei der Verarbeitung einer Traversierung besteht darin, sie in eine gemeinsame Sprache zu parsen. In Neptune besteht diese gemeinsame Sprache aus einer Reihe von TinkerPop Schritten, die Teil der TinkerPop API sind. Jeder dieser Schritte stellt eine Recheneinheit innerhalb der Traversierung dar.

Sie können eine Gremlin-Traversierung als Zeichenfolge oder als Bytecode an Neptune senden. Der REST-Endpunkt und die Java-Client-Treibermethode `submit()` senden Traversierungen als Zeichenfolgen, wie in diesem Beispiel:

```
client.submit("g.V()")
```

Anwendungen und Sprachtreiber, die Gremlin Language Variants (GLV) verwenden, senden Traversierungen als Bytecode.

Konvertierung von Traversierungen

Der zweite Schritt bei der Verarbeitung einer Traversierung besteht darin, ihre TinkerPop Schritte in eine Reihe von konvertierten und nicht umgewandelten Neptunschritten umzuwandeln. Die meisten Schritte in der Apache TinkerPop Gremlin-Abfragesprache werden in Neptune-specific Schritte umgewandelt, die für die Ausführung auf der zugrunde liegenden Neptune-Engine optimiert sind. Wenn bei einer Durchquerung ein TinkerPop Schritt ohne Neptun-Äquivalent gefunden wird, werden dieser Schritt und alle nachfolgenden Schritte der Durchquerung von der Abfrage-Engine verarbeitet. TinkerPop

Weitere Informationen zu den Schritten, die unter bestimmten Umständen konvertiert werden können, finden Sie unter Unterstützung für Gremlin-Schritte.

Optimierung von Traversierungen

Der letzte Schritt bei der Verarbeitung der Traversierung besteht darin, die konvertierten und nicht konvertierten Schritte durch den Optimierer laufen zu lassen, um den besten Ausführungsplan zu ermitteln. Das Ergebnis dieser Optimierung ist der Ausführungsplan, den die Neptune-Engine verarbeitet.

Verwenden der Neptune Gremlin **`Explain`** API zur Optimierung von Abfragen

Die Neptune-API explain ist nicht mit dem Gremlin-Schritt `explain()` identisch. Sie gibt den endgültigen Ausführungsplan zurück, den die Neptune-Engine bei Ausführung der Abfrage verarbeiten würde. Da sie keine Verarbeitung ausführt, gibt sie unabhängig von den verwendeten Parametern denselben Plan zurück und die Ausgabe enthält keine Statistiken zu einer tatsächlichen Ausführung.

Betrachten Sie die folgende einfache Traversierung, die alle Flughafen-Eckpunkte für Anchorage sucht:

```
g.V().has('code','ANC')
```

Es gibt zwei Möglichkeiten für die Ausführung dieser Traversierung über die Neptune-API `explain`. Die erste Möglichkeit besteht in einem REST-Aufruf an den explain-Endpunkt wie folgt:

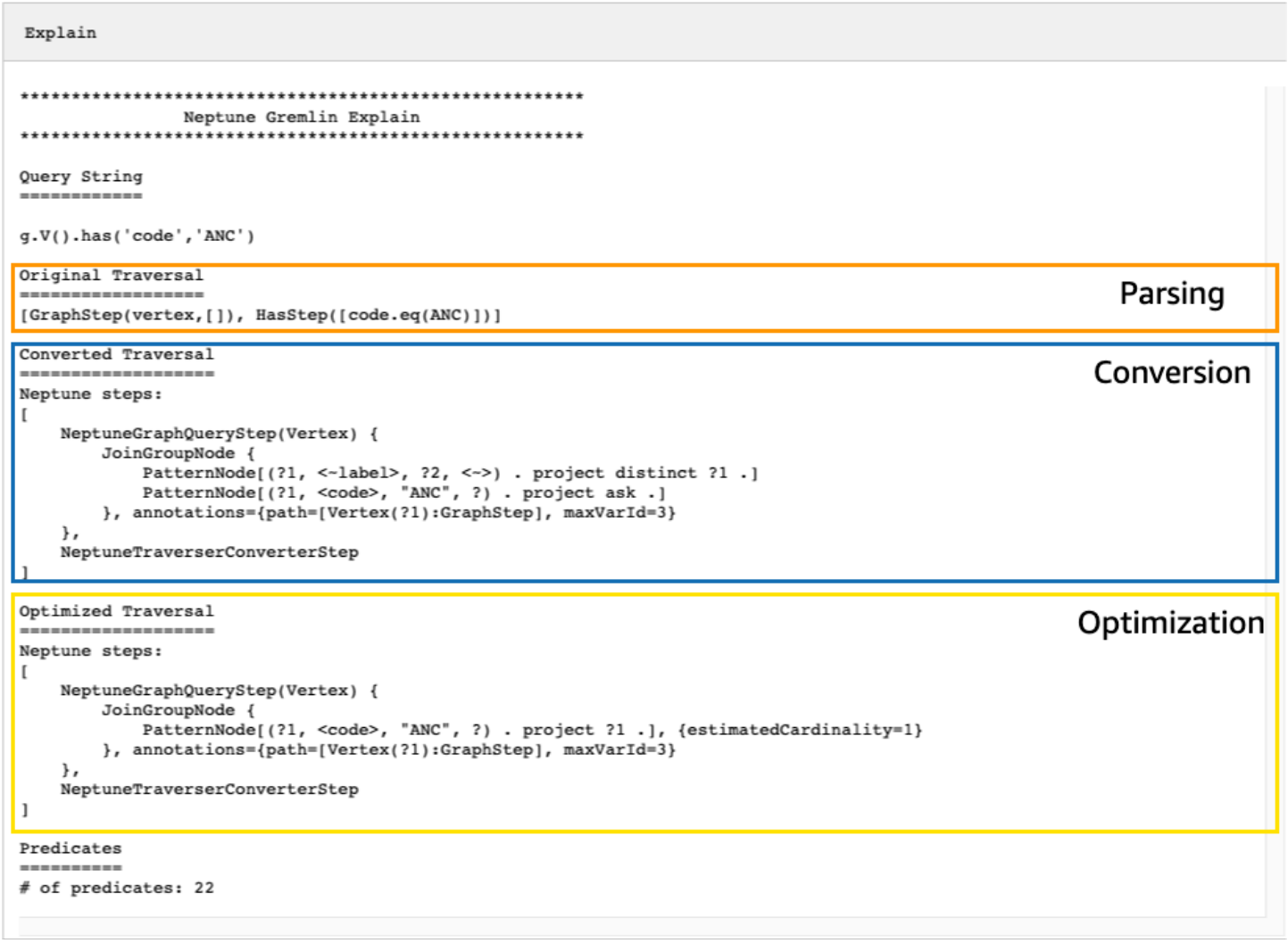
```
curl -X POST https://your-neptune-endpoint:port/gremlin/explain -d
 '{"gremlin":"g.V().has('code','ANC')"}'
```

Die zweite Möglichkeit besteht in der Verwendung der Zellen-Magics %%gremlin der Neptune-Workbench mit dem Parameter `explain`. Hierdurch werden die Traversierung in der Zelle an die Neptune-API `explain` übergeben und die Ausgabe angezeigt, wenn Sie die Zelle ausführen:

```
%%gremlin explain

g.V().has('code','ANC')
```

Die resultierende Ausgabe der `explain`-API beschreibt den Neptune-Ausführungsplan für die Traversierung. Wie Sie in der Abbildung unten sehen können, umfasst der Plan jeden der 3 Schritte in der Verarbeitungspipeline:



Traversierungsoptimierung durch Untersuchung nicht konvertierter Schritte

Zu den ersten Dingen, nach denen Sie in der Ausgabe der Neptune-API `explain` suchen sollten, gehören Gremlin-Schritte, die nicht in native Neptune-Schritte konvertiert wurden. Wenn in einem Abfrageplan ein Schritt gefunden wird, der nicht in einen nativen Neptune-Schritt konvertiert werden kann, werden dieser und alle nachfolgenden Schritte im Plan durch den Gremlin-Server verarbeitet.

Im Beispiel oben wurden alle Schritte der Traversierung konvertiert. Betrachten wir die Ausgabe der API `explain` für diese Traversierung:

```
g.V().has('code','ANC').out().choose(hasLabel('airport'), values('code'), constant('Not
 an airport'))
```

Wie Sie in der Abbildung unten sehen können, konnte Neptune den Schritt `choose()` nicht konvertieren:

```
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
```

Es gibt verschiedene Möglichkeiten, wie Sie die Leistung der Traversierung optimieren können. Die erste Möglichkeit besteht darin, sie ohne den Schritt neu zu schreiben, der nicht konvertiert werden konnte. Eine andere Möglichkeit besteht darin, den Schritt an das Ende der Traversierung zu verschieben, damit alle anderen Schritte in native Schritte konvertiert werden können.

Ein Abfrageplan mit Schritten, die nicht konvertiert werden können, muss nicht immer optimiert werden. Wenn sich die Schritte, die nicht konvertiert werden können, am Ende der Traversierung befinden und eher die Formatierung der Ausgabe als die Art der Traversierung des Diagramms betreffen, haben sie möglicherweise nur geringe Auswirkungen auf die Leistung.

Sie sollten bei der Untersuchung der Ausgabe der Neptune-API `explain` auch auf Schritte achten, die keine Indizes verwenden. Die folgende Traversierung sucht nach allen Flughäfen mit Flügen, die in Anchorage landen:

```
g.V().has('code','ANC').in().values('code')
```

Die Ausgabe der Explain-API für diese Traversierung ist wie folgt:

```
*******************************************************
                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
```

Die `WARNING`-Meldung am Ende der Ausgabe erscheint, weil der Schritt `in()` in der Traversierung nicht mit einem der 3 Indizes behandelt werden kann, die Neptune verwaltet (siehe Indizierung von Anweisungen in Neptune und Gremlin-Anweisungen in Neptune). Da der Schritt `in()` keinen Kantenfilter enthält, kann er nicht mit dem Index `SPOG`, `POGS` oder `GPSO` aufgelöst werden. Stattdessen muss Neptune einen Union-Scan ausführen, um die angeforderten Eckpunkte zu suchen, was sehr viel ineffizienter ist.

In dieser Situation gibt es zwei Möglichkeiten für die Optimierung der Traversierung. Die erste Möglichkeit besteht in der Hinzufügung eines oder mehrerer Kriterien zum Schritt `in()`, um eine indizierte Suche zur Auflösung der Abfrage zu verwenden. Für das Beispiel oben könnte das sein:

```
g.V().has('code','ANC').in('route').values('code')
```

Die Ausgabe der Neptune-API `explain` für die überarbeitete Traversierung enthält nicht mehr die `WARNING`-Meldung:

```
****************************************************
                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
```

Wenn Sie zahlreiche Traversierungen dieser Art ausführen, besteht eine weitere Möglichkeit darin, sie in einem Neptune-DB-Cluster auszuführen, für den der optionale OSGP-Index aktiviert ist (siehe Aktivieren eines OSGP-Indexes). Die Aktivierung eines OSGP-Indexes hat Nachteile:

- Er muss im DB-Cluster aktiviert werden, bevor Daten geladen werden.
- Die Einfügeraten für Eckpunkte und Kanten können um bis zu 23 % verlangsamt werden.
- Die Speichernutzung wird um ungefähr 20 % steigen.
- Leseabfragen, die Anforderungen über alle Indizes verteilen, können zu einer höheren Latenz führen.

Ein OSGP-Index ist für einen eingeschränkten Satz von Abfragemustern sehr sinnvoll. Wenn Sie diese jedoch nicht häufig ausführen, ist es in der Regel besser, die geschriebenen Traversierungen mithilfe der drei primären Indizes aufzulösen.

Verwenden einer großen Anzahl von Prädikaten

Neptune behandelt jede Kantenbezeichnung und jeden eindeutigen Eckpunkt- oder Kanteneigenschaftsnamen in Ihrem Diagramm als Prädikat. ist jedoch für die Verarbeitung einer relativ geringen Anzahl unterschiedlicher Prädikate vorgesehen. Wenn Ihre Diagrammdaten mehr als einige tausend Prädikate enthalten, kann sich die Leistung verschlechtern.

Die Neptune-Ausgabe für `explain` warnt Sie, wenn dies der Fall ist:

```
Predicates
==========
# of predicates: 9549
WARNING: high predicate count (# of distinct property names and edge labels)
```

Wenn eine Überarbeitung des Datenmodells zur Reduzierung der Anzahl der Bezeichnungen und Eigenschaften nicht praktikabel ist, können Sie Traversierungen am besten optimieren, indem Sie diese in einem DB-Cluster mit aktiviertem OSGP-Index ausführen, wie oben beschrieben.

Verwenden der Neptune Gremlin **`Profile`** API zur Optimierung von Traversals

Die Neptune-API `profile` unterscheidet sich erheblich vom Gremlin-Schritt `profile()`. Wie bei der `explain`-API enthält die Ausgabe den Abfrageplan, den die Neptune-Engine bei der Ausführung der Traversierung verwendet. Zusätzlich enthält die `profile`-Ausgabe tatsächliche Ausführungsstatistiken für die Traversierung, abhängig von der Einstellung der Parameter.

Betrachten Sie erneut die folgende einfache Traversierung, die alle Flughafen-Eckpunkte für Anchorage sucht:

```
g.V().has('code','ANC')
```

Wie im Fall der `explain`-API können Sie die `profile`-API mit einem REST-Aufruf aufrufen:

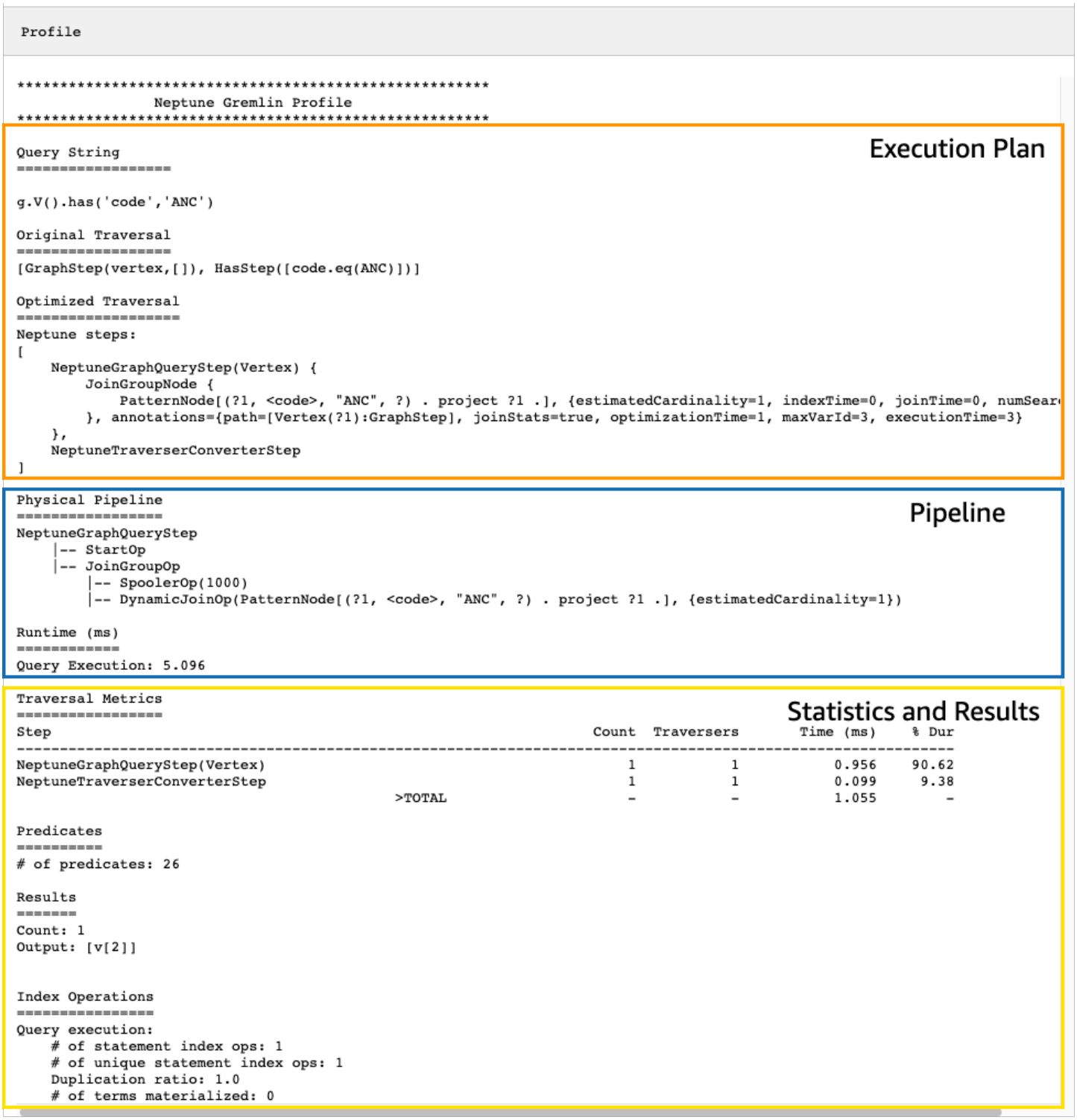
```
curl -X POST https://your-neptune-endpoint:port/gremlin/profile -d
 '{"gremlin":"g.V().has('code','ANC')"}'
```

Sie verwenden auch die Zellen-Magics %%gremlin der Neptune-Workbench mit dem Parameter `profile`. Hierdurch werden die Traversierung in der Zelle an die Neptune-API `profile` übergeben und die Ausgabe angezeigt, wenn Sie die Zelle ausführen:

```
%%gremlin profile

g.V().has('code','ANC')
```

Die resultierende Ausgabe der `profile`-API enthält sowohl den Neptune-Ausführungsplan für die Traversierung als auch Statistiken zur Ausführung des Plans, wie diese Abbildung zeigt:




```
Profile

************************************************************
                Neptune Gremlin Profile
************************************************************

Query String
==================

g.V().has('code','ANC')

Original Traversal
==================
[GraphStep(vertex,[]), HasStep([code.eq(ANC)])]

Optimized Traversal
===================
Neptune steps:
[
    NeptuneGraphQueryStep(Vertex) {
        JoinGroupNode {
            PatternNode[(?1, <code>, "ANC", ?) . project ?1 .], {estimatedCardinality=1, indexTime=0, joinTime=0, numSear
        }, annotations={path=[Vertex(?1):GraphStep], joinStats=true, optimizationTime=1, maxVarId=3, executionTime=3}
    },
    NeptuneTraverserConverterStep
]

Physical Pipeline
=================
NeptuneGraphQueryStep
    |-- StartOp
    |-- JoinGroupOp
        |-- SpoolerOp(1000)
        |-- DynamicJoinOp(PatternNode[(?1, <code>, "ANC", ?) . project ?1 .], {estimatedCardinality=1})

Runtime (ms)
============
Query Execution: 5.096

Traversal Metrics
=================
Step                                                               Count  Traversers       Time (ms)    % Dur
-------------------------------------------------------------------------------------------------------------
NeptuneGraphQueryStep(Vertex)                                          1           1           0.956    90.62
NeptuneTraverserConverterStep                                          1           1           0.099     9.38
                                            >TOTAL                     -           -           1.055        -

Predicates
==========
# of predicates: 26

Results
=======
Count: 1
Output: [v[2]]


Index Operations
================
Query execution:
    # of statement index ops: 1
    # of unique statement index ops: 1
    Duplication ratio: 1.0
    # of terms materialized: 0
```

In der `profile`-Ausgabe enthält der Abschnitt zum Ausführungsplan nur den endgültigen Ausführungsplan für die Traversierung, keine Zwischenschritte. Der Abschnitt zur Pipeline enthält die physischen Pipeline-Operationen, die ausgeführt wurden, sowie die tatsächliche Dauer (in

Millisekunden) der Ausführung der Traversierung. Die Laufzeitmetrik ist äußerst nützlich, um während der Optimierung die Dauer von zwei verschiedenen Traversierungsversionen zu vergleichen.

> **Note**
>
> Die anfängliche Laufzeit einer Traversierung ist in der Regel länger als die nachfolgenden Laufzeiten, da während der ersten Laufzeit die relevanten Daten im Cache gespeichert werden.

Der dritte Abschnitt der `profile`-Ausgabe enthält Ausführungsstatistiken und die Ergebnisse der Traversierung. Die folgende Traversierung zeigt, wie diese Informationen für die Optimierung einer Traversierung nützlich sein können. Sie sucht alle Flughäfen, deren Name mit „Anchora“ beginnt, und alle Flughäfen, die in zwei Hops von diesen Flughäfen aus erreichbar sind. Anschließend werden Flughafencodes, Flugrouten und Entfernungen zurückgegeben:

```
%%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'))
```

Traversal-Metriken in der Neptune-Profil-API-Ausgabe

Der erste Satz von Metriken, der in allen `profile`-Ausgaben verfügbar ist, sind die Traversierungsmetriken. Sie sind den Metriken für den Gremlin-Schritt `profile()` ähnlich, mit einigen Unterschieden:

```
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        -
```

Die erste Spalte der Tabelle für Traversierungsmetriken listet die Schritte auf, die durch die Traversierung ausgeführt werden. Die ersten beiden Schritte sind im Allgemeinen die Neptune-specific Schritte, und. `NeptuneGraphQueryStep` `NeptuneTraverserConverterStep`

`NeptuneGraphQueryStep` stellt die Ausführungszeit für den gesamten Teil der Traversierung dar, der von der Neptune-Engine konvertiert und nativ ausgeführt werden konnte.

`NeptuneTraverserConverterStep`stellt den Prozess dar, bei dem die Ausgabe dieser konvertierten Schritte in TinkerPop Transverser umgewandelt wird, sodass Schritte, die nicht konvertiert werden konnten, verarbeitet werden können, falls vorhanden, oder die Ergebnisse in einem TinkerPop-compatible Format zurückgegeben werden können.

Im obigen Beispiel haben wir mehrere nicht konvertierte Schritte, sodass wir sehen, dass jeder dieser TinkerPop Schritte (`ProjectStep`,`PathStep`) dann als Zeile in der Tabelle erscheint.

In der zweiten Spalte der Tabelle wird die Anzahl der dargestellten Durchquerer angegeben`Count`, die den Schritt durchlaufen haben, während in der dritten Spalte die Anzahl der Durchquerer angegeben wird`Traversers`, die diesen Schritt durchlaufen haben, wie in der Dokumentation zum Profilschritt erklärt. TinkerPop

Im Beispiel werden 3.856 Eckpunkte und 3.856 Traverser von `NeptuneGraphQueryStep` zurückgegeben. Diese Zahlen bleiben während der gesamten restlichen Verarbeitung gleich, da `ProjectStep` und `PathStep` die Ergebnisse formatieren, nicht filtern.

> **Note**
>
> Im TinkerPop Gegensatz dazu optimiert der Neptune-Motor die Leistung nicht, indem er seine und seine *`NeptuneGraphQueryStep`* Stufen vergrößert. `NeptuneTraverserConverterStep` Bulking ist eine TinkerPop Operation, bei der Traversen auf demselben Scheitelpunkt kombiniert werden, um den Betriebsaufwand zu reduzieren. Dadurch unterscheiden sich die Zahlen und. `Count` `Traversers` Da Bulking nur in Schritten erfolgt, an die Neptune delegiert TinkerPop, und nicht in Schritten, die Neptune nativ verarbeitet, unterscheiden sich die Spalten und D selten. `Count` `Traverser`

Die Spalte „Zeit“ meldet die Anzahl der Millisekunden, die für den Schritt benötigt wurden. Die Spalte `% Dur` meldet, wie viel Prozent der gesamten Verarbeitungszeit der Schritt in Anspruch genommen hat. Diese Metriken zeigen Ihnen, worauf Sie sich bei der Optimierung konzentrieren sollten, da sie die zeitaufwändigsten Schritte identifizieren.

Indexbetriebsmetriken in der **`Neptune-Profil-API-Ausgabe`**

Ein weiterer Satz von Metriken in der Ausgabe der Neptune-API „profile“ sind die Indexoperationen:

```
Index Operations
================
Query execution:
    # of statement index ops: 23191
    # of unique statement index ops: 5960
    Duplication ratio: 3.89
    # of terms materialized: 0
```

Diese geben Folgendes an:

- Die Gesamtzahl der Indexsuchvorgänge.
- Die Anzahl der eindeutigen ausgeführten Indexsuchvorgänge.
- Das Verhältnis zwischen der Gesamtzahl der Indexsuchvorgänge und den eindeutigen Indexsuchvorgängen. Je niedriger diese Zahl ist, desto weniger Redundanz gibt es.
- Die Anzahl der Begriffe aus dem Begriffsverzeichnis.

Metriken in der **`Neptune-Profil-API-Ausgabe`** wiederholen

Wenn die Traversierung den Schritt `repeat()` wie im Beispiel oben gezeigt verwendet, erscheint in der `profile`-Ausgabe ein Abschnitt mit Wiederholungsmetriken:

```
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
```

Diese geben Folgendes an:

- Die Anzahl der Schleifen für eine Zeile (Spalte `Iteration`).
- Die Anzahl der von der Schleife aufgerufenen Elemente (Spalte `Visited`).
- Die Anzahl der von der Schleife ausgegebenen Elemente (Spalte `Output`).
- Das letzte von der Schleife ausgegebene Element (Spalte `Until`).
- Die Anzahl der von der Schleife ausgesendeten Elemente (Spalte `Emit`).
- Die Anzahl der Elemente, die von der Schleife an die nachfolgende Schleife übergeben wurden (Spalte `Next`).

Diese Wiederholungsmetriken sind sehr nützlich, um den Verzweigungsfaktor der Traversierung zu verstehen und ein Gefühl dafür zu bekommen, wie viel Arbeit von der Datenbank geleistet wird. Sie können diese Zahlen für die Diagnose von Leistungsproblemen verwenden, insbesondere wenn die Leistung derselben Traversierung mit unterschiedlichen Parametern stark unterschiedlich ist.

Full-text Suchmetriken in der **`Neptune-Profil-API-Ausgabe`**

Wenn bei einer Traversierung eine Volltextsuche verwendet wird, wie im Beispiel oben, wird in der `profile`-Ausgabe ein Abschnitt mit Metriken für die Volltextsuche (Full Text Search, FTS) angezeigt:

```
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
```

Hier wird die an den ElasticSearch (ES) -Cluster gesendete Anfrage angezeigt und es werden verschiedene Metriken über die Interaktion mit ElasticSearch dieser angezeigt, anhand derer Sie Leistungsprobleme im Zusammenhang mit der Volltextsuche lokalisieren können:

- Zusammenfassende Informationen zu den Aufrufen des ElasticSearch Index:
  - Die gesamte Anzahl der Millisekunden, die alle remoteCalls benötigten, um die Abfrage (`total`) zu erfüllen.

  - Die durchschnittliche Anzahl der Millisekunden, die für einen remoteCall (`avg`) aufgewendet wurden.
  - Die Mindestanzahl der Millisekunden, die für einen remoteCall (`min`) aufgewendet wurden.
  - Die maximale Anzahl der Millisekunden, die für einen remoteCall (`max`) aufgewendet wurden.
- Gesamtzeit, die von RemoteCalls to ElasticSearch () `remoteCallTime` verbraucht wurde.
- Die Anzahl der RemoteCalls an (). ElasticSearch `remoteCalls`
- Die Anzahl der Millisekunden, die für Zusammenführungen von Ergebnissen () aufgewendet wurden. ElasticSearch `joinTime`
- Die Anzahl der Millisekunden, die für Index-Lookups (`indexTime`) aufgewendet wurden.
- Die Gesamtzahl der von () zurückgegebenen ElasticSearch Ergebnisse. `remoteResults`

Native Unterstützung für Gremlin-Schritte in Amazon Neptune

Die Amazon-Neptune-Engine bietet zurzeit keine vollständige native Unterstützung für alle Gremlin-Schritte, wie in Optimieren von Gremline-Abfragen beschrieben. Die aktuelle Unterstützung lässt sich in vier Kategorien einteilen:

- Gremlin-Schritte, die jederzeit in native Neptune-Engine-Operationen konvertiert werden können
- Gremlin-Schritte, die in einigen Fällen in native Neptune-Engine-Operationen konvertiert werden können
- Gremlin-Schritte, die niemals in native Neptune-Engine-Operationen konvertiert werden können
- Gremlin-Schritte, die in Neptune überhaupt nicht unterstützt werden

Gremlin-Schritte, die jederzeit in native Neptune-Engine-Operationen konvertiert werden können

Viele Gremlin-Schritte können in native Neptune-Engine-Operationen konvertiert werden, solange sie die folgenden Bedingungen erfüllen:

- Ihnen geht in der Abfrage kein Schritt voraus, der nicht konvertiert werden kann.
- Der übergeordnete Schritt, wenn vorhanden, kann konvertiert werden.
- Alle untergeordneten Schritte, wenn vorhanden, können konvertiert werden.

Die folgenden Gremlin-Schritte können stets in native Neptune-Engine-Operationen konvertiert werden, solange sie die folgenden Bedingungen erfüllen:

- 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( )

Gremlin-Schritte, die in einigen Fällen in native Neptune-Engine-Operationen konvertiert werden können

Einige Gremlin-Schritte können in einigen Situationen in native Neptune-Engine-Operationen konvertiert werden, in anderen nicht:

- addE( ) – Der Schritt `addE()` kann grundsätzlich in eine native Neptune-Engine-Operation konvertiert werden, es sei denn, der unmittelbar folgende Schritt ist ein `property()`-Schritt mit einer Traversierung als Schlüssel.
- addV( ) – Der Schritt `addV()` kann grundsätzlich in eine native Neptune-Engine-Operation konvertiert werden, es sei denn, ihm folgt unmittelbar ein `property()`-Schritt, der eine Traversierung als Schlüssel enthält oder es sind mehrere Bezeichnungen zugewiesen.
- aggregate( ) – Der Schritt `aggregate()` kann grundsätzlich in eine native Neptune-Engine-Operation konvertiert werden, es sei denn, der Schritt wird in einer untergeordneten Traversierung oder Subtraversierung verwendet oder es handelt sich beim gespeicherten Wert nicht um einen Eckpunkt-, Kanten-, ID-, Bezeichnungs- oder Eigenschaftswert.

  Im folgenden Beispiel wird `aggregate()` nicht konvertiert, da er in einer untergeordneten Traversierung verwendet wird:

  ```
  g.V().has('code','ANC').as('a')
       .project('flights').by(select('a')
       .outE().aggregate('x'))
  ```

  In diesem Beispiel wird aggregate( ) nicht konvertiert, da `min()` gespeichert wird:

  ```
  g.V().has('code','ANC').outE().aggregate('x').by(values('dist').min())
  ```

- barrier( ) – Der Schritt `barrier()` kann grundsätzlich in eine native Neptun-Engine-Operation konvertiert werden, es sei denn, der folgende Schritt wird nicht konvertiert.
- cap( ) – Der einzige Fall, in dem der Schritt `cap()` konvertiert wird, ist in Kombination mit dem Schritt `unfold()`, um die nicht entfaltete Version eines Aggregats von Eckpunkt-, Kanten-, ID- oder Eigenschaftswerten zurückzugeben. In diesem Beispiel wird `cap()` konvertiert, weil `.unfold()` folgt:

  ```
  g.V().has('airport','country','IE').aggregate('airport').limit(2)
       .cap('airport').unfold()
  ```

  Wenn Sie `.unfold()` entfernen, wird `cap()` jedoch nicht konvertiert:

  ```
  g.V().has('airport','country','IE').aggregate('airport').limit(2)
       .cap('airport')
  ```

- coalesce () — Der `coalesce( )` Schritt wird nur konvertiert, wenn er dem auf der Rezeptseite empfohlenen Upsert-Muster folgt. TinkerPop Andere coalesce( )-Muster sind nicht erlaubt. Die Konvertierung erfolgt nur, wenn alle untergeordneten Traversierungen konvertiert werden können, alle denselben Typ als Ausgabe erzeugen (Eckpunkt, Kante, ID, Wert, Schlüssel oder Bezeichnung), alle zu einem neuen Element traversieren und sie nicht den Schritt `repeat( )` enthalten.
- constant( ) – Der Schritt constant() wird zurzeit nur konvertiert, wenn er innerhalb des `sack( ).by( )`-Teils einer Traversierung verwendet wird, um einen konstanten Wert zuzuweisen, z. B.:

```
g.V().has('code','ANC').sack(assign).by(constant(10)).out().limit(2)
```

- cyclicPath( ) – Der Schritt `cyclicPath( )` kann grundsätzlich in eine native Neptune-Engine-Operation konvertiert werden, es sei denn, der Schritt wird mit den Modulatoren `by( )`, `from( )` oder `to( )` verwendet. In den folgenden Abfragen wird `cyclicPath( )` beispielsweise nicht konvertiert:

```
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( ) – Der Schritt `drop( )` kann grundsätzlich in eine native Neptune-Engine-Operation konvertiert werden, es sei denn, der Schritt wird innerhalb des Schritts `sideEffect(` oder `optional( )` verwendet.
- fold () — Es gibt nur zwei Situationen, in denen der Schritt fold () konvertiert werden kann, nämlich wenn er in dem Upsert-Muster verwendet wird, das auf der TinkerPop Rezeptseite empfohlen wird, und wenn er in einem `group( ).by( )` Kontext wie diesem verwendet wird:

```
g.V().has('code','ANC').out().group().by().by(values('code', 'city').fold())
```

- has () — Der Schritt `has () `kann im Allgemeinen in eine native Neptune-Engine-Operation umgewandelt werden, vorausgesetzt, Abfragen mit `T` verwenden das Prädikat` `,` `oder` P.eq `. P.neq P.contains Erwarte Variationen von `has () `, die implizieren, dass diese Instanzen von `P` ebenfalls in native Versionen konvertiert werden, wie zum Beispiel `hasID ('id1234')`, was `has (eq, T.id, 'id1234') `entspricht.
- id( ) – Der Schritt `id( )` wird konvertiert, wenn er nicht in einer Eigenschaft verwendet wird, z. B.:

```
g.V().has('code','ANC').properties('code').id()
```

- mergeE () — Der `mergeE()` Schritt kann in eine native Neptune-Engine-Operation umgewandelt werden, wenn die Parameter (die Zusammenführungsbedingung, on`Create` undon`Match`) konstant sind (entweder `null` eine Konstante `Map` oder `select()` von a). `Map` Alle Beispiele in Upserting Edges können konvertiert werden.
- mergeV () — Der Schritt mergeV () kann in eine native Neptune-Engine-Operation umgewandelt werden, wenn die Parameter (die Zusammenführungsbedingung, on`Create` undon`Match`) konstant sind (entweder `null` eine Konstante oder von a). `Map` `select()` `Map` Alle Beispiele, bei denen Scheitelpunkte nach oben verschoben werden, können konvertiert werden.
- order( ) – Der Schritt `order()` kann grundsätzlich in eine native Neptun-Engine-Operation konvertiert werden, es sei denn, eine der folgenden Bedingungen wird erfüllt:
  - Der Schritt `order()` ist Teil einer verschachtelten untergeordneten Traversierung wie folgt:

    ```
    g.V().has('code','ANC').where(V().out().order().by(id))
    ```

  - Es wird eine lokale Anordnung verwendet, zum Beispiel mit `order(local)`.
  - In der Modulation `by()`, nach der sortiert werden werden soll, wird ein benutzerdefinierter Komparator verwendet. Ein Beispiel ist diese Verwendung von `sack()`:

    ```
    g.withSack(0).
      V().has('code','ANC').
          repeat(outE().sack(sum).by('dist').inV()).times(2).limit(10).
          order().by(sack())
    ```

  - Es gibt mehrere Anordnungen für dasselbe Element.
- project( ) – Der Schritt `project()` kann grundsätzlich in eine native Neptune-Engine-Operation konvertiert werden, es sei denn, die Anzahl der `by()`-Anweisungen, die auf `project()` folgen, stimmt nicht mit der Anzahl der angegebenen Bezeichnungen überein, z. B.:

  ```
  g.V().has('code','ANC').project('x', 'y').by(id)
  ```

- range( ) – Der Schritt `range()` wird nur konvertiert, wenn das untere Ende des jeweiligen Bereichs null ist (z. B. `range(0,3)`).
- repeat( ) – Der Schritt `repeat()` kann grundsätzlich in eine native Neptune-Engine-Operation konvertiert werden, es sei denn, er ist in einem anderen `repeat()`-Schritt verschachtelt, z. B.:

```
g.V().has('code','ANC').repeat(out().repeat(out()).times(2)).times(2)
```

- sack( ) – Der Schritt `sack()` kann grundsätzlich in eine native Neptune-Engine-Operation konvertiert werden, außer in den folgenden Fällen:
  - Wenn ein nicht-numerischer sack-Operator verwendet wird.
  - Wenn ein anderer numerischer sack-Operator als `+`, `-`, `mult`, `div`, `min` und `max` verwendet wird.
  - Wenn `sack()` innerhalb eines `where()`-Schritts verwendet wird, um nach einem sack-Wert zu filtern, z. B.:

    ```
    g.V().has('code','ANC').sack(assign).by(values('code')).where(sack().is('ANC'))
    ```

- sum( ) – Der Schritt `sum()` kann grundsätzlich in eine native Neptune-Engine-Operation konvertiert werden, jedoch nicht, wenn er zur Berechnung einer globalen Summierung verwendet wird, z. B.:

  ```
  g.V().has('code','ANC').outE('routes').values('dist').sum()
  ```

- union( ) – Der Schritt `union()` kann in eine native Neptune-Engine-Operation konvertiert werden, solange er abgesehen vom letzten Schritt der letzte Schritt in der Abfrage ist.
- unfold () — Der `unfold()` Schritt kann nur dann in eine native Neptune-Engine-Operation umgewandelt werden, wenn er in dem auf der TinkerPopRezeptseite empfohlenen Upsert-Muster verwendet wird und wenn er zusammen mit dem folgenden verwendet wird: `cap()`

  ```
  g.V().has('airport','country','IE').aggregate('airport').limit(2)
       .cap('airport').unfold()
  ```

- where( ) – Der Schritt `where()` kann grundsätzlich in eine native Neptune-Engine-Operation konvertiert werden, außer in den folgenden Fällen:
  - Wenn by()-Modulationen verwendet werden, z. B.:

    ```
    g.V().hasLabel('airport').as('a')
         .where(gt('a')).by('runways')
    ```

  - Wenn andere Vergleichsoperatoren als `eq`, `neq`, `within` und `without` verwendet werden.
  - Wenn vom Benutzer bereitgestellte Aggregationen verwendet werden.

Gremlin-Schritte, die niemals in native Neptune-Engine-Operationen konvertiert werden können

Die folgenden Gremlin-Schritte werden in Neptune unterstützt, können jedoch nie in native Neptune-Engine-Operationen konvertiert werden. Sie werden stattdessen vom Gremlin-Server ausgeführt.

- choose( )
- coin( )
- scheitern ()
- inject( )
- match( )
- math( )
- max( )
- mean( )
- min( )
- option( )
- optional( )
- path( )
- propertyMap( )
- sample( )
- skip( )
- tail( )
- timeLimit( )
- tree( )

Gremlin-Schritte, die in Neptune überhaupt nicht unterstützt werden

Die folgenden Gremlin-Schritte werden in Neptune überhaupt nicht unterstützt. In den meisten Fällen liegt dies daran, dass sie `GraphComputer` erfordern, was Neptune derzeit nicht unterstützt.

- connectedComponent( )
- io( )
- shortestPath( )
- withComputer( )

- pageRank( )
- peerPressure( )
- program( )

Der Schritt `io()` wird tatsächlich teilweise unterstützt, da er für `read()` über eine URL verwendet werden kann, nicht jedoch für `write()`.

Verwenden von Gremlin mit der Neptune-DFE-Abfrage-Engine

Sie können die alternative Abfrage-Engine von Neptune, die sogenannte DFE, auf zwei Arten aktivieren:

- Setzen Sie den `neptune_dfe_query_engine`Instanzparameter `enabled` für alle Abfragen auf der Instanz auf.
- Verwenden Sie den `useDFE`Abfragehinweis für einzelne Abfragen (`neptune_dfe_query_engine`muss auf den Standardwert gesetzt sein`viaQueryHint`).

Wenn DFE aktiviert ist, übersetzt Neptune schreibgeschützte Gremlin-Abfragen und -Traversals in eine logische Zwischendarstellung und führt sie, wann immer möglich, auf der DFE-Engine aus.

Die DFE unterstützt jedoch noch nicht alle Gremlin-Schritte. Wenn ein Schritt nicht nativ auf dem DFE ausgeführt werden kann, greift Neptune darauf zurück, TinkerPop um den Schritt auszuführen. Die Berichte `explain` und `profile` enthalten Warnungen, wenn dies auftritt.

Berichterstattung über Gremlin-Schritte in DFE

Gremlin DFE ist eine experimentelle Funktion und kann verwendet werden, indem entweder der Instanzparameter aktiviert oder der Abfragehinweis verwendet wird. `Neptune#useDFE` Weitere Informationen finden Sie unter Gremlin mit der Neptune DFE-Abfrage-Engine verwenden.

Die folgenden Schritte können in Gremlin DFE verwendet werden.

Pfad- und Durchquerungsschritte:

asDate (), barrier (), call (), cap (), dateAdd (), dateDiff (), disjunct (), drop (), fail (), filter (), flatMap (), id (), identity (), index (), intersect (), inject (), label (), length (), loops (), map (), order () lokal), Pfad (), project (), range (), repeat (), reverse (), sack (), sample (), select (), sideEffect (), split (), unfold (), union ()

Schritte zur Aggregation und Erfassung:

aggregieren (global), kombinieren (), zählen (), dedup (), dedup (lokal), fold (), group (), groupCount (),

Mathematische Schritte:

max (), Mittelwert (), min (), Summe ()

Schritte der Elemente:

AndereV (), elementMap (), element (), v (), out (), in (), beide (), oute (), inE (), sowohl E (), outV (), inV (), beide V (), otherV ()

Schritte zur Immobilie:

Eigenschaften (), Schlüssel (), ValueMap (), Wert ()

Schritte filtern:

and (), coalesce (), coin (), has (), is (), local (), none (), not (), or (), where ()

Schritte zur Bearbeitung von Zeichenketten:

concat (), lTrim (), rTrim (), substring (), toLower (), toUpper (), trim ()

Prädikate:

- Vergleiche: eq, neq, lt, lte, gt, gte
- Enthält: innerhalb, ohne
- TextP: endend mit, enthaltend, nicht, nichtStartingWith, nichtenthaltend EndingWith
- P: und, oder, zwischen, draußen, drinnen

Einschränkungen

Repeat with Limit, Labels innerhalb, Repeat Traversal und Dedup werden in DFE noch nicht unterstützt.

```
// 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'))
```

Pfade mit verschachtelten Wiederholungen oder Verzweigungsschritten werden noch nicht unterstützt.

```
// 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')
```

Überlappung der Abfrageplanung

Wenn der Übersetzungsprozess einen Gremlin-Schritt ohne entsprechenden nativen DFE-Operator versucht er vor dem Rückgriff auf TinkerPop, andere zwischengeschaltete Abfrageteile zu finden, die nativ in der DFE-Engine ausgeführt werden können. Hierzu wird auf die Traversierung auf der obersten Ebene eine Überlappungslogik angewendet. So werden unterstützte Schritte verwendet, wann immer möglich.

Jede solche zwischengeschaltete Abfrageübersetzung ohne Präfix wird mit `NeptuneInterleavingStep` in den `explain`- und `profile`-Ausgaben dargestellt.

Zum Leistungsvergleich sollten Sie die Überlappung in einer Abfrage deaktivieren und die DFE-Engine weiter verwenden, um den Präfixteil auszuführen. Oder vielleicht möchten Sie nur die TinkerPop Engine für die Ausführung von Abfragen ohne Präfix verwenden. Sie können dies mithilfe des Abfragehinweises `disableInterleaving` tun.

So wie der Abfragehinweis useDFE mit dem Wert `false` verhindert, dass eine Abfrage überhaupt in der DFE ausgeführt wird, deaktiviert der Abfragehinweis `disableInterleaving` mit dem Wert `true` die DFE-Überlappung für die Übersetzung einer Abfrage. Beispiel:

```
g.with('Neptune#disableInterleaving', true)
```

```
.V().has('genre','drama').in('likes')
```

Die Gremlin-Erklärung und die Profilausgabe wurden aktualisiert

Gremlin explain stellt Details zu der optimierten Traversierung bereit, die Neptune für die Ausführung einer Abfrage verwendet. In der DFE-Beispielausgabe für `explain` finden Sie ein Beispiel für die `explain`-Ausgabe, wenn die DFE-Engine aktiviert ist.

Die Gremlin-`profile`-API führt eine angegebene Gremlin-Traversierung durch, sammelt verschiedene Metriken zur Ausführung und erstellt einen profile-Bericht mit Details zum optimierten Abfrageplan und zu den Laufzeitstatistiken verschiedener Operatoren. In der DFE-Beispielausgabe für `profile` finden Sie ein Beispiel für die `profile`-Ausgabe, wenn die DFE-Engine aktiviert ist.

> **Note**
>
> Da es sich bei der DFE-Unterstützung für Gremlin um eine experimentelle Funktion handelt, kann sich das genaue Format der `profile` UND-Ausgabe `explain` ändern.

Zugriff auf das Neptun-Diagramm mit openCypher

Neptune unterstützt die Erstellung von Diagrammanwendungen mit openCypher, die zu den zurzeit beliebtesten Abfragesprachen für Entwickler gehört, die mit Graphdatenbanken arbeiten. Entwickler, Geschäftsanalysten und Datenwissenschaftler schätzen die SQL-inspired Syntax von OpenCypher, weil sie eine vertraute Struktur zum Verfassen von Abfragen für Diagrammanwendungen bietet.

openCypher ist eine deklarative Abfragesprache für Eigenschaftsdiagramme. Ursprünglich von Neo4j entwickelt, wurde sie 2015 als Open-Source-Software veröffentlicht und ist unter einer Apache 2-Open-Source-Lizenz für das openCypher-Projekt verfügbar. Die Syntax ist in der Cypher Query Language Reference, Version 9 dokumentiert.

Die Einschränkungen und Unterschiede der Neptune-Unterstützung für die openCypher-Spezifikation finden Sie unter Einhaltung der OpenCypher-Spezifikationen in Amazon Neptune.

> **Note**
>
> Die aktuelle Neo4j-Implementierung der Cypher-Abfragesprache unterscheidet sich in einigen Punkten von der OpenCypher-Spezifikation. Wenn Sie aktuellen Neo4j-Cypher-Code zu

Neptune migrieren, finden Sie unter Neptune-Kompatibilität mit Neo4j und Umschreiben von Cypher-Abfragen zur Ausführung in openCypher auf Neptune weitere Informationen.

Ab Engine-Version 1.1.1.0 ist openCypher für die produktive Verwendung in Neptune verfügbar.

Gremlin und openCypher: Ähnlichkeiten und Unterschiede

Gremlin und openCypher sind beides Abfragesprachen für Eigenschaftsdiagramme und ergänzen sich in vielen Hinsichten.

Gremlin wurde für Programmierer entwickelt und fügt sich nahtlos in Code ein. Daher ist Gremlin von Natur aus imperativ, während die deklarative Syntax von openCypher für Entwickler mit SQL- oder SPARQL-Erfahrung möglicherweise vertrauter wirkt. Gremlin wirkt vielleicht für Datenwissenschaftler natürlicher, die Python in einem Jupyter-Notebook verwenden, während openCypher für geschäftliche Anwender mit SQL-Hintergrund vielleicht intuitiver wirkt.

Das Gute ist, dass Sie sich nicht zwischen Gremlin und openCypher entscheiden müssen. Abfragen in einer der beiden Sprachen können für dasselbe Diagramm ausgeführt werden, unabhängig davon, welche der beiden Sprachen zur Eingabe dieser Daten verwendet wurde. Abhängig von der Aufgabe ist es möglicherweise vorteilhafter, Gremlin für einige Dinge und openCypher für andere Dinge zu verwenden.

Gremlin verwendet eine imperative Syntax, mit der Sie die Schritte steuern können, mit denen Sie durch Ihr Diagramm navigieren. Jeder Schritt nimmt einen Datenstrom auf, führt eine Aktion für ihn aus (mithilfe eines Filters, einer Map usw.) und gibt dann die Ergebnisse für den nächsten Schritt aus. Eine Gremlin-Abfrage hat in der Regel das Format `g.V()`, gefolgt von zusätzlichen Schritten.

In openCypher verwenden Sie eine SQL-inspirierte deklarative Syntax, die ein Muster von Knoten und Beziehungen angibt, das in Ihrem Diagramm mithilfe einer motif-Syntax gesucht werden soll (wie `()-[]->()`). Eine openCypher-Abfrage beginnt häufig mit einer `MATCH`-Klausel, gefolgt von anderen Klauseln wie `WHERE`, `WITH` und `RETURN`.

Erste Schritte mit openCypher

Sie können Eigenschaftsdiagrammdaten in Neptune mit openCypher unabhängig davon abfragen, wie sie geladen wurden. Sie können openCypher jedoch nicht verwenden, um Daten abzufragen, die als RDF geladen wurden.

Der Neptune-Massen-Loader akzeptiert Eigenschaftsdiagrammdaten in einem CSV-Format für Gremlin und in einem CSV-Format für openCypher. Natürlich können Sie Ihrem Diagramm auch Eigenschaftsdaten hinzufügen, indem Sie and/or Gremlin OpenCypher-Abfragen verwenden.

Es sind zahlreiche Online-Tutorials zum Erlernen der Cypher-Abfragesprache verfügbar. Einige kurze Beispiele für openCypher-Abfragen helfen Ihnen vielleicht, sich ein Bild von der Sprache zu machen. Der mit Abstand beste und einfachste Weg, mit der Verwendung von openCypher zum Abfragen Ihres Neptune-Diagramms zu beginnen, ist jedoch die Verwendung der openCypher-Notebooks in der Neptune-Workbench. Die Workbench ist Open Source und wird unter gehostet. GitHub https://github.com/aws-samples/amazon-neptune-samples

Sie finden die OpenCypher-Notizbücher im GitHub Neptune Graph-Notebook-Repository. Schauen Sie sich insbesondere die Air-routesVisualisierung und die Notizbücher der englischen Premier Teams für OpenCypher an.

Die von OpenCypher verarbeiteten Daten haben die Form einer ungeordneten Reihe von Karten. key/value Die wichtigste Methode zur Verfeinerung, Bearbeitung und Erweiterung dieser Maps besteht in der Verwendung von Klauseln, die Aufgaben wie Musterabgleich, Einfügen, Aktualisieren und Löschen der Paare ausführen. key/value

In openCypher gibt es mehrere Klauseln für die Suche nach Datenmustern in Diagrammen, von denen `MATCH` am häufigsten verwendet wird. `MATCH` ermöglicht Ihnen die Angabe des Musters von Knoten, Beziehungen und Filtern, nach denen Sie in Diagrammen suchen möchten. Beispiel:

- Alle Knoten abrufen

```
MATCH (n) RETURN n
```

- Verbundene Knoten suchen

```
MATCH (n)-[r]->(d) RETURN n, r, d
```

- Einen Pfad finden

```
MATCH p=(n)-[r]->(d) RETURN p
```

- Alle Knoten mit einer Bezeichnung abrufen

```
MATCH (n:airport) RETURN n
```

Die erste Abfrage oben gibt jeden einzelnen Knoten in Ihrem Diagramm zurück. Die nächsten beiden Abfragen geben jeden Knoten zurück, der eine Beziehung hat. Dies wird generell jedoch nicht empfohlen! In beinahe allen Fällen sollten Sie die zurückgegebenen Daten eingrenzen. Hierzu können Sie Knoten- oder Beziehungsbezeichnungen und Eigenschaften angeben, wie im vierten Beispiel.

Eine praktische Übersicht über die openCypher-Syntax finden Sie im Github-Beispiel-Repository für Neptune.

Neptune-openCypher-Status-Servlet und Statusendpunkt

Der openCypher-Statusendpunkt bietet Zugriff auf Informationen zu Abfragen, die zurzeit auf dem Server ausgeführt werden oder auf die Ausführung warten. Sie können diese Abfragen auch abbrechen. Der Endpunkt ist:

```
https://(the server):(the port number)/openCypher/status
```

Sie können die HTTP-Methoden `GET` und `POST` verwenden, um den aktuellen Status vom Server abzurufen oder eine Abfrage abzubrechen. Sie können die Methode `DELETE` auch verwenden, um eine ausgeführte oder wartende Abfrage abzubrechen.

Parameter für Statusanforderungen

Statusabfrage-Parameter

- **`includeWaiting`** (`true` oder `false`) – Wenn auf `true` festgelegt und keine anderen Parameter vorhanden sind, werden Statusinformationen für wartende und ausgeführte Abfragen zurückgegeben.
- **`cancelQuery`** – Wird nur mit den Methoden `POST` und `GET` verwendet, um anzuzeigen, dass es sich um eine Abbruchanforderung handelt. Die Methode `DELETE` benötigt diesen Parameter nicht.

  Der Wert des Parameters `cancelQuery` wird nicht verwendet. Wenn jedoch `cancelQuery` vorhanden ist, ist der Parameter `queryId` erforderlich, um die Abfrage zu identifizieren, die abgebrochen werden soll.
- **`queryId`** – Enthält die ID einer bestimmten Abfrage.

  Wenn mit den Methoden `GET` oder `POST` verwendet und wenn der Parameter `cancelQuery` nicht vorhanden ist, veranlasst `queryId` die Rückgabe von Statusinformationen für die spezifische

identifizierte Abfrage. Wenn der Parameter `cancelQuery` vorhanden ist, wird die spezifische Abfrage abgebrochen, die `queryId` identifiziert.

Gibt bei Verwendung mit der Methode `DELETE` gibt `queryId` stets eine bestimmte Abfrage an, die abgebrochen werden soll.

- **`silent`** – Wird nur beim Abbruch einer Abfrage verwendet. Wenn auf `true` festgelegt, erfolgt der Abbruch im Hintergrund.

Statusanforderungs-Antwortfelder

Statusantwortfelder, wenn die ID einer bestimmten Abfrage nicht angegeben wird

- akzeptiert QueryCount — Die Anzahl der Abfragen, die akzeptiert, aber noch nicht abgeschlossen wurden, einschließlich Abfragen in der Warteschlange.
- laufend QueryCount — Die Anzahl der aktuell laufenden OpenCypher-Abfragen.
- queries – Eine Liste der aktuellen openCypher-Abfragen.

Statusantwortfelder für eine bestimmte Abfrage

- queryId – Eine GUID-ID für die Abfrage. Neptune weist diesen ID-Wert automatisch jeder Abfrage zu. Sie können auch eine eigene ID zuweisen (siehe Einfügen einer benutzerdefinierten ID in eine Neptune-Gremlin- oder -SPARQL-Abfrage).
- queryString – Die übermittelte Abfrage. Die Abfrage wird nach 1024 Zeichen abgeschnitten, wenn sie länger ist.
- query EvalStats — Statistiken für diese Abfrage:
  - waited – Der Zeitraum in Millisekunden, über den die Abfrage gewartet hat.
  - elapsed – Der Zeitraum in Millisekunden, über den die Abfrage bis jetzt ausgeführt wurde.
  - cancelled – `True` gibt an, dass die Abfrage abgebrochen wurde; `False` gibt an, dass sie nicht abgebrochen wurde.

Beispiele für Statusanforderungen und Antworten

- Anforderung des Status aller Abfragen, einschließlich wartender Abfragen:

AWS CLI

```
aws neptunedata get-open-cypher-query-status \
  --endpoint-url https://your-neptune-endpoint:port \
  --include-waiting
```

Weitere Informationen finden Sie unter get-open-cypher-query-status in der Befehlsreferenz. 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)
```

AWS SDK-Beispiele in anderen Sprachen finden Sie unter. AWS SDK

awscurl

```
awscurl https://your-neptune-endpoint:port/openCypher/status \
  --region us-east-1 \
  --service neptune-db \
  -X POST \
  -d "includeWaiting=true"
```

> **Note**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

curl

```
curl https://your-neptune-endpoint:port/openCypher/status \
  --data-urlencode "includeWaiting=true"
```

Antwort:

```
{
  "acceptedQueryCount" : 0,
  "runningQueryCount" : 0,
  "queries" : [ ]
}
```

- Anforderung des Status ausgeführter Abfragen, ohne wartende Abfragen:

  AWS CLI

  ```
  aws neptunedata get-open-cypher-query-status \
    --endpoint-url https://your-neptune-endpoint:port
  ```

  Weitere Informationen finden Sie unter get-open-cypher-query-status in der Befehlsreferenz. 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)
```

AWS SDK-Beispiele in anderen Sprachen finden Sie unter. AWS SDK

awscurl

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

> **Note**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

curl

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

Antwort:

```
{
  "acceptedQueryCount" : 0,
  "runningQueryCount" : 0,
  "queries" : [ ]
}
```

- Anforderung des Status einer einzelnen Abfrage:

  AWS CLI

  ```
  aws neptunedata get-open-cypher-query-status \
    --endpoint-url https://your-neptune-endpoint:port \
    --query-id eadc6eea-698b-4a2f-8554-5270ab17ebee
  ```

Weitere Informationen finden Sie unter get-open-cypher-query-status in der Befehlsreferenz. 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)
```

AWS SDK-Beispiele in anderen Sprachen finden Sie unter. AWS 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**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

curl

```
curl https://your-neptune-endpoint:port/openCypher/status \
```

```
  --data-urlencode "queryId=eadc6eea-698b-4a2f-8554-5270ab17ebee"
```

Antwort:

```
{
  "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
  }
}
```

- Anforderungen des Abbruchs einer Abfrage

  AWS CLI

  ```
  aws neptunedata cancel-open-cypher-query \
    --endpoint-url https://your-neptune-endpoint:port \
    --query-id f43ce17b-db01-4d37-a074-c76d1c26d7a9
  ```

  Weitere Informationen finden Sie unter cancel-open-cypher-query in der Befehlsreferenz. 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)
```

AWS SDK-Beispiele in anderen Sprachen finden Sie unter. AWS 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**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

curl

1. Verwenden von POST:

```
curl -X POST https://your-neptune-endpoint:port/openCypher/status \
  --data-urlencode "cancelQuery" \
  --data-urlencode "queryId=f43ce17b-db01-4d37-a074-c76d1c26d7a9"
```

2. Verwenden von GET:

```
curl -X GET https://your-neptune-endpoint:port/openCypher/status \
  --data-urlencode "cancelQuery" \
  --data-urlencode "queryId=588af350-cfde-4222-bee6-b9cedc87180d"
```

3. Verwenden von DELETE:

```
curl -X DELETE \
  "https://your-neptune-endpoint:port/openCypher/status?queryId=b9a516d1-
d25c-4301-bb80-10b2743ecf0e"
```

Antwort:

```
{
  "status" : "200 OK",
  "payload" : true
}
```

Der HTTPS-Endpunkt von Amazon Neptune OpenCypher

Themen

- OpenCypher Lese- und Schreibabfragen auf dem HTTPS-Endpunkt
- Das standardmäßige OpenCypher JSON-Ergebnisformat
- Optionale HTTP-Header am Ende für mehrteilige Antworten OpenCypher

> **Note**
>
> Neptune unterstützt HTTP/2 derzeit keine REST-API-Anfragen. Clients müssen dies verwenden, HTTP/1.1 wenn sie eine Verbindung zu Endpunkten herstellen.

OpenCypher Lese- und Schreibabfragen auf dem HTTPS-Endpunkt

Der OpenCypher HTTPS-Endpunkt unterstützt Lese- und Aktualisierungsabfragen, die `GET` sowohl die als auch die `POST` Methode verwenden. Die Methoden `DELETE` und `PUT` werden nicht unterstützt.

In den folgenden Anweisungen erfahren Sie, wie Sie mithilfe des `curl` Befehls und HTTPS eine Verbindung zum OpenCypher Endpunkt herstellen. Sie müssen diese Anweisungen für eine Amazon-EC2-Instance befolgen, die sich in derselben Virtual Private Cloud (VPC) wie Ihre Neptune-DB-Instance befindet.

Die Syntax lautet wie folgt:

```
HTTPS://(the server):(the port number)/openCypher
```

Hier ist ein Beispiel für eine Leseabfrage:

AWS CLI

```
aws neptunedata execute-open-cypher-query \
  --endpoint-url https://your-neptune-endpoint:port \
  --open-cypher-query "MATCH (n1) RETURN n1"
```

Weitere Informationen finden Sie unter execute-open-cypher-query in der Befehlsreferenz. 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'])
```

AWS SDK-Beispiele in anderen Sprachen finden Sie unter. AWS 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**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

curl

```
curl https://your-neptune-endpoint:port/openCypher \
  -d "query=MATCH (n1) RETURN n1"
```

Hier ist eine write/update Beispielabfrage:

AWS CLI

```
aws neptunedata execute-open-cypher-query \
  --endpoint-url https://your-neptune-endpoint:port \
  --open-cypher-query "CREATE (n:Person { age: 25 })"
```

Weitere Informationen finden Sie unter execute-open-cypher-query in der Befehlsreferenz. 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'])
```

AWS SDK-Beispiele in anderen Sprachen finden Sie unter. AWS 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**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

curl

```
curl https://your-neptune-endpoint:port/openCypher \
  -d "query=CREATE (n:Person { age: 25 })"
```

Das standardmäßige OpenCypher JSON-Ergebnisformat

Das folgende JSON-Format wird entweder standardmäßig oder durch explizites Festlegen des Anforderungs-Headers auf `Accept: application/json` zurückgegeben. Dieses Format soll mithilfe nativer Features der meisten Bibliotheken leicht in Objekte geparst werden können.

Das zurückgegebene JSON-Dokument enthält ein einzelnes Feld (`results`), das die Abfragerückgabewerte enthält. Die folgenden Beispiele zeigen die JSON-Formatierung häufiger Werte.

Beispiel für eine Wertantwort:

```
{
  "results": [
    {
      "count(a)": 121
    }
  ]
}
```

Beispiel für eine Knotenantwort:

```
{
  "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
        }
      }
    }
  ]
}
```


Beispiel für eine Beziehungsantwort:


```
{
  "results": [
    {
      "r": {
        "~id": "7389",
        "~entityType": "relationship",
        "~start": "22",
        "~end": "151",
        "~type": "route",
        "~properties": {
          "dist": 956
        }
      }
    }
  ]
}
```

Beispiel für eine Pfadantwort:


```
{
  "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
          }
        }
      ]
    }
  ]
}
```

Optionale HTTP-Header am Ende für mehrteilige Antworten OpenCypher

Diese Funktion ist ab Version 1.4.5.0 der Neptune Engine verfügbar.

Die HTTP-Antwort auf OpenCypher Abfragen und Aktualisierungen wird in der Regel in mehreren Abschnitten zurückgegeben. Wenn nach dem Senden der ersten Antwort-Chunks (mit dem HTTP-Statuscode 200) Fehler auftreten, kann es schwierig sein, das Problem zu diagnostizieren. Standardmäßig meldet `Neptune solche Fehler, indem es eine Fehlermeldung an den Nachrichtentext anhängt, der aufgrund des Streaming-Charakters der Antwort beschädigt sein kann.

Verwenden von abschließenden Headern

Um die Fehlererkennung und -diagnose zu verbessern, können Sie Trailing-Header aktivieren, indem Sie Ihrer Anfrage einen Trailer-Header (TE: Trailers) für Transfer-Encoding (TE) hinzufügen. Anschließend fügt Neptune zwei neue Header-Felder in die Trailing-Header der Antwortblöcke ein:

- `X-Neptune-Status`— enthält den Antwortcode gefolgt von einem Kurznamen. Im Erfolgsfall wäre der nachgestellte Header beispielsweise: `X-Neptune-Status: 200 OK`. Im Fehlerfall wäre der Antwortcode ein Neptune-Engine-Fehlercode wie. `X-Neptune-Status: 500 TimeLimitExceededException`

- `X-Neptune-Detail`— ist bei erfolgreichen Anfragen leer. Im Fehlerfall ist die JSON-Fehlermeldung enthalten. Da in HTTP-Header-Werten nur ASCII-Zeichen zulässig sind, ist die JSON-Zeichenfolge URL-codiert. Die Fehlermeldung wird weiter an die Antwortmeldung angefügt.

Weitere Informationen finden Sie auf der MDN-Seite über TE-Anforderungsheader.

OpenCypher Beispiel für die Verwendung von Trailing-Headern

Dieses Beispiel zeigt, wie abschließende Header bei der Diagnose einer Abfrage helfen, deren Zeitlimit überschritten wird:

```
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
```

Aufschlüsselung der Antworten:

Das vorherige Beispiel zeigt, wie eine OpenCypher Antwort mit abschließenden Headern bei der Diagnose von Abfragefehlern helfen kann. Hier sehen wir vier aufeinanderfolgende Teile: (1) anfängliche Header mit einem Status von 200 OK, der anzeigt, dass das Streaming beginnt, (2) teilweise (fehlerhafte) JSON-Ergebnisse, die vor dem Ausfall erfolgreich gestreamt wurden, (3) die angehängte Fehlermeldung mit dem Timeout und (4) nachfolgende Header, die den endgültigen Status (500) und detaillierte Fehlerinformationen enthalten. TimeLimitExceededException

Verwendung der AWS SDK zum Ausführen von OpenCypher-Abfragen

Mit dem AWS SDK können Sie OpenCypher-Abfragen für Ihr Neptune-Diagramm mit einer Programmiersprache Ihrer Wahl ausführen. Das Neptune Data API SDK (Dienstname`neptunedata`) bietet die ExecuteOpenCypherQueryAktion zum Senden von OpenCypher-Abfragen.

Sie müssen diese Beispiele von einer Amazon EC2 EC2-Instance in derselben Virtual Private Cloud (VPC) wie Ihr Neptune-DB-Cluster oder von einem Standort aus ausführen, der über eine Netzwerkverbindung zu Ihrem Cluster-Endpunkt verfügt.

Direkte Links zur API-Referenzdokumentation für den `neptunedata` Service in jeder SDK-Sprache finden Sie unten:

Programmiersprache	Neptunedata API-Referenz
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/

Programmiersprache	Neptunedata API-Referenz
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
CLI	https://docs.aws.amazon.com/cli/latest/reference/neptunedata/

openCypher AWS SDK-Beispiele

Die folgenden Beispiele zeigen, wie Sie einen `neptunedata` Client einrichten, eine OpenCypher-Abfrage ausführen und die Ergebnisse drucken. Ersetzen Sie *`YOUR_NEPTUNE_HOST`* und *`YOUR_NEPTUNE_PORT`* durch den Endpunkt und Port Ihres Neptune-DB-Clusters.

⚠ Client-side Timeout und Konfiguration erneut versuchen

Das SDK-Client-Timeout steuert, wie lange der Client auf eine Antwort wartet. Es steuert nicht, wie lange die Abfrage auf dem Server ausgeführt wird. Wenn der Client das Timeout überschreitet, bevor der Server fertig ist, wird die Abfrage möglicherweise weiterhin auf Neptune ausgeführt, während der Client keine Möglichkeit hat, die Ergebnisse abzurufen. Wir empfehlen, das clientseitige Lese-Timeout auf `0` (kein Timeout) oder auf einen Wert zu setzen, der mindestens ein paar Sekunden länger ist als die serverseitige

neptune_query_timeout-Einstellung auf Ihrem Neptune-DB-Cluster. Auf diese Weise kann Neptune kontrollieren, wann das Timeout bei Abfragen abläuft.
Wir empfehlen außerdem, die maximale Anzahl an Wiederholungsversuchen auf 1 (keine Wiederholungsversuche) festzulegen. Wenn das SDK eine Abfrage wiederholt, die noch auf dem Server ausgeführt wird, kann dies zu doppelten Vorgängen führen. Dies ist besonders wichtig bei Mutationsabfragen, bei denen ein erneuter Versuch zu unbeabsichtigten doppelten Schreibvorgängen führen kann.

Python

1. Folgen Sie den Installationsanweisungen, um Boto3 zu installieren.
2. Erstellen Sie eine Datei mit dem Namen `openCypherExample.py` und fügen Sie den folgenden Code ein:

```
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. Führen Sie das Beispiel aus: `python openCypherExample.py`

Java

1. Folgen Sie den Installationsanweisungen, um das AWS SDK for Java einzurichten.
2. Verwenden Sie den folgenden Code`NeptunedataClient`, um eine OpenCypher-Abfrage einzurichten, auszuführen und das Ergebnis auszudrucken:

```
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. Folgen Sie den Installationsanweisungen, um das SDK für einzurichten. AWS JavaScript Installieren Sie das Neptunedata-Client-Paket:. `npm install @aws-sdk/client-neptunedata`
2. Erstellen Sie eine Datei mit dem Namen `openCypherExample.js` und fügen Sie den folgenden Code ein:

```
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. Führen Sie das Beispiel aus: `node openCypherExample.js`

Verwenden des Bolt-Protokolls für openCypher-Abfragen an Neptune

Bolt ist ein aussagekräftiges client/server Protokoll, das ursprünglich von Neo4j entwickelt und unter der Creative Commons 3.0-Lizenz lizenziert wurde. Attribution-ShareAlike Es ist Client-gesteuert, was bedeutet, dass der Client stets den Nachrichtenaustausch initiiert.

Um über die Bolt-Treiber von Neo4j eine Verbindung zu Neptune herzustellen, ersetzen Sie einfach mittels des `bolt`-URI-Schemas URL und Portnummer durch Ihre Cluster-Endpunkte. Wenn Sie eine einzelne Neptune-Instance ausführen, verwenden Sie den Endpunkt read_write. Wenn mehrere Instances ausgeführt werden, werden zwei Treiber empfohlen, ein Treiber für den Writer und ein Treiber für alle Read Replicas. Wenn es nur die beiden Standardendpunkte gibt, reichen ein read_write- und ein read_only-Treiber aus. Wenn es jedoch auch benutzerdefinierte Endpunkte gibt, sollten Sie für jeden Endpunkt eine Treiber-Instance erstellen.

> **Note**
>
> Obwohl die Bolt-Spezifikation besagt, dass Bolt eine Verbindung entweder über TCP oder herstellen kann WebSockets, unterstützt Neptune nur TCP-Verbindungen für Bolt.

Neptune ermöglicht bis zu 1000 gleichzeitige Bolt-Verbindungen auf allen Instance-Größen außer t3.medium und t4g.medium. Auf den Instances t3.medium und t4g.medium sind nur 512 Verbindungen zulässig.

Beispiele für openCypher-Abfragen in verschiedenen Sprachen, die Bolt-Treiber verwenden, finden Sie in der Neo4j-Dokumentation Drivers & Language Guides.

> **Important**
>
> Die Neo4j Bolt-Treiber für Python JavaScript, .NET und Golang unterstützten zunächst nicht die automatische Verlängerung von AWS Signature v4-Authentifizierungstoken. Das bedeutet, dass sich der Treiber nach Ablauf der Signatur (häufig innerhalb von 5 Minuten) nicht authentifizieren konnte und nachfolgende Anforderungen fehlschlugen. Die folgenden Python- JavaScript, .NET- und Go-Beispiele waren alle von diesem Problem betroffen. Weitere Informationen finden Sie unter Neo4j Python-Treiberproblem #834, Neo4j .NET-Problem #664, JavaScriptNeo4j-Treiberproblem #993 und Neo4j GoLang-Treiberproblem #429.
> Ab Treiberversion 5.8.0 wurde eine neue Vorschau-API zur erneuten Authentifizierung für den Go-Treiber veröffentlicht (siehe v5.8.0 – Feedback zur erneuten Authentifizierung gewünscht).

Herstellen einer Verbindung mit Neptune über Bolt mit Java

Sie können einen Treiber für jede gewünschte Version aus dem Maven-MVN-Repository herunterladen oder diese Abhängigkeit zu Ihrem Projekt hinzufügen:

```
<dependency>
  <groupId>org.neo4j.driver</groupId>
  <artifactId>neo4j-java-driver</artifactId>
  <version>4.3.3</version>
</dependency>
```

Um dann mit einem dieser Bolt-Treiber eine Verbindung zu Neptune in Java herzustellen, erstellen Sie eine Treiberinstanz für die primary/writer Instanz in Ihrem Cluster mit Code wie dem folgenden:

```
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());
```

Wenn Sie über eine oder mehrere Leserreplikate verfügen, können Sie auf ähnliche Weise eine Treiber-Instance für sie erstellen, indem Sie den folgenden Code verwenden:

```
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());
```

Oder mit einem Timeout:

```
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());
```

Wenn es benutzerdefinierte Endpunkte gibt, kann es sich auch lohnen, für jeden dieser Endpunkte eine Treiber-Instance zu erstellen.

Beispiel für eine Python-openCypher-Abfrage über Bolt

So erstellen Sie eine openCypher-Abfrage in Python über 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)
```

Beachten Sie, dass die `auth`-Parameter ignoriert werden.

Beispiel für eine .NET-openCypher-Abfrage über Bolt

Um eine OpenCypher-Abfrage in.NET mit Bolt zu erstellen, müssen Sie zunächst den Neo4j-Treiber mit installieren. NuHet Um synchrone Aufrufe auszuführen, verwenden Sie die `.Simple`-Version wie folgt:

```
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();
      }
    }
  }
}
```

Ein Beispiel für eine Java-openCypher-Abfrage über Bolt mit IAM-Authentifizierung

Der folgende Java-Code zeigt, wie openCypher-Abfragen in Java über Bolt mit IAM-Authentifizierung ausgeführt werden. Der JavaDoc Kommentar beschreibt seine Verwendung. Sobald eine Treiber-Instance verfügbar ist, können Sie diese verwenden, um mehrere authentifizierte Anfragen zu senden.

```
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);
  }
}
```

Ein Beispiel für eine Python-openCypher-Abfrage über Bolt mit IAM-Authentifizierung

Mittels der folgenden Python-Klasse können Sie openCypher-Abfragen in Python über Bolt mit IAM-Authentifizierung ausführen:

```
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)
```

Sie erstellen mittels dieser Klasse einen Treiber wie folgt:

```
  authToken = NeptuneAuthToken(creds, REGION, URL)
  driver = GraphDatabase.driver(URL, auth=authToken, encrypted=True)
```

Ein Node.js Beispiel mit IAM-Authentifizierung und Bolt

Der folgende Node.js Code verwendet das AWS SDK für JavaScript Version 3 und die ES6-Syntax, um einen Treiber zu erstellen, der Anfragen authentifiziert:

```
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();
}
```

Ein Beispiel für eine .NET-openCypher-Abfrage über Bolt mit IAM-Authentifizierung

Um die IAM-Authentifizierung in .NET zu aktivieren, müssen Sie eine Anforderung bei Herstellung der Verbindung signieren. Das folgende Beispiel zeigt, wie Sie einen `NeptuneAuthToken`-Helper erstellen, um ein Authentifizierungstoken zu generieren:

```
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);
    }
  }
}
```

Hier wird beschrieben, wie Sie eine openCypher-Abfrage in .NET über Bolt mit IAM-Authentifizierung erstellen. Das folgende Beispiel verwendet den `NeptuneAuthToken`-Helper:

```
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);
      }
    }
  }
}
```

Sie können dieses Beispiel starten, indem der folgende Code auf .NET 6 oder .NET 7 mit den folgenden Paketen ausgeführt wird:

- **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();
    }
  }
}
```

Ein Beispiel für eine Golang-openCypher-Abfrage über Bolt mit IAM-Authentifizierung

Das folgende Beispiel zeigt, wie OpenCypher-Abfragen in Go mithilfe des Bolt-Protokolls mit IAM-Authentifizierung durchgeführt werden. Es verwendet das AWS SDK for Go v2 für die Sigv4-Signierung und den Neo4j Go-Treiber v5 mit einer `AuthTokenManager` Struktur, die die Token-Manager-Schnittstelle (`github.com/neo4j/neo4j-go-driver/v5/neo4j/auth.TokenManager`) des Neo4j Go-Treibers implementiert, um Anmeldeinformationen automatisch zu aktualisieren, bevor sie ablaufen.

Erstellen Sie zunächst eine, die Token generiert. `AuthTokenManager` SigV4-signed Speichern Sie das als`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
}
```

Verwenden Sie dann den Token-Manager, um einen Treiber zu erstellen und einen Knoten anhand der ID zu finden. Beachten Sie die Verwendung einer parametrisierten Abfrage (`$nodeId`) anstelle einer Zeichenketteninterpolation:

```
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**
>
> Das in diesem Beispiel verwendete **`auth.TokenManager`** Interface (**`github.com/neo4j/neo4j-go-driver/v5/neo4j/auth`**) wurde im Neo4j Go-Treiber v5.14.0 allgemein verfügbar. Diese Schnittstelle ermöglicht die automatische Aktualisierung der Anmeldeinformationen, was für die Neptune IAM-Authentifizierung erforderlich ist, da Sigv4-Signaturen nur für einen kurzen Zeitraum gültig sind und neu generiert werden müssen, wenn der Treiber neue Verbindungen herstellt.

Dieses Beispiel wurde mit den folgenden Go-Modulen validiert:

```
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
  )
```

Bolt-Verbindungsverhalten in Neptune

Dies sind einige Punkte, die Sie bei Neptune-Bolt-Verbindungen beachten sollten:

- Da Bolt-Verbindungen auf der TCP-Ebene erstellt werden, können Sie diesen (anders als bei HTTP-Endpunkten) keinen Application Load Balancer voranstellen.
- Der Port, den Neptune für Bolt-Verbindungen verwendet, ist der Port Ihres DB-Clusters.
- Basierend auf der übergebenen Bolt-Präambel wählt der Neptune-Server die am besten geeignete Bolt-Version aus (1, 2, 3 oder 4.0).
- Die maximale Anzahl der Verbindungen zum Neptune-Server, die gleichzeitig auf einem Client geöffnet sein können, beträgt 1 000.
- Wenn der Client eine Verbindung nach Abschluss einer Abfrage nicht schließt, kann sie für die nächste Abfrage verwendet werden.

- Wenn eine Verbindung jedoch 20 Minuten inaktiv ist, schließt der Server sie automatisch.
- Wenn die IAM-Authentifizierung nicht aktiviert ist, können Sie `AuthTokens.none()` statt eines Platzhalter-Benutzernamens und Passwort verwenden. Beispiel für Java:

```
GraphDatabase.driver("bolt://(your cluster endpoint URL):(your cluster port)",
 AuthTokens.none(),

 Config.builder().withEncryption().withTrustStrategy(TrustStrategy.trustSystemCertificates())
```

- Wenn die IAM-Authentifizierung aktiviert ist, wird eine Bolt-Verbindung nach Ablauf von 10 Tagen nach der Herstellung stets für einige Minuten unterbrochen, wenn sie nicht bereits geschlossen wurde.
- Wenn ein Client eine Abfrage zur Ausführung über eine Verbindung sendet, ohne die Ergebnisse einer vorherigen Abfrage verwendet zu haben, wird die neue Abfrage verworfen. Um stattdessen die vorherigen Ergebnisse zu verwerfen, muss der Client eine Rücksetzungsmeldung über die Verbindung senden.
- Es kann jeweils nur eine Transaktion gleichzeitig für eine bestimmte Verbindung erstellt werden.
- Wenn während einer Transaktion eine Ausnahme auftritt, setzt der Neptune-Server die Transaktion zurück und schließt die Verbindung. In diesem Fall erstellt der Treiber eine neue Verbindung für die nächste Abfrage.
- Beachten Sie, dass Sitzungen nicht Thread-sicher sind. Mehrere parallele Operationen müssen mehrere getrennte Sitzungen verwenden.

Beispiele für parametrisierte openCypher-Abfragen

Neptune unterstützt parametrisierte openCypher-Abfragen. So können Sie dieselbe Abfragestruktur mehrmals mit unterschiedlichen Argumenten verwenden. Da sich die Abfragestruktur nicht ändert, kann Neptune den abstrakten Syntaxbaum (AST) zwischenspeichern, statt ihn mehrmals analysieren zu müssen.

Beispiel für eine parametrisierte openCypher-Abfrage, die den HTTPS-Endpunkt verwendet

Beispiel für die Verwendung einer parametrisierten Abfrage über den Neptune-openCypher-HTTPS-Endpunkt Die Abfrage ist:

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

```
RETURN n
```

Die Parameter sind wie folgt definiert:

```
parameters={"name": "john", "age": 20}
```

Sie können die parametrisierte Abfrage wie folgt einreichen:

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}'
```

Weitere Informationen finden Sie unter execute-open-cypher-query in der Befehlsreferenz. 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'])
```

AWS SDK-Beispiele in anderen Sprachen finden Sie unter. AWS 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**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

curl

Verwenden von POST:

```
curl https://your-neptune-endpoint:port/openCypher \
  -d "query=MATCH (n {name: \$name, age: \$age}) RETURN n" \
  -d "parameters={\"name\": \"john\", \"age\": 20}"
```

Verwenden von 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"
```

Verwenden von 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"
```

Beispiele für parametrisierte openCypher-Abfragen über Bolt

Dies ist ein Python-Beispiel für eine parametrisierte openCypher-Abfrage über das Bolt-Protokoll:

```
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()
```

Dies ist ein Java-Beispiel für eine parametrisierte openCypher-Abfrage über das Bolt-Protokoll:

```
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);
```

openCypher-Datenmodell

Die Neptune openCypher-Engine basiert auf demselben Eigenschaftsdiagrammmodell wie Gremlin. Insbesondere gilt:

- Jeder Knoten hat eine oder mehrere Bezeichnungen. Wenn Sie einen Knoten ohne Bezeichnungen einfügen, wird die Standardbezeichnung `vertex` angefügt. Wenn Sie versuchen, alle Bezeichnungen eines Knotens zu löschen, wird ein Fehler ausgegeben.
- Eine Beziehung ist eine Entität mit genau einem Beziehungstyp, die eine unidirektionale Verbindung zwischen zwei Knoten herstellt (d. h. von einem Knoten zum anderen Knoten).
- Knoten und Beziehungen können Eigenschaften haben, müssen jedoch nicht. Neptune unterstützt Knoten und Beziehungen ohne Eigenschaften.
- Neptune unterstützt keine MetaProperties, die auch nicht in der openCypher-Spezifikation enthalten sind.
- Eigenschaften im Diagramm können mehrere Werte haben, wenn sie mit Gremlin erstellt wurden. Das bedeutet, dass eine Knoten- oder Beziehungseigenschaft verschiedene Werte haben kann,

nicht nur einen. Neptune hat die openCypher-Semantik erweitert, um mehrwertige Eigenschaften korrekt zu behandeln.

Die unterstützten Datentypen sind in openCypher-Datenformat dokumentiert. Derzeit empfehlen wir jedoch nicht, `Array`-Eigenschaftswerte in ein openCypher-Diagramm einzufügen. Obwohl es möglich ist, einen Array-Eigenschaftswert über den Massen-Loader einzufügen, behandelt ihn die aktuelle Neptune-openCypher-Version als einen Satz von Eigenschaften mit mehreren Werten und nicht als einen einzelnen Listenwert.

Im Folgenden finden Sie eine Liste der Datentypen, die in dieser Version unterstützt werden:

- `Bool`
- `Byte`
- `Short`
- `Int`
- `Long`
- `Float` (Umfasst Plus- und Minus-Infinity und NaN, jedoch nicht INF)
- `Double` (Umfasst Plus- und Minus-Infinity und NaN, jedoch nicht INF)
- `DateTime`
- `String`

Die OpenCypher-Erklärungsfunktion

Das openCypher-Feature `explain` ist ein Self-Service-Tool in Amazon Neptune, das Ihnen hilft, den Ausführungsansatz der Neptune-Engine besser zu verstehen. Um explain aufzurufen, übergeben Sie einen Parameter an eine openCypher-HTTPS-Anforderung mit`explain=`*`mode`*, wobei der Wert für *`mode`* einer der folgenden sein kann:

- **`static`** – Im Modus `static` gibt `explain` lediglich die statische Struktur des Abfrageplans aus. Die Abfrage selbst wird nicht ausgeführt.
- **`dynamic`** – Im Modus `dynamic` führt `explain` auch die Abfrage aus und fügt dynamische Aspekte des Abfrageplans ein. Dies kann die Anzahl der durch die Operatoren fließenden Zwischenbindungen, das Verhältnis zwischen eingehenden und ausgehenden Bindungen sowie die von den Operatoren insgesamt in Anspruch genommene Zeit umfassen.

- **`details`** – Im Modus `details` gibt `explain` die im dynamischen Modus angezeigten Informationen sowie zusätzliche Details aus, wie die tatsächliche SPARQL-Abfragezeichenfolge und die geschätzte Bereichsanzahl für das Muster, das einem Join-Operator zugrunde liegt.

Verwenden `POST` Sie zum Beispiel den Modus with: `dynamic`

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
```

Weitere Informationen finden Sie unter execute-open-cypher-explain-query in der Befehlsreferenz. 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'))
```

AWS SDK-Beispiele in anderen Sprachen finden Sie unter. AWS 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**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

curl

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

Einschränkungen für OpenCypher werden in Neptune erklärt

Für die aktuelle Version von openCypher explain gelten folgende Einschränkungen:

- Explain-Pläne sind derzeit nur für Abfragen verfügbar, die schreibgeschützte Operationen ausführen. Abfragen, die eine Mutation wie `CREATE`, `DELETE`, `MERGE`, `SET` usw. ausführen, werden nicht unterstützt.
- Operatoren und Ausgabe für bestimmte Pläne unterliegen Änderungen in zukünftigen Versionen.

DFE-Operatoren in OpenCypher erklären die Ausgabe

Um die Informationen nutzen zu können, die das openCypher-Feature `explain` bereitstellt, müssen Sie wissen, wie die DFE-Abfrage-Engine funktioniert. (DFE ist die Engine, die Neptune für die Verarbeitung von openCypher-Abfragen verwendet.)

Die DFE-Engine übersetzt jede Abfrage in eine Operator-Pipeline. Ab dem ersten Operator werden durch diese Operator-Pipeline Zwischenlösungen von einem Operator zum nächsten geleitet. Jede Zeile in der explain-Tabelle stellt ein Ergebnis dar, bis zur Bewertung.

Folgende Operatoren können in einem DFE-Abfrageplan vorkommen:

DFEApply – Führt die im Argumentabschnitt angegebene Funktion für den in der angegebenen Variablen gespeicherten Wert aus.

DFEBindRelation— Verbindet Variablen mit den angegebenen Namen

DFEChunkLocalSubQuery— Dies ist ein nicht blockierender Vorgang, der als Wrapper für ausgeführte Unterabfragen fungiert.

DFEDistinctColumn— Gibt die eindeutige Teilmenge der Eingabewerte auf der Grundlage der angegebenen Variablen zurück.

DFEDistinctRelation— Gibt die eindeutige Teilmenge der Eingabelösungen auf der Grundlage der angegebenen Variablen zurück.

DFEDrain – Wird am Ende einer Unterabfrage angezeigt und dient als Abschlussschritt für diese Unterabfrage. Die Anzahl der Lösungen wird als `Units In` aufgezeichnet. `Units Out` ist immer null.

DFEForwardValue— Kopiert alle Eingabeblöcke direkt als Ausgabeblöcke, die an den nachgeschalteten Operator übergeben werden.

DFEGroupByHashIndex— Führt auf der Grundlage eines zuvor berechneten Hash-Index (unter Verwendung der Operation) eine Gruppierungsoperation für die Eingabelösungen durch. `DFEHashIndexBuild` Als Ausgabe wird die angegebene Eingabe um eine Spalte erweitert, die einen Gruppenschlüssel für jede Eingabelösung enthält.

DFEHashIndexBuild— Erzeugt als Nebeneffekt einen Hash-Index über einer Reihe von Variablen. Dieser Hash-Index wird normalerweise in späteren Vorgängen wiederverwendet. Weitere Informationen dazu, wo dieser Hash-Index verwendet werden kann, finden Sie unter `DFEHashIndexJoin` und `DFEGroupByHashIndex`.

DFEHashIndexJoin— Führt eine Verknüpfung der eingehenden Lösungen anhand eines zuvor erstellten Hash-Index durch. Weitere Informationen dazu, wo dieser Hash-Index erstellt werden kann, finden Sie unter `DFEHashIndexBuild`.

DFEJoinExists— Nimmt eine linke und rechte Eingabebeziehung auf und behält Werte aus der linken Relation bei, die einen entsprechenden Wert in der rechten Beziehung haben, wie durch die angegebenen Join-Variablen definiert.

– Dies ist eine nicht blockierende Operation, die als Wrapper für eine Unterabfrage dient und ihre wiederholte Ausführung ermöglicht, um sie in Schleifen zu verwenden.

DFEMergeChunks— Dabei handelt es sich um eine Blockierungsoperation, bei der Blöcke aus dem vorgelagerten Operator zu einem einzigen Lösungsblock zusammengefasst werden, der dann an den nachgeschalteten Operator weitergegeben wird (Umkehrung von). `DFESplitChunks`

DFEMinus – Nimmt eine linke und eine rechte Eingabebeziehung und behält Werte aus der linken Beziehung bei, die keinen entsprechenden Wert in der rechten Beziehung haben, wie durch die angegebenen Join-Variablen definiert. Wenn sich die Variablen in beiden Beziehungen nicht überschneiden, gibt dieser Operator einfach die linke Eingabebeziehung zurück.

DFENotExists— Nimmt eine Eingabebeziehung für die linke und rechte Hand entgegen und behält Werte aus der linken Beziehung bei, für die in der rechten Beziehung, wie sie durch die angegebenen Join-Variablen definiert ist, kein entsprechender Wert vorhanden ist. Wenn sich die Variablen in beiden Beziehungen nicht überschneiden, gibt dieser Operator eine leere Beziehung zurück.

DFEOptionalJoin— Führt einen linken äußeren Join durch (auch OPTIONALER Join genannt): Lösungen von der linken Seite, die mindestens einen Join-Partner auf der rechten Seite haben, werden zusammengefügt, und Lösungen von der linken Seite ohne Join-Partner auf der rechten Seite werden unverändert weitergeleitet. Dies ist eine blockierende Operation.

DFEPipelineJoin— Verbindet die Eingabe anhand des durch das Argument definierten Tupelmusters. `pattern`

DFEPipelineRangeCount— Zählt die Anzahl der Lösungen, die einem bestimmten Muster entsprechen, und gibt eine einzige einjährige Lösung zurück, die den Zählwert enthält.

DFEPipelineScan— Durchsucht die Datenbank nach dem angegebenen `pattern` Argument, mit oder ohne einen bestimmten Filter für Spalte (n).

DFEProject – Nimmt mehrere Eingabespalten an und projiziert nur die gewünschten Spalten.

DFEReduce – Führt die angegebene Aggregationsfunktion für angegebene Variablen aus.

DFERelationalJoin— Verbindet die Eingabe des vorherigen Operators auf der Grundlage der angegebenen Musterschlüssel mithilfe eines Merge-Joins. Dies ist eine blockierende Operation.

DFERouteChunks— Entnimmt Eingabe-Chunks von seiner einzelnen Eingangskante und leitet diese Chunks entlang seiner mehreren ausgehenden Kanten weiter.

DFESelectRows— Dieser Operator nimmt selektiv Zeilen aus seinen linken Eingaberelationslösungen und leitet sie an seinen nachgeschalteten Operator weiter. Die ausgewählten Zeilen basieren auf den Zeilenbezeichnern, die in der rechten Eingabebeziehung des Operators angegeben wurden.

DFESerialize – Serialisiert die Endergebnisse einer Abfrage in eine JSON-Zeichenfolgenserialisierung, wobei jede Eingabelösung dem entsprechenden Variablennamen zugeordnet wird. Bei Knoten- und Kantenergebnissen werden diese Ergebnisse in einer Zuordnung von Entitätseigenschaften und Metadaten serialisiert.

DFESort – Nimmt eine Eingabebeziehung und erzeugt eine sortierte Beziehung auf der Grundlage des angegebenen Sortierschlüssels.

DFESplitByGroup— Teilt jeden einzelnen Eingabeblock von einer eingehenden Kante in kleinere Ausgangsblöcke auf, die Zeilengruppen entsprechen, die durch Zeilen-IDs des entsprechenden Eingabeblocks von der anderen Eingangskante identifiziert werden.

DFESplitChunks— Teilt jeden einzelnen Eingabeblock in kleinere Ausgangsblöcke auf (Umkehrung von). `DFEMergeChunks`

DFEStreamingHashIndexBuild— Streaming-Version von. `DFEHashIndexBuild`

DFEStreamingGroupByHashIndex— Streaming-Version von`DFEGroupByHashIndex`.

DFESubquery – Dieser Operator erscheint am Anfang aller Pläne und kapselt die Teile des Plans, die in der DFE-Engine ausgeführt werden. Dies ist der gesamte Plan für openCypher.

DFESymmetricHashJoin— Verbindet die Eingabe des vorherigen Operators auf der Grundlage der angegebenen Musterschlüssel mithilfe eines Hash-Joins. Dies ist eine nicht blockierende Operation.

DFESync – Dieser Operator ist ein Synchronisationsoperator, der nicht blockierende Pläne unterstützt. Er nimmt Lösungen von zwei eingehenden Kanten entgegen und leitet diese Lösungen an die entsprechenden Downstream-Kanten weiter. Zu Synchronisationszwecken können die Eingaben entlang eines dieser Kanten intern gepuffert werden.

DFETee – Ein Verzweigungsoperator, der denselben Satz von Lösungen an mehrere Operatoren sendet.

DFETermResolution— Führt eine Lokalisierungs- oder Globalisierungsoperation für seine Eingaben durch, was zu Spalten mit jeweils lokalisierten oder globalisierten Bezeichnern führt.

– Erweitert Wertelisten aus einer Eingabespalte als einzelne Elemente in die Ausgabespalte.

DFEUnion – Nimmt zwei oder mehr Eingabebeziehungen und erzeugt eine Vereinigung dieser Beziehungen unter Verwendung des gewünschten Ausgabeschemas.

SolutionInjection— Erscheint in der Explain-Ausgabe vor allem anderen, mit dem Wert 1 in der Spalte Units Out. Dies dient jedoch als no-op-Anweisung und fügt keine Lösungen in die DFE-Engine ein.

TermResolution— Erscheint am Ende von Plänen und übersetzt Objekte aus der Neptune-Engine in OpenCypher-Objekte.

Spalten `in` OpenCypher erklären die Ausgabe

Die Abfrageplaninformationen, die Neptune als openCypher-Ausgabe für explain generiert, enthalten Tabellen mit einem einzelnen Operator pro Zeile. Diese Tabelle hat die folgenden Spalten:

ID – Die numerische ID dieses Operators im Plan.

Aus #1 (und Aus #2) – Die ID des Operators, der diesem Operator nachgeschaltet ist. Es kann höchstens zwei nachgeschaltete Operatoren geben.

Name – Der Name dieses Operators.

Argumente – Alle relevanten Details für den Operator. Dies umfasst Eingabeschema, Ausgabeschema, Muster (für `PipelineScan` und `PipelineJoin`) usw.

Modus – Eine Bezeichnung, die das grundlegende Operatorverhalten beschreibt. Diese Spalte ist meistens leer (`-`). Eine Ausnahme ist `TermResolution`. Hier kann der Modus `id2value_opencypher` sein, um die Auflösung einer ID zu einem openCypher-Wert anzugeben.

Einheiten ein – Die Anzahl der Lösungen, die als Eingabe an diesen Operator übergeben wurden. Operatoren ohne vorgeschaltete Operatoren (wie `DFEPipelineScan` und `SolutionInjections`) und eine `DFESubquery` ohne eingefügten statischen Wert hätten den Wert null.

Einheiten aus – Die Anzahl der Lösungen, die als Ausgabe für diesen Operator erzeugt wurden. `DFEDrain` ist ein Sonderfall, da die Anzahl der Lösungen in `Units In` aufgezeichnet wird und `Units Out` stets null ist.

Verhältnis – Das Verhältnis von `Units Out` zu `Units In`.

Zeit (ms) – Die von diesem Operator verbrauchte CPU-Zeit in Millisekunden.

Ein einfaches Beispiel für openCypher-Ausgabe für explain

Dies ist ein einfaches Beispiel für eine openCypher-Ausgabe für `explain`. Bei der Abfrage handelt es sich um eine Suche mit einem einzelnen Knoten im Flugrouten-Datensatz nach einem Knoten mit

dem FlughafencodeATL, der `explain` unter Verwendung des `details` Modus im Standard-ASCII-Ausgabeformat aufgerufen wird.

So rufen Sie diese Abfrage auf`explain`:

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
```

Weitere Informationen finden Sie unter execute-open-cypher-explain-query in der Befehlsreferenz. 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'))
```

AWS SDK-Beispiele in anderen Sprachen finden Sie unter. AWS 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**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

curl

```
curl https://your-neptune-endpoint:port/openCypher \
  -d "query=MATCH (n {code: 'ATL'}) RETURN n" \
  -d "explain=details"
```

Die `explain` Ausgabe:

```
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      #
#########################################################################################
```

`SolutionInjection` erscheint auf der obersten Ebene vor allem anderen mit 1 Einheit aus. Beachten Sie, dass keine Lösungen eingefügt werden. Sie sehen, dass der nächste Operator (`DFESubquery`) 0 Einheiten in enthält.

Nach `SolutionInjection` auf der obersten Ebene gibt es die Operatoren `DFESubquery` und `TermResolution`. `DFESubquery`kapselt die Teile des Abfrageausführungsplans, der per Push an die DFE-Engine übertragen wird. (Bei openCypher-Abfragen wird der gesamte Abfrageplan von der DFE ausgeführt.) Alle Operatoren im Abfrageplan sind innerhalb von `subQuery1` verschachtelt, worauf von `DFESubquery` verwiesen wird. Die einzige Ausnahme ist `TermResolution`, das interne IDs zu vollständig serialisierten openCypher-Objekten materialisiert.

Alle Namen aller Operatoren, die per Push an die DFE-Engine übertragen werden, beginnen mit dem Präfix `DFE`. Wie oben erwähnt, wird der gesamte openCypher-Abfrageplan von der DFE ausgeführt. Daher beginnen alle Operatoren außer dem letzten Operator `TermResolution` mit `DFE`.

In `subQuery1` kann es null oder mehr Operatoren `DFEChunkLocalSubQuery` oder `DFELoopSubQuery` geben, die einen Teil des per Push übertragenen Ausführungsplans kapseln, der in einem speicherbegrenzten Mechanismus ausgeführt wird. `DFEChunkLocalSubQuery` enthält hier eine einzelne `SolutionInjection`, die als Eingabe für die Unterabfrage verwendet wird. Um die Tabelle für diese Unterabfrage in der Ausgabe zu finden, suchen Sie nach der

`subQuery=graph URI`, die in Spalte `Arguments` für den Operator `DFEChunkLocalSubQuery` oder `DFELoopSubQuery` angegeben ist.

In `subQuery1` scannt `DFEPipelineScan` mit `ID` 0 die Datenbank nach einem angegebenen `pattern`. Das Muster scannt nach einer Entität mit der Eigenschaft `code` (gespeichert als Variable `?n_code2`) für alle Bezeichnungen. (Sie könnten nach einer angegebenen Bezeichnung filtern, indem Sie `airport` an `n:airport` anfügen). Das Argument `inlineFilters` zeigt, dass die Filterung für die Eigenschaft `code` gleich `ATL` ist.

Als Nächstes verknüpft der Operator `DFEChunkLocalSubQuery` die Zwischenergebnisse einer Unterabfrage, die `DFEPipelineJoin` enthält. Dies stellt sicher, dass `?n` tatsächlich ein Knoten ist, da der vorherige `DFEPipelineScan` auf jede Entität mit der Eigenschaft `code` scannt.

Beispiel für eine **`Erklärungsausgabe`** für eine Beziehungssuche mit einem Limit

Diese Abfrage sucht nach Beziehungen zwischen zwei anonymen Knoten mit Typ `route` und gibt höchstens 10 zurück. Auch hier ist der Modus `explain details` und das Ausgabeformat ist das Standard-ASCII-Format.

Hier scannt `DFEPipelineScan` auf Kanten, die am anonymen Knoten `?anon_node7` starten und am anderen anonymen Knoten `?anon_node21` enden, wobei der Beziehungstyp als `?p_type1` gespeichert ist. Es gibt einen Filter für `?p_type1` (`el://route`) (wobei `el` für Kantenbezeichnung steht), was `[p:route]` in der Abfragezeichenfolge entspricht.

`DFEDrain` erfasst die Ausgabelösung mit der Einschränkung 10, wie in Spalte `Arguments` gezeigt. `DFEDrain` wird beendet, sobald die Einschränkung erreicht wird oder alle Lösungen erzeugt wurden, je nachdem, was zuerst eintritt.

Um `explain` für diese Abfrage aufzurufen:

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
```

Weitere Informationen finden Sie unter execute-open-cypher-explain-query in der Befehlsreferenz. 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'))
```

AWS SDK-Beispiele in anderen Sprachen finden Sie unter. AWS 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**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

curl

```
curl https://your-neptune-endpoint:port/openCypher \
  -d "query=MATCH ()-[p:route]->() RETURN p LIMIT 10" \
  -d "explain=details"
```

Die `explain` Ausgabe:

```
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      #
################################################################################################
```

Beispiel für eine **`Erklärungsausgabe`** für eine Wertausdrucksfunktion

Die Funktion ist:

```
MATCH (a) RETURN DISTINCT labels(a)
```

In der folgenden `explain`-Ausgabe scannt `DFEPipelineScan` (ID 0) auf alle Knotenbezeichnungen. Dies entspricht `MATCH (a`.

`DFEChunkLocalSubquery` (ID 1) aggregiert die Bezeichnung `?a` für jedes `?a`. Dies entspricht `labels(a)`. Sie können dies über `DFEApply` und `DFEReduce` anzeigen.

`BindRelation` (ID 2) wird verwendet, um den generischen `?__gen_labels0fa2` der Spalte in `?labels(a)` umzubenennen.

`DFEDistinctRelation` (ID 4) ruft lediglich eindeutige Bezeichnungen ab (mehrere :airport-Knoten würden zu duplizierten Bezeichnungen labels(a): ["airport"] führen.) Dies entspricht `DISTINCT labels(a)`.

Um `explain` für diese Abfrage aufzurufen:

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
```

Weitere Informationen finden Sie unter execute-open-cypher-explain-query in der Befehlsreferenz. 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'))
```

AWS SDK-Beispiele in anderen Sprachen finden Sie unter. AWS 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**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

curl

```
curl https://your-neptune-endpoint:port/openCypher \
  -d "query=MATCH (a) RETURN DISTINCT labels(a)" \
  -d "explain=details"
```

Die `explain` Ausgabe:

```
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      #
#########################################################################################
```

Beispiel für eine **`Erklärungsausgabe`** für eine mathematische Wertausdrucksfunktion

In diesem Beispiel führt `RETURN abs(-10)` eine einfache Auswertung aus, wobei der absolute Wert einer Konstante angenommen wird, `-10`.

`DFEChunkLocalSubQuery` (ID 1) führt eine Lösungsinjektion für den statischen Wert `-10` aus, der in der Variablen gespeichert ist, `?100`.

`DFEApply` (ID 2) ist der Operator, der die absolute Wertfunktion `abs()` für den statischen Wert ausführt, der in der Variablen `?100` gespeichert ist.

Hier ist die Abfrage und die resultierende `explain` Ausgabe.

Um `explain` für diese Abfrage aufzurufen:

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
```

Weitere Informationen finden Sie unter execute-open-cypher-explain-query in der Befehlsreferenz. 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'))
```

AWS SDK-Beispiele in anderen Sprachen finden Sie unter. AWS 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**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

curl

```
curl https://your-neptune-endpoint:port/openCypher \
  -d "query=RETURN abs(-10)" \
  -d "explain=details"
```

Die `explain` Ausgabe:

```
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      #
#########################################################################################################
```

Beispiel für eine **`Explain-Ausgabe`** für eine VLP-Abfrage (Variable Length Path)

Dies ist ein Beispiel für einen komplexeren Abfrageplan für die Verarbeitung einer Abfrage mit variabler Pfadlänge. Dieses Beispiel zeigt der Übersichtlichkeit halber lediglich einen Teil der `explain`-Ausgabe.

In `subQuery1`, `DFEPipelineScan` (ID 0) und `DFEChunkLocalSubQuery` (ID 1), die die Unterabfrage `...graph_1` einfügt, sind für das Scannen auf einen Knoten mit dem Code `YPO` verantwortlich.

In `subQuery1`, `DFEChunkLocalSubQuery` (ID 2), die die Unterabfrage `...graph_2` einfügt, ist für das Scannen auf einen Knoten mit dem Code `LAX` verantwortlich.

In `subQuery1` fügt `DFEChunkLocalSubQuery` (ID 3) die Unterabfrage `...graph3` ein, die `DFELoopSubQuery` (ID 17) enthält, die wiederum die Unterabfrage `...graph5` einfügt. Diese Operation ist für die Auflösung des Musters `-[*2]->` mit variabler Länge in der Abfragezeichenfolge zwischen zwei Knoten verantwortlich.

Um für diese Abfrage aufzurufen`explain`:

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
```

Weitere Informationen finden Sie unter execute-open-cypher-explain-query in der Befehlsreferenz. 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'))
```

AWS SDK-Beispiele in anderen Sprachen finden Sie unter. AWS 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**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

curl

```
curl https://your-neptune-endpoint:port/openCypher \
  -d "query=MATCH p=(a {code: 'YPO'})-[*2]->(b{code: 'LAX'}) return p" \
  -d "explain=details"
```

Die `explain` Ausgabe:

```
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      #
...
```

Transaktionen in Neptune openCypher

Die openCypher-Implementierung in Amazon Neptune verwendet die von Neptune definierte Transaktionssemantik. Die vom Bolt-Treiber bereitgestellten Isolationsstufen haben jedoch einige spezifische Auswirkungen auf die Bolt-Transaktionssemantik, wie in den folgenden Abschnitten beschrieben.

Read-only Bolt-Transaktionsabfragen

Es gibt verschiedene Möglichkeiten für die Verarbeitung schreibgeschützter Abfragen mit verschiedenen Transaktionsmodellen und Isolationsstufen:

Implizite schreibgeschützte Transaktionsabfragen

Dies ist ein Beispiel für eine implizite schreibgeschützte Transaktion:

```
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();
}
```

Da Read-Replicas nur schreibgeschützte Abfragen akzeptieren, werden alle Abfragen für Read-Replicas unabhängig vom in der Sitzungskonfiguration festgelegten Zugriffsmodus als leseimplizite Transaktionen ausgeführt. Neptune wertet leseimplizite Transaktionen als schreibgeschützte Abfragen unter isolierter SNAPSHOT-Semantik aus.

Bei einem Fehler werden leseimplizite Transaktionen standardmäßig wiederholt.

Schreibgeschützte Autocommit-Transaktionsabfragen

Dies ist ein Beispiel für eine schreibgeschützte Autocommit-Transaktion:

```
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();
}
```

Wenn der Zugriffsmodus in der Sitzungskonfiguration auf READ festgelegt ist, wertet Neptune Autocommit-Transaktionsabfragen als schreibgeschützte Abfragen unter SNAPSHOT-Isolationssemantik aus. Beachten Sie, dass Read-Replicas nur schreibgeschützte Abfragen akzeptieren.

Wenn Sie Autocommit-Abfragen nicht in einer Sitzungskonfiguration übergeben, werden sie standardmäßig mit Mutationsabfragenisolierung verarbeitet. Daher ist es wichtig, sie in einer Sitzungskonfiguration zu übergeben, die den Zugriffsmodus explizit auf READ festlegt.

Bei einem Fehler werden schreibgeschützte Autocommit-Abfragen nicht wiederholt.

Explizite schreibgeschützte Transaktionsabfragen

Dies ist ein Beispiel für eine explizite schreibgeschützte Transaktion:

```
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();
}
```

Wenn der Zugriffsmodus in der Sitzungskonfiguration auf `READ` festgelegt ist, wertet Neptune explizite schreibgeschützte Transaktionen als schreibgeschützte Abfragen unter `SNAPSHOT`-Isolationssemantik aus. Beachten Sie, dass Read-Replicas nur schreibgeschützte Abfragen akzeptieren.

Wenn Sie explizite schreibgeschützte Transaktionen nicht in einer Sitzungskonfiguration übergeben, werden sie standardmäßig mit Mutationsabfragenisolierung verarbeitet. Daher ist es wichtig, sie in einer Sitzungskonfiguration zu übergeben, die den Zugriffsmodus explizit auf `READ` festlegt.

Bei einem Fehler werden explizite schreibgeschützte Abfragen standardmäßig wiederholt.

Bolt-Mutationstransaktionsabfragen

Wie bei schreibgeschützten Abfragen gibt es auch hier verschiedene Möglichkeiten für die Verarbeitung von Mutationsabfragen mit verschiedenen Transaktionsmodellen und Isolationsstufen:

Implizite Mutationstransaktionsabfragen

Dies ist ein Beispiel für eine implizite Mutationstransaktion:

```
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();
}
```

Lesevorgänge als Teil von Mutationsabfragen werden unter `READ COMMITTED`-Isolation mit den üblichen Garantien für Neptune-Mutationstransaktionen ausgeführt.

Unabhängig davon, ob Sie die Transaktion ausdrücklich in einer Sitzungskonfiguration übergeben oder nicht, wird die Transaktion stets als Schreibtransaktion behandelt.

Informationen zu Konflikten finden Sie unter Konfliktlösung mithilfe von Timeouts Lock-Wait.

Autocommit-Mutationstransaktionsabfragen

Autocommit-Mutationsabfragen erben dasselbe Verhalten wie implizite Mutationstransaktionen.

Wenn Sie die Transaktion nicht in einer Sitzungskonfiguration übergeben, wird die Transaktion standardmäßig als Schreibtransaktion behandelt.

Bei einem Fehler werden Autocommit-Mutationsabfragen nicht automatisch wiederholt.

Explizite Mutationstransaktionsabfragen

Dies ist ein Beispiel für eine explizite Mutationstransaktion:

```
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();
}
```

Explizite Mutationsabfragen erben dasselbe Verhalten wie implizite Mutationstransaktionen.

Wenn Sie die Transaktion nicht in einer Sitzungskonfiguration übergeben, wird die Transaktion standardmäßig als Schreibtransaktion behandelt.

Informationen zu Konflikten finden Sie unter Konfliktlösung mithilfe von Timeouts Lock-Wait.

Streaming-Abfrageergebnisse mit dem Bolt-Protokoll

Wenn Sie eine OpenCypher-Abfrage ausführen, die eine große Anzahl von Datensätzen zurückgibt, müssen Sie nicht die gesamte Ergebnismenge auf einmal in den Client-Speicher laden. Neptune unterstützt das Streamen von Ergebnissen über das Bolt-Protokoll, sodass der Treiber Datensätze stapelweise abrufen und sie inkrementell verarbeiten kann, sobald sie ankommen. Dies kann von Vorteil sein, wenn Sie Ergebnisse so verarbeiten möchten, wie sie vom Server zurückgegeben werden, erfordert jedoch die Konfiguration `FetchSize` und Verwendung von verzögerter Iteration, um zu verhindern, dass die gesamte Ergebnismenge in den Speicher geladen wird.

Dies funktioniert über den Bolt v4.0-PULL-Mechanismus. Anstatt alle Datensätze in einer einzigen Nachricht anzufordern, sendet der Treiber eine PULL-Anfrage mit einer Batchgröße. Beispiel: „Gib mir die nächsten 50 Datensätze.“ Neptune sendet diese Anzahl von Datensätzen zusammen mit einer Markierung zurück, die angibt, ob weitere verfügbar sind. Der Treiber fordert dann den nächsten Batch an, wenn der Client bereit ist. Dieser Vorgang wird fortgesetzt, bis die Ergebnismenge vollständig aufgebraucht ist.

Sie steuern die Batchgröße mithilfe der `FetchSize` Einstellung in der Sitzungskonfiguration des Treibers. In Kombination mit verzögerter Iteration (die Datensätze werden einzeln verarbeitet, sobald sie zurückgegeben werden), begrenzt dies die clientseitige Speichernutzung, unabhängig davon, wie groß die Gesamtergebnismenge ist.

Important

`FetchSize`steuert den Datenfluss zwischen Server und Client. Es hat keinen Einfluss darauf, wie Neptune die Abfrage ausführt. Neptune beginnt mit der Ausführung der Abfrage und der Erzeugung von Ergebnissen, sobald es die RUN-Meldung empfängt. Die Ergebnisse werden serverseitig zwischengespeichert, bis der Client sie mit einer PULL-Nachricht anfordert. Wenn der Client innerhalb des konfigurierten Abfrage-Timeouts keine Ergebnisse verarbeitet, beendet Neptune die Abfrage, verwirft gepufferte Ergebnisse und gibt serverseitige Ressourcen frei. Es gibt kein separates serverseitiges Pufferlimit. Die Pufferung wird fortgesetzt, bis der Client die Ergebnisse abruft oder das Abfrage-Timeout erreicht ist. Streaming ist am effektivsten bei Abfragen, die nicht aggregieren und inkrementell zu Ergebnissen führen können. Abfragen`ORDER BY`, die Aggregationsfunktionen (wie`count()`, `collect()` oder`sum()`) enthalten oder `DISTINCT` erfordern, dass Neptune die vollständige Ergebnismenge berechnet, bevor Datensätze zurückgegeben werden. In diesen Fällen wird die clientseitige Speichernutzung pro Batch `FetchSize` immer noch begrenzt, aber der Server muss die gesamte Ergebnismenge im Speicher speichern, bevor das Streaming beginnt.

Note

Der OpenCypher HTTPS-Endpunkt liefert Ergebnisse mithilfe der HTTP-Chunked-Transferverschlüsselung, bei der Daten während der Erstellung an den Client gestreamt werden. Der Client kann das Tempo der Lieferung jedoch nicht kontrollieren. Es gibt kein Äquivalent `FetchSize` für den HTTPS-Endpunkt. Um den clientseitigen Speicherverbrauch bei großen Ergebnismengen zu kontrollieren, verwenden Sie eine Bolt-Treiberverbindung mit `FetchSize` konfiguriertem.

Konfiguration FetchSize und Verarbeitung von Ergebnissen in Batches

Legen Sie in `FetchSize` der Sitzungs- oder Treiberkonfiguration fest, wie viele Datensätze die Treiberanfragen von Neptune pro PULL-Nachricht enthalten. Beachten Sie, dass dadurch der Fluss vom Treiber zum Server `FetchSize` gesteuert wird, nicht die Batchverarbeitung auf Anwendungsebene. Um Ergebnisse in Batches auf Anwendungsebene zu verarbeiten (z. B. um Zwischenarbeiten zwischen Datensatzgruppen durchzuführen), verwenden Sie eine Hilfsmethode, die Datensätze aus dem Lazy-Iterator des Treibers sammelt.

Die folgenden Beispiele zeigen beides: die Konfiguration in der Sitzung und die Verwendung einer Hilfsmethode, um Datensätze zur Verarbeitung in Batches `FetchSize` auf Anwendungsebene zu gruppieren.

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();
}
```

Eifriger Konsum versus inkrementeller Konsum

Die folgenden Methoden blockieren, bis die gesamte Ergebnismenge im Speicher erfasst ist. Dadurch wird verhindert, dass Ihre Anwendung auf Ergebnisse reagiert, sobald sie eintreffen:

- Java: `result.list()`
- Python:`result.data()`, `list(result)`
- Gehe:`result.Collect(ctx)`, `neo4j.EagerResultTransformer`
- .NET: `result.ToListAsync()`
- Node.js: `const { records } = await result`

Verwenden Sie die in den obigen Beispielen gezeigten Muster Lazy Iteration und Batch Helper, um die Ergebnisse inkrementell zu verarbeiten.

Hinweise zur OpenCypher-Abfrage

> **Important**
>
> Der OpenCypher-Abfragehinweis ist erst ab Engine-Version 1.3.2.0 und höher verfügbar.

In Amazon Neptune können Sie die `USING` Klausel verwenden, um Abfragehinweise für OpenCypher-Abfragen anzugeben. Mit diesen Hinweisen können Sie Optimierungs- und Bewertungsstrategien steuern.

Die Syntax für Abfragehinweise lautet:

```
USING {scope}:{hint} {value}
```

1. `{scope}`definiert den Bereich, in dem der Hinweis gilt: `Query` oder`Clause`.

   Ein Bereichswert von `Query` bedeutet, dass der Abfragehinweis für die gesamte Abfrage gilt (Abfrageebene).

   Ein Bereichswert von `Clause` bedeutet, dass der Abfragehinweis für die Klausel gilt, der der Hinweis vorausgeht (Klauselebene).
2. `{hint}`ist der Name des Abfragehinweises, der angewendet wird.
3. `{value}`ist das Argument für`{hint}`.

Bei den Werten kann die Groß- und Kleinschreibung nicht beachtet werden.

Gehen Sie beispielsweise wie folgt vor, um den Abfrageplan-Cache für eine Abfrage zu aktivieren:

```
Using QUERY:PLANCACHE "enabled"
MATCH (a:Person {firstName: "Erin", lastName: $lastName})
 RETURN a
```

> **Note**
>
> Derzeit werden die Abfragehinweise PLANCACHE, TIMEOUTMILLISECONDS und Assume für den Abfragebereich unterstützt. ConsistentDataTypes Die unterstützten Abfragehinweise sind unten aufgeführt.

Themen

- Cache-Hinweis für den OpenCypher-Abfrageplan
- AssumeConsistentDataTypes Hinweis
- Hinweis auf das Timeout für OpenCypher-Abfragen

Cache-Hinweis für den OpenCypher-Abfrageplan

Das Verhalten des Abfrageplan-Caches kann pro Abfrage (parametrisiert oder nicht) durch einen Abfragehinweis auf Abfrageebene außer Kraft gesetzt werden. `QUERY:PLANCACHE` Es muss zusammen mit der Klausel verwendet werden. `USING` Der Abfragehinweis akzeptiert `enabled` oder `disabled` als Wert. Weitere Hinweise zum Abfrageplan-Cache finden Sie unterPlan-Cache in Amazon Neptune abfragen.

AWS CLI

Erzwingen, dass der Plan zwischengespeichert oder wiederverwendet wird:

```
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"
```

Mit Parametern:

```
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}'
```

Erzwingen, dass der Plan weder zwischengespeichert noch wiederverwendet wird:

```
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"
```

Weitere Informationen finden Sie unter execute-open-cypher-query in der Befehlsreferenz. 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'])
```

AWS SDK-Beispiele in anderen Sprachen finden Sie unter. AWS SDK

awscurl

Erzwingen, dass der Plan zwischengespeichert oder wiederverwendet wird:

```
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**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

curl

Erzwingen, dass der Plan zwischengespeichert oder wiederverwendet wird:

```
curl https://your-neptune-endpoint:port/openCypher \
  -d "query=Using QUERY:PLANCACHE \"enabled\" MATCH(n) RETURN n LIMIT 1"
```

Mit Parametern:

```
curl https://your-neptune-endpoint:port/openCypher \
  -d "query=Using QUERY:PLANCACHE \"enabled\" RETURN \$arg" \
  -d "parameters={\"arg\": 123}"
```

Erzwingen, dass der Plan weder zwischengespeichert noch wiederverwendet wird:

```
curl https://your-neptune-endpoint:port/openCypher \
  -d "query=Using QUERY:PLANCACHE \"disabled\" MATCH(n) RETURN n LIMIT 1"
```

AssumeConsistentDataTypes Hinweis

OpenCypher folgt einem Paradigma, bei dem Übereinstimmungen numerischer Datentypen (z. B. int, byte, short, long usw.) im Rahmen der Semantik zur Typenförderung durchgeführt werden. Wenn zum Beispiel alle Eigenschaften mit einem Eingabewert 10 mit kurzem Typ unter Semantik zur Typenförderung gesucht werden, würde es auch nach Eigenschaften suchen, die 10 als langen Wert haben. In einigen Fällen kann die Typumwandlung zu Mehraufwand führen und dazu führen, dass Abfragepläne weniger effizient sind, als sie es sein könnten, wenn keine Typumwandlung durchgeführt würde. Insbesondere in Fällen, in denen Datentypen konsistent in den Daten verwendet werden (z. B. wenn das Alter aller Personen als Langwert gespeichert wird), verursacht die Durchführung von Typbeförderungen Mehraufwand, ohne das Abfrageergebnis zu beeinflussen.

Um eine Optimierung für Fälle zu ermöglichen, in denen bekannt ist, dass die in der Datenbank gespeicherten numerischen Eigenschaftsdaten einen konsistenten Typ haben, kann ein Abfragehinweis mit dem Namen `assumeConsistentDataTypes` (mit Wert`true/false`, Standardeinstellung`false`) verwendet werden. Wenn dieser Abfragehinweis mit einem Wert von geliefert wird`true`, geht die Engine davon aus, dass die einzigen Eigenschaftswerte immer lang oder doppelt sind, und überspringt die Semantik der Typweiterleitung. In der Abfrage angegebene numerische Werte werden entweder als Langwerte (für Werte ohne Gleitkommawerte) oder als Doppelwerte (für Fließkommawerte) betrachtet.

Wenn die Daten konsistent einen einzigen Datentyp verwenden (z. B. alle Altersstufen werden als gespeichert`long`), kann mithilfe des `assumeConsistentDataTypes` Hinweises die Abfrage optimiert werden, indem unnötige Gleichheitsprüfungen für verschiedene numerische Typen übersprungen werden. Wenn die Daten jedoch inkonsistente Datentypen für dieselbe Eigenschaft aufweisen, kann die Verwendung des Hinweises dazu führen, dass einige Ergebnisse übersehen werden, da die Abfrage nur dem einzigen Datentyp entspricht, den der Hinweis annimmt.

```
# 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.
```

Der Unterschied kann auch anhand der Erläuterung bestätigt werden.

Ohne die Erklärung:

```
# 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
```

Mit dem Hinweis:

```
# 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
```

Hinweis auf das Timeout für OpenCypher-Abfragen

Das Verhalten beim Abfrage-Timeout kann für jede Abfrage anhand eines Abfragehinweises auf Abfrageebene konfiguriert werden. `QUERY:TIMEOUTMILLISECONDS` Es muss zusammen mit der Klausel verwendet werden. `USING` Der Abfragehinweis akzeptiert nichtnegative Werte vom Typ long als Wert.

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"
```

Weitere Informationen finden Sie unter execute-open-cypher-query in der Befehlsreferenz. 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'])
```

AWS SDK-Beispiele in anderen Sprachen finden Sie unter. AWS 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**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

curl

```
curl https://your-neptune-endpoint:port/openCypher \
  -d "query=USING QUERY:TIMEOUTMILLISECONDS 100 MATCH(n) RETURN n LIMIT 1"
```

Das Verhalten beim Abfrage-Timeout berücksichtigt das Minimum an Timeout auf Cluster-Ebene und Timeout auf Abfrageebene. Im Folgenden finden Sie Beispiele, um das Verhalten von Abfrage-Timeouts zu verstehen. Weitere Informationen zum Abfrage-Timeout auf Clusterebene finden Sie unter 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"
```

Wenn eine Abfrage das Timeout überschreitet, beendet Neptune sie und gibt einen Timeout-Fehler zurück. Ob Sie eine Abfrage, bei der das Timeout abgelaufen ist, erneut versuchen müssen, hängt von der Art des Fehlers und Ihrer Arbeitslast ab. Anleitungen finden Sie unter the section called “Ausnahmen und Wiederholungen”.

Einschränkungen für Neptune openCypher

Die Amazon-Neptune-Version von openCypher unterstützt weiterhin nicht alles, was in Cypher Query Language Reference, Version 9 spezifiziert ist, wie in Einhaltung der OpenCypher-Spezifikationen beschrieben. Zukünftige Versionen werden voraussichtlich viele dieser Einschränkungen beheben.

Ausnahmen für Neptune openCypher

Bei der Arbeit mit openCypher in Amazon Neptune können verschiedene Ausnahmen auftreten. Im Folgenden sind häufig auftretende Ausnahmen aufgeführt, die Sie entweder vom HTTPS-Endpunkt oder vom Bolt-Treiber erhalten können (alle Ausnahmen vom Bolt-Treiber werden als Serverstatus-

Ausnahmen gemeldet). Wenn die Abhilfe darin besteht, die Anfrage erneut zu versuchen, finden Sie unter Hinweise the section called “Ausnahmen und Wiederholungen” zur Entwicklung einer effektiven Wiederholungsstrategie.

HTTP-Code	Fehlermeldung	Abrufbar?	Abhilfe	
400	(Syntaxfehler, direkt vom openCypher-Parser propagiert)	Nein	Korrigieren Sie die Abfragesyntax und versuchen Sie es erneut.	
500	`Operation terminated (out of memory)`	Ja	Fügen Sie der Abfrage zusätzliche Filterkriterien hinzu, um den Speicherbedarf zu reduzieren.	
500	Operation beendet (Frist überschritten)	Ja	Erhöhen Sie das Abfrage-Timeout in der DB-Cluster-Parametergruppe oder wiederholen Sie die Anforderung.	
500	Operation beendet (vom Benutzer abgebrochen)	Ja	Wiederholen Sie die Anforderung.	
500	Die Datenbank zurücksetzung ist in Bearbeitung. Wiederholen Sie die Anforderung,	Ja	Wiederholen Sie die Anforderung, wenn die Zurücksetzung abgeschlossen ist.	

HTTP-Code	Fehlermeldung	Abrufbar?	Abhilfe	
	wenn der Cluster verfügbar ist.			
500	Die Operation ist aufgrund widersprü chlicher gleichzei tiger Vorgänge fehlgeschlagen (bitte wiederhol en). Zurzeit wird ein Rollback für Transaktionen ausgeführt.	Ja	Wiederholen Sie die Anforderung.	
400	*(operatio n name)* operation/feature Ausnahme nicht unterstützt	Nein	Die angegeben e Operation wird nicht unterstützt.	
400	Es wurde ein openCyphe r-Update für ein schreibge schütztes Replikat versucht.	Nein	Ändern Sie den Zielendpunkt in den Writer-En dpunkt.	
400	Malformed QueryException (Neptune zeigt den internen Parserstatus nicht an)	Nein	Korrigieren Sie die Abfragesy ntax und versuchen Sie es erneut.	

HTTP-Code	Fehlermeldung	Abrufbar?	Abhilfe	
400	Der Knoten kann nicht gelöscht werden, da er immer noch Beziehungen hat. Um diesen Knoten zu löschen, müssen Sie zuerst seine Beziehungen löschen.	Nein	Verwenden Sie `MATCH(n) DETACH DELETE(n)` anstelle von `MATCH (n) DELETE n`.	

HTTP-Code	Fehlermeldung	Abrufbar?	Abhilfe	
400	Ungültige Operation: Es wird versucht, die letzte Bezeichnung eines Knotens zu entfernen. Ein Knoten muss mindestens eine Bezeichnung besitzen.	Nein	In Neptune müssen alle Knoten mindestens eine Bezeichnung haben. Wenn Knoten ohne explizite Bezeichnung erstellt werden, wird die Standardbezeichnung `vertex` zugewiesen. Ändern Sie die Logik der and/or Abfragean wendung, um das letzte Label nicht zu löschen. Eine einzelne Bezeichnung eines Knotens kann aktualisiert werden, indem eine neue Bezeichnung festgelegt und dann die alte Bezeichnung entfernt wird.	

HTTP-Code	Fehlermeldung	Abrufbar?	Abhilfe	
500	Die maximale Anzahl der Anfragen wurde verletzt, Configure dQueueCapacity = {} für ConnID = {}	Ja	Derzeit können nur 8.192 gleichzeitige Anforderungen verarbeitet werden, unabhängig von Stack und Protokoll.	
500	Das maximale Verbindungslimit wurde überschritten.	Ja	Pro Instance sind lediglich 1 000 gleichzeitige Bolt-Verbindungen zulässig (für HTTP gibt es keine Einschränkungen).	
400	Erwartet wird [entweder: Knoten, Beziehung oder Pfad], erhalten wurde ein Literal.	Nein	Prüfen Sie, ob Sie die korrekten Argumente und die korrekte Abfragesyntax übergeben, und wiederholen Sie die Anforderung.	

HTTP-Code	Fehlermeldung	Abrufbar?	Abhilfe	
400	Der Eigenschaftswert muss ein einfaches Literal sein. Oder: Es wurde eine Map für Set-Eigenschaften erwartet, es wurde jedoch keine gefunden.	Nein	Eine SET-Klausel akzeptiert nur einfache Literale, keine zusammengesetzten Typen.	
400	Die gefundene Entität, die zum Löschen übergeben wurde, wurde nicht gefunden-	Nein	Prüfen Sie, ob die Entität, die Sie löschen möchten, in der Datenbank vorhanden ist.	
400	Der Benutzer hat keinen Zugriff auf die Datenbank.	Nein	Überprüfen Sie die Richtlinie für die verwendete IAM-Rolle.	
400	Es wurde kein Token als Teil der Anforderung übergeben.	Nein	Als Teil der Abfrageanforderung für einen IAM-fähigen Cluster muss ein ordnungsgemäß signiertes Token übergeben werden.	

HTTP-Code	Fehlermeldung	Abrufbar?	Abhilfe	
400	Die Fehlermeldung wird propagiert.	Nein	Wenden Sie sich mit der Anforderungs-ID an den AWS Support.	
500	Operation wurde beendet (interner Fehler).	Ja	Wenden Sie sich mit der Anforderungs-ID an den AWS Support.	

OpenCypher-Erweiterungen in Amazon Neptune

Amazon Neptune unterstützt die Referenzversion 9 der OpenCypher-Spezifikation. Einzelheiten finden Sie Einhaltung der OpenCypher-Spezifikationen in Amazon Neptune in Amazon Neptune. Darüber hinaus unterstützt Amazon Neptune die hier aufgeführten Funktionen. Sofern keine spezifischen Versionen erwähnt werden, sind die Funktionen in Neptune Database und Neptune Analytics verfügbar.

Query-time S3-Datenzugriff

Verfügbar in Neptune Database 1.4.7.0 und höher.

Neptune unterstützt die `neptune.read()` Funktion, CSV- oder Parquet-Daten aus Amazon S3 direkt in OpenCypher-Abfragen zu lesen. Im Gegensatz zum Bulk-Loader, der Daten vor der Abfrage importiert, `neptune.read()` greift er bei der Ausführung der Abfrage auf Amazon S3 S3-Daten zu.

Eine vollständige Dokumentation finden Sie unter. neptune.read ()

Die Funktion Neptune-specific **`join ()`**

Verfügbar in Neptune Database und Neptune Analytics.

Neptune implementiert die Funktion `join()`, die in der openCypher-Spezifikation nicht enthalten ist. Sie erstellt aus einer Liste von Zeichenfolgen-Literalen und einem Zeichenfolgen-Trennzeichen ein Zeichenfolgen-Literal. Sie verwendet zwei Argumente:

- Das erste Argument ist eine Liste von Zeichenfolgenliteralen.
- Das zweite Argument ist die Trennzeichenfolge, die aus null, einem oder mehreren Zeichen bestehen kann.

Beispiel:

```
join(["abc", "def", "ghi"], ", ")    // Returns "abc, def, ghi"
```

Die Neptune-specific Funktion remove () KeyFromMap

Verfügbar in Neptune Database und Neptune Analytics.

Neptune implementiert die Funktion `removeKeyFromMap()`, die in der openCypher-Spezifikation nicht enthalten ist. Sie entfernt einen angegebenen Schlüssel aus einer Map und gibt die resultierende neue Map zurück.

Die Funktion verwendet zwei Argumente:

- Das erste Argument ist die Map, aus der der Schlüssel entfernt werden soll.
- Das zweite Argument ist der Schlüssel, der aus der Map entfernt werden soll.

Die Funktion `removeKeyFromMap()` ist besonders nützlich in Situationen, in denen Sie Werte für einen Knoten oder eine Beziehung festlegen möchten, indem Sie eine Liste von Maps entladen. Beispiel:

```
UNWIND [{`~id`: 'id1', name: 'john'}, {`~id`: 'id2', name: 'jim'}] as val
CREATE (n {`~id`: val.`~id`})
SET n = removeKeyFromMap(val, '~id')
```

Benutzerdefinierte ID-Werte für Knoten- und Beziehungseigenschaften

Verfügbar in Neptune Database 1.2.0.2 und höher und Neptune Analytics.

Ab Engine-Version 1.2.0.2 hat Neptune die openCypher-Spezifikation erweitert, sodass Sie jetzt die `id`-Werte für Knoten und Beziehungen in den Klauseln `CREATE`, `MERGE` und `MATCH` angeben können. So können Sie benutzerfreundliche Zeichenfolgen anstelle systemgenerierter UUIDs zuweisen, um Knoten und Beziehungen zu identifizieren.

In Neptune Analytics sind benutzerdefinierte ID-Werte für Kanten nicht verfügbar.

> **Warning**
>
> Diese Erweiterung der openCypher-Spezifikation ist abwärtsinkompatibel, da `~id` jetzt als reservierter Eigenschaftsname gilt. Wenn Sie in Ihren Daten und Abfragen `~id` bereits als Eigenschaft verwenden, müssen Sie die vorhandene Eigenschaft zu einem neuen Eigenschaftsschlüssel migrieren und den alten entfernen. Siehe Was tun, wenn Sie derzeit `~id als Eigenschaft` verwenden.

Dies ist ein Beispiel für die Erstellung von Knoten und Beziehungen mit benutzerdefinierten IDS:

```
CREATE (n {`~id`: 'fromNode', name: 'john'})
  -[:knows {`~id`: 'john-knows->jim', since: 2020}]
  ->(m {`~id`: 'toNode', name: 'jim'})
```

Wenn Sie versuchen, eine benutzerdefinierte ID zu erstellen, die bereits verwendet wird, gibt Neptune den Fehler `DuplicateDataException` aus.

Dies ist ein Beispiel für die Verwendung einer benutzerdefinierten ID in einer `MATCH`-Klausel:

```
MATCH (n {`~id`: 'id1'})
RETURN n
```

Dies ist ein Beispiel für die Verwendung benutzerdefinierter IDs in einer `MERGE`-Klausel:

```
MATCH (n {name: 'john'}), (m {name: 'jim'})
MERGE (n)-[r {`~id`: 'john->jim'}]->(m)
RETURN r
```

Was tun, wenn Sie derzeit **`~id als Eigenschaft`** verwenden

An Engine-Version 1.2.0.2 wird der `~id`-Schlüssel in openCypher-Klauseln nicht mehr als Eigenschaft, sondern als `id` behandelt. Das bedeutet, dass der Zugriff auf eine Eigenschaft nicht mehr möglich ist, wenn Sie diese mit `~id` benannt haben.

Wenn Sie die Eigenschaft `~id` verwenden, müssen Sie vor dem Upgrade auf Engine-Version `1.2.0.2` oder höher zunächst die vorhandene Eigenschaft `~id` zu einem neuen Eigenschaftsschlüssel migrieren und dann die Eigenschaft `~id` entfernen. Betrachten Sie die folgende Abfrage:

- Sie erstellt eine neue Eigenschaft mit dem Namen 'newId' für alle Knoten.
- Sie kopiert den Wert der Eigenschaft '~id' in die Eigenschaft 'newID'.
- Sie entfernt die Eigenschaft '~id' aus den Daten.

```
MATCH (n)
WHERE exists(n.`~id`)
SET n.newId = n.`~id`
REMOVE n.`~id`
```

Dies muss für alle Beziehungen in den Daten ausgeführt werden, die die Eigenschaft `~id` besitzen.

Sie müssen auch alle Abfragen ändern, die auf die Eigenschaft `~id` verweisen. Beispielsweise würde diese Abfrage:

```
MATCH (n)
WHERE n.`~id` = 'some-value'
RETURN n
```

... wie folgt geändert werden:

```
MATCH (n)
WHERE n.newId = 'some-value'
RETURN n
```

Unterstützung für CALL-Unterabfragen in Neptune

Verfügbar in Neptune Database 1.4.1.0 und höher sowie Neptune Analytics.

Amazon Neptune unterstützt `CALL` Unterabfragen. Eine `CALL` Unterabfrage ist ein Teil der Hauptabfrage, der in einem isolierten Bereich für jede Eingabe der Unterabfrage ausgeführt wird. `CALL`

Angenommen, ein Diagramm enthält Daten über Personen, ihre Freunde und Städte, in denen sie gelebt haben. Mithilfe einer `CALL` Unterabfrage können wir die beiden größten Städte abrufen, in denen jeder Freund von jemandem gelebt hat:

```
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
```

In diesem Beispiel wird der `CALL { ... }` darin enthaltene Abfrageteil für jede Abfrage ausgeführt, auf `friend` die die vorherige MATCH-Klausel zutrifft. Wenn die innere Abfrage ausgeführt wird, beziehen sich die `ORDER` `LIMIT` UND-Klauseln auf die Städte, in denen ein bestimmter Freund gelebt hat. Wir erhalten also (höchstens) zwei Städte pro Freund.

Alle Abfrageklauseln sind in Unterabfragen verfügbar. `CALL` Dies schließt auch verschachtelte `CALL` Unterabfragen ein. Es gibt einige Einschränkungen für die erste `WITH` Klausel und die ausgegebenen Variablen, die im Folgenden erklärt werden.

Gültigkeitsbereich der Variablen innerhalb der CALL-Unterabfrage

Die Variablen aus den Klauseln vor der Unterabfrage, die in der `CALL` Unterabfrage verwendet werden, müssen von der ersten Klausel importiert werden. `WITH` Im Gegensatz zu regulären `WITH` Klauseln kann sie nur eine Liste von Variablen enthalten, erlaubt aber kein Aliasing und kann nicht zusammen mit`DISTINCT`,,`ORDER BY`, `WHERE` oder verwendet werden. `SKIP` `LIMIT`

Von der CALL-Unterabfrage zurückgegebene Variablen

Die Variablen, die von der `CALL` Unterabfrage ausgegeben werden, werden mit der letzten `RETURN` Klausel angegeben. Beachten Sie, dass sich die ausgegebenen Variablen nicht mit Variablen vor der `CALL` Unterabfrage überschneiden dürfen.

Einschränkungen

Derzeit werden Aktualisierungen innerhalb einer `CALL` Unterabfrage nicht unterstützt.

Neptune OpenCypher-Funktionen

Verfügbar in Neptune Database 1.4.1.0 und höher sowie Neptune Analytics.

Text IndexOf

```
textIndexOf(text :: STRING, lookup :: STRING, from = 0 :: INTEGER?, to =
-1 :: INTEGER?) :: (INTEGER?)
```

Gibt den Index des ersten Vorkommens von `lookup` im Bereich von `text` Offset `from` (einschließlich) bis Offset `to` (ausschließlich) zurück. Wenn -1 `to` ist, wird der Bereich bis zum Ende von fortgesetzt`text`. Die Indizierung basiert auf Null und wird in Unicode-Skalarwerten (keine Surrogatcodepunkte) ausgedrückt.

```
RETURN textIndexOf('Amazon Neptune', 'e')
{
  "results": [{
      "textIndexOf('Amazon Neptune', 'e')": 8
    }]
}
```

coll ToSet

`collToSet(values :: LIST OF ANY?) :: (LIST? OF ANY?)`

Gibt eine neue Liste zurück, die nur die eindeutigen Elemente aus der ursprünglichen Liste enthält. Die Reihenfolge der ursprünglichen Liste wird beibehalten (z. B. `[1, 6, 5, 1, 5]` Rückgabe`[1, 6, 5]`).

```
RETURN collToSet([1, 6, 5, 1, 1, 5])
{
  "results": [{
      "collToSet([1, 6, 5, 1, 1, 5])": [1, 6, 5]
    }]
}
```

collSubtract

`collSubtract(first :: LIST OF ANY?, second :: LIST OF ANY?) :: (LIST? OF ANY?)`

Gibt eine neue Liste zurück, die alle eindeutigen Elemente für das `first` Ausschließen von Elementen enthält. `second`

```
RETURN collSubtract([2, 5, 1, 0], [1, 5])
{
  "results": [{
      "collSubtract([2, 5, 1, 0], [1, 5])": [0, 2]
    }]
}
```

collIntersection

`collIntersection(first :: LIST? OF ANY?, second :: LIST? OF ANY?) :: (LIST? OF ANY?)`

Gibt eine neue Liste zurück, die alle eindeutigen Elemente des Schnittpunkts von `first` und `second` enthält.

```
RETURN collIntersection([2, 5, 1, 0], [1, 5])
{
  "results": [{
      "collIntersection([2, 5, 1, 0], [1, 5])": [1, 5]
    }]
}
```

Funktionen zum Sortieren

In den folgenden Abschnitten werden Funktionen zum Sortieren von Sammlungen definiert. Diese Funktionen verwenden (in einigen Fällen optionale) `config` Zuordnungsargumente oder eine Liste mehrerer solcher Zuordnungen, die den Sortierschlüssel und and/or die Sortierrichtung definieren:

```
{ key: STRING, order: STRING }
```

Hier `key` ist entweder eine Map- oder eine Node-Eigenschaft, deren Wert für die Sortierung verwendet werden soll. `order`ist entweder "" oder `asc` "`desc`" (Groß- und Kleinschreibung wird nicht beachtet), um eine aufsteigende bzw. absteigende Sortierung anzugeben. Standardmäßig erfolgt die Sortierung in aufsteigender Reihenfolge.

collSort

`collSort(coll :: LIST OF ANY, config :: MAP?) :: (LIST? OF ANY?)`

Gibt eine neue sortierte Liste zurück, die die Elemente aus der `coll` Eingabeliste enthält.

```
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?)`

Gibt eine Liste von Maps zurück, sortiert nach dem Wert der angegebenen `key` Eigenschaft.

```
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?)
```

Gibt eine Liste von Maps zurück, sortiert nach dem Wert der angegebenen `key` Eigenschaften, wobei optional Limit und Skip angewendet wird.

```
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?)`

Gibt eine sortierte Version der `coll` Eingabeliste zurück und sortiert die Knotenelemente nach den Werten ihrer jeweiligen `key` Eigenschaften.

```
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"
          }
        }]
    }]
}
```

Zeitliche Funktionen

Temporale Funktionen sind ab Neptune Version 1.4.5.0 und höher verfügbar.

Tag

```
day(temporal :: (datetime | date)) :: (LONG)
```

Gibt den Wert `day` des Monats aus einem Oder-Wert zurück. `datetime` `date` Für`datetime`: Werte werden auf der Grundlage der Eingabe vor dem Extrahieren des Tages auf UTC normalisiert. Für`date`: Der Tag wird basierend auf der Zeitzone extrahiert.

Die `datetime` Eingabe ist sowohl in Neptune Database als auch in Neptune Analytics verfügbar:

```
RETURN day(datetime('2021-06-03T01:48:14Z'))
{
  "results": [{
      "day(datetime('2021-06-03T01:48:14Z'))": 3
    }]
}
```

Hier `datetime` ist das auf UTC normalisiert, sodass + 08:00 wieder auf den 2. Juni verschoben wird.

```
RETURN day(datetime('2021-06-03T00:00:00+08:00'))
{
  "results": [{
      "day(datetime('2021-06-03T00:00:00+08:00'))": 2
    }]
}
```

Die `date` Eingabe ist nur in Neptune Analytics verfügbar:

```
RETURN day(date('2021-06-03Z'))
{
  "results": [{
      "day(date('2021-06-03Z'))": 3
    }]
}
```

Die Zeitzone wird `date` beibehalten, wobei der 3. Juni beibehalten wird.

```
RETURN day(date('2021-06-03+08:00'))
{
  "results": [{
      "day(date('2021-06-03+08:00'))": 3
    }]
}
```

Monat

`month(temporal :: (datetime | date)) :: (LONG)`

Gibt den Monat aus einem `datetime` `date` OR-Wert (1—12) zurück. Für`datetime`: Werte werden auf der Grundlage der Eingabe vor dem Extrahieren des Monats auf UTC normalisiert. Für`date`: Der Monat wird basierend auf der Zeitzone extrahiert.

Die `datetime` Eingabe ist sowohl in Neptune Database als auch in Neptune Analytics verfügbar:

```
RETURN month(datetime('2021-06-03T01:48:14Z'))
{
  "results": [{
      "month(datetime('2021-06-03T01:48:14Z'))": 6
    }]
}
```

Hier `datetime` ist das auf UTC normalisiert, sodass + 08:00 wieder auf den 31. Mai verschoben wird.

```
RETURN month(datetime('2021-06-01T00:00:00+08:00'))
{
  "results": [{
      "month(datetime('2021-06-01T00:00:00+08:00'))": 5
    }]
}
```

Die `date` Eingabe ist nur in Neptune Analytics verfügbar:

```
RETURN month(date('2021-06-03Z'))
{
  "results": [{
      "month(date('2021-06-03Z'))": 6
    }]
}
```

Die Zeitzone wird `date` beibehalten, wobei der 1. Juni beibehalten wird.

```
RETURN month(date('2021-06-01+08:00'))
{
  "results": [{
```

```
      "month(date('2021-06-01+08:00'))": 6
    }]
}
```

Jahr

`year(temporal :: (datetime | date)) :: (LONG)`

Gibt das Jahr anhand eines Werts `datetime` oder `date` zurück. Für`datetime`: Die Werte werden auf der Grundlage der Eingabe auf UTC normalisiert, bevor das Jahr extrahiert wird. Für`date`: Das Jahr wird auf der Grundlage der Zeitzone extrahiert.

Die `datetime` Eingabe ist sowohl in Neptune Database als auch in Neptune Analytics verfügbar:

```
RETURN year(datetime('2021-06-03T01:48:14Z'))
{
  "results": [{
      "year(datetime('2021-06-03T01:48:14Z'))": 2021
    }]
}
```

Hier `datetime` ist das auf UTC normalisiert, also + 08:00 verschiebt sich zurück auf den 31. Dezember 2020.

```
RETURN year(datetime('2021-01-01T00:00:00+08:00'))
{
  "results": [{
      "year(datetime('2021-01-01T00:00:00+08:00'))": 2020
    }]
}
```

Die `date` Eingabe ist nur in Neptune Analytics verfügbar:

```
RETURN year(date('2021-06-03Z'))
{
  "results": [{
      "year(date('2021-06-03Z'))": 2021
    }]
}
```

Die Zeitzone wird `date` beibehalten, wobei Juni 2021 beibehalten wird.

```
RETURN year(date('2021-01-01+08:00'))
{
  "results": [{
      "year(date('2021-01-01+08:00'))": 2021
    }]
}
```

Neptune OpenCypher-Funktionen

Verfügbar in Neptune Database 1.4.6.0 und höher und Neptune Analytics.

reduzieren ()

Reduce verarbeitet jedes Listenelement sequentiell, indem es es mit einer laufenden Summe oder einem „Akkumulator“ kombiniert. Ausgehend von einem Anfangswert aktualisiert es den Akkumulator nach jeder Operation und verwendet diesen aktualisierten Wert in der nächsten Iteration.

`for i in (0, ..., n) acc = acc X list[I], where X denotes any binary operator`

Sobald alle Elemente verarbeitet wurden, wird das endgültige kumulierte Ergebnis zurückgegeben.

Eine typische Reduce () -Struktur wäre - `reduce(accumulator = initial , variable IN list | expression)`

Typspezifikationen:

`- 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`

Einschränkungen:

Derzeit unterstützt der `reduce()` Ausdruck nur:

- Numerische Multiplikation
- Numerische Addition
- Zeichenfolgenverkettung
- Verkettung von Listen

Sie werden durch den Operator oder dargestellt. + * Der Ausdruck sollte ein binärer Ausdruck sein, wie unten angegeben - `expression pattern: accumulator + any variable or accumulator * any variable`

Behandlung von Überläufen:

Neptune erkennt während der `reduce()` Auswertung einen numerischen Überlauf und reagiert je nach Datentyp unterschiedlich:

```
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.
```

Beispiele:

Sehen Sie sich die folgenden Beispiele für die Funktion reduce () an.

```


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 unterstützt ein `CALL` Verfahren `neptune.read` zum Lesen von Daten aus Amazon S3 und zum Ausführen einer OpenCypher-Abfrage (Lesen, Einfügen, Aktualisieren) unter Verwendung der Daten. Die Prozedur gibt jede Zeile in der Datei als deklarierte Ergebnisvariablenzeile zurück. Es verwendet die IAM-Anmeldeinformationen des Anrufers, um auf die Daten in Amazon S3 zuzugreifen. Informationen Verwaltung der Berechtigungen für neptune.read () zum Einrichten der Berechtigungen finden Sie unter. Die AWS Region des Amazon S3 S3-Buckets muss sich in derselben Region befinden, in der sich die Instance befindet. Derzeit werden regionsübergreifende Lesevorgänge nicht unterstützt.

Syntax

```
CALL neptune.read(
  {
    source: "string",
    format: "parquet/csv",
    concurrency: 10
  }
)
YIELD row
...
```

Eingaben

- source (erforderlich) — Amazon S3 S3-URI zu einem einzelnen Objekt. Das Amazon S3 S3-Präfix für mehrere Objekte wird nicht unterstützt.
- Format (erforderlich) — `parquet` und `csv` werden unterstützt.
  - Weitere Informationen zum unterstützten Parquet-Format finden Sie unterUnterstützte Parquet-Spaltentypen.
  - Weitere Informationen zum unterstützten CSV-Format finden Sie unterGremlin-Ladedatenformat.
- Parallelität (optional) — Typ: Ganzzahl 0 oder höher. Standard: 0 Gibt die Anzahl der Threads an, die zum Lesen der Datei verwendet werden sollen. Wenn der Wert 0 ist, wird die von der Ressource maximal zulässige Anzahl von Threads verwendet. Für Parquet wird empfohlen, die Einstellung auf eine bestimmte Anzahl von Zeilengruppen festzulegen.

Outputs

Der Befehl neptune.read gibt zurück:

- row - Typ: MAP
  - Jede Zeile in der Datei, wobei die Schlüssel die Spalten und die Werte die in jeder Spalte gefundenen Daten sind.
  - Sie können auf die Daten jeder Spalte wie bei einer Eigenschaft access (`row.col`) zugreifen.

Bewährte Methoden für neptune.read ()

Neptune S3-Lesevorgänge können speicherintensiv sein. Bitte verwenden Sie Instance-Typen, die sich gut für Produktions-Workloads eignen, wie unter Instance-Typen für Amazon Neptune auswählen beschrieben.

Die Speichernutzung und die Leistung von `neptune.read()` Anfragen werden durch eine Vielzahl von Faktoren wie Dateigröße, Anzahl der Spalten, Anzahl der Zeilen und Dateiformat beeinflusst. Je nach Struktur funktionieren kleine Dateien (z. B. CSV-Dateien mit 100 MB oder weniger, Parquet-Dateien mit 20 MB oder weniger) möglicherweise zuverlässig auf den meisten produktionsgeeigneten Instance-Typen, wohingegen größere Dateien möglicherweise viel Speicherplatz benötigen, den kleinere Instance-Typen nicht bereitstellen können.

Beim Testen dieser Funktion wird empfohlen, mit kleinen Dateien zu beginnen und diese schrittweise zu skalieren, um sicherzustellen, dass Ihr Lese-Workload an Ihre Instance-Größe angepasst werden kann. Wenn Sie feststellen, dass `neptune.read()` Anfragen zu Ausnahmen wegen unzureichenden Speichers oder zu Instanzneustarts führen, sollten Sie erwägen, Ihre Dateien in kleinere Teile aufzuteilen, die Dateikomplexität zu reduzieren oder ein Upgrade auf größere Instance-Typen durchzuführen.

Beispiele für Abfragen mit Parquet

Die folgende Beispielabfrage gibt die Anzahl der Zeilen in einer bestimmten Parquet-Datei zurück:

```
CALL neptune.read(
  {
    source: "<s3 path>",
    format: "parquet"
  }
)
YIELD row
```

```
RETURN count(row)
```

Sie können das Abfragebeispiel mithilfe der `execute-open-cypher-query` Operation in ausführen, AWS CLI indem Sie den folgenden Code ausführen:

```
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
```

Eine Abfrage kann flexibel mit Zeilen umgehen, die aus einer Parquet-Datei gelesen werden. Die folgende Abfrage erstellt beispielsweise einen Knoten mit einem Feld, das auf Daten aus der Parquet-Datei festgelegt ist:

```
CALL neptune.read(
  {
    source: "<s3 path>",
    format: "parquet"
  }
)
YIELD row
CREATE (n {someField: row.someCol})
RETURN n
```

> **Warning**
>
> Es wird nicht als gute Praxis angesehen, eine umfangreiche, ergebnisreiche Klausel wie `MATCH(n)` vor einer `CALL` Klausel zu verwenden. Dies würde aufgrund eines Kreuzprodukts zwischen eingehenden Lösungen aus früheren Klauseln und den von neptune.read gelesenen Zeilen zu einer lang andauernden Abfrage führen. Es wird empfohlen, die Abfrage mit neptune.read zu starten. `CALL`

Unterstützte Parquet-Spaltentypen

Parquet-Datentypen:

- NULL
- BOOLEAN

- FLOAT
- DOUBLE
- STRING
- GANZZAHL MIT VORZEICHEN: UINT8, UINT16, UINT32, UINT64
- MAP: Unterstützt nur eine Ebene. Unterstützt keine verschachtelten Elemente.
- LIST: Unterstützt nur eine Ebene. Unterstützt keine verschachtelten Elemente.

Neptune-specific Datentypen:

Im Gegensatz zu den Eigenschaftsspaltenüberschriften des CSV-Formats müssen die Eigenschaftsspaltenüberschriften des Parquet-Formats nur die Eigenschaftsnamen enthalten, sodass weder die Typnamen noch die Kardinalität erforderlich sind.

Es gibt jedoch einige spezielle Spaltentypen im Parquet-Format, für die Anmerkungen in den Metadaten erforderlich sind, darunter der Typ Beliebig, der Datentyp, der DateTime-Typ und der Geometrietyp. Das folgende Objekt ist ein Beispiel für die erforderliche Metadaten-Anmerkung für Dateien, die Spalten dieser speziellen Typen enthalten:

```
"metadata": {
    "anyTypeColumns": ["UserCol1"],
    "dateTypeColumns": ["UserCol2"],
    "dateTimeTypeColumns": ["UserCol3"],
    "geometryTypeColumns": ["UserCol4"]
}
```

Im Folgenden finden Sie Einzelheiten zur erwarteten Nutzlast im Zusammenhang mit diesen Typen:

- Der Spaltentyp Beliebig wird in den Benutzerspalten unterstützt. Ein Any-Typ ist ein Typ „syntaktischer Zucker“ für alle anderen Typen, die wir unterstützen. Es ist äußerst nützlich, wenn eine Benutzerspalte mehrere Typen enthält. Die Nutzlast eines Werts vom Typ Any ist eine Liste von JSON-Zeichenfolgen wie folgt:`{"value": "10", "type": "Int"};{"value": "1.0", "type": "Float"}`, die in jeder einzelnen JSON-Zeichenfolge ein Wertfeld und ein Typfeld enthält. Der Kardinalitätswert einer Any-Spalte ist festgelegt, was bedeutet, dass die Spalte mehrere Werte akzeptieren kann.
  - Neptune unterstützt die folgenden Typen in einem beliebigen Typ: Bool (oder Boolean), Byte, Short, Int, Long,,,, Float UnsignedByte UnsignedShort, Double UnsignedInt UnsignedLong, Date, dateTime, String und Geometry.

  - Der Vektortyp wird in Any Type nicht unterstützt.
  - Verschachtelt Jeder Typ wird nicht unterstützt. Beispiel, `{"value": {"value": "10", "type": "Int"}, "type": "Any"}`.
- Spalten vom Typ Date und Datetime werden in den Benutzerspalten unterstützt. Die Nutzdaten dieser Spalten müssen als Zeichenketten im XSD-Format oder in einem der folgenden Formate bereitgestellt werden:
  - yyyy- MM-dd
  - yyyy- MM-ddTHH:mm
  - yyyy- MM-ddTHH:mm:ss
  - yyyy- MM-ddTHH:mm:ssZ
  - yyyy- MM-ddTHH:mm:ss.SSSZ
  - yyyy- MM-ddTHH:mm:ss [+|-] hhmm
  - yyyy- MM-ddTHH:mm:ss.SSS [+|-] hhmm
- In den Benutzerspalten wird der Spaltentyp Geometrie unterstützt. Die Nutzdaten dieser Spalten dürfen nur Geometrie-Primitive des Typs Point enthalten, die als Zeichenketten im Well-known Textformat (WKT) bereitgestellt werden. Beispielsweise wäre POINT (30 10) ein gültiger Geometriewert.

Beispiel für eine Parkett-Ausgabe

Bei einer Parquet-Datei wie dieser:

```
<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   |
+--------+---------+-------------+----------------------+------------+------------
+----------+
```

Hier ist ein Beispiel für die Ausgabe, die von neptune.read mit der folgenden Abfrage zurückgegeben wurde:

```
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"
 }
 }]
}
```

Derzeit gibt es keine Möglichkeit, einem Datenfeld, das aus einer Parquet-Datei stammt, eine Knoten- oder Kantenbeschriftung zuzuweisen. Es wird empfohlen, die Abfragen in mehrere Abfragen zu partitionieren, eine für jede 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)
```

Beispiele für Abfragen mit CSV

In diesem Beispiel gibt die Abfrage die Anzahl der Zeilen in einer bestimmten CSV-Datei zurück:

```
CALL neptune.read(
  {
    source: "<s3 path>",
    format: "csv"
  }
)
YIELD row
```

```
RETURN count(row)
```

Sie können das Abfragebeispiel mithilfe der Operation execute-open-cypher-query in ausführen, indem Sie den AWS CLI folgenden Code ausführen:

```
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
```

Eine Abfrage kann flexibel mit Zeilen umgehen, die aus einer CSV-Datei gelesen werden. Die folgende Abfrage erstellt beispielsweise einen Knoten mit einem Feld, das auf Daten aus einer CSV-Datei festgelegt ist:

```
CALL neptune.read(
  {
    source: "<s3 path>",
    format: "csv"
  }
)
YIELD row
CREATE (n {someField: row.someCol})
RETURN n
```

> **Warning**
>
> Es wird nicht als gute Praxis angesehen, vor einer CALL-Klausel eine umfangreiche Ergebnisklausel wie MATCH (n) zu verwenden. Dies würde aufgrund eines Kreuzprodukts zwischen eingehenden Lösungen aus früheren Klauseln und den von neptune.read gelesenen Zeilen zu einer lang andauernden Abfrage führen. Es wird empfohlen, die Abfrage mit CALL neptune.read zu starten.

Eigenschaftsspaltenüberschriften

Sie können eine Spalte (`:`) für eine Eigenschaft angeben, indem Sie die folgende Syntax verwenden. Bei den Typnamen muss die Groß- und Kleinschreibung nicht berücksichtigt werden. Wenn ein Doppelpunkt in einem Eigenschaftsnamen vorkommt, muss ihm ein umgekehrter Schrägstrich vorangestellt werden:. `\:`

```
propertyname:type
```

Note

- Leerzeichen, Kommas, Zeilenumbrüche und Zeilenumbrüche sind in den Spaltenüberschriften nicht zulässig, sodass Eigenschaftsnamen diese Zeichen nicht enthalten dürfen.
- Sie können eine Spalte für einen Array-Typ festlegen, indem Sie dem Typ `[]` hinzufügen:

  ```
  propertyname:type[]
  ```

- Edge-Eigenschaften können nur einen einzelnen Wert haben und verursachen einen Fehler, wenn ein Array-Typ oder ein zweiter Wert angegeben ist. Das folgende Beispiel zeigt die Spaltenüberschrift für eine Eigenschaft namens age vom Typ Int:

  ```
  age:Int
  ```

Für jede Zeile in der Datei ist eine Ganzzahl in dieser Position erforderlich oder sie muss leer sein. Zeichenkettenarrays sind zulässig, aber Zeichenketten in einem Array dürfen das Semikolon (`;`) nicht enthalten, es sei denn, es wird mit einem umgekehrten Schrägstrich () maskiert. `\;`

Unterstützte CSV-Spaltentypen

- BOOL (oder BOOLEAN) — Zulässige Werte: true, false. Gibt ein boolesches Feld an. Jeder andere Wert als true wird als falsch behandelt.
- FLOAT — Bereich: 32-Bit-Gleitkommazahl nach IEEE 754, einschließlich Infinity, INF, -Infinity, -INF und NaN (keine Zahl).
- DOUBLE - Bereich: 64-Bit-IEEE 754-Fließkommazahl, einschließlich Infinity, INF, -Infinity, -INF und NaN (keine Zahl).
- ZEICHENFOLGE -
  - Anführungszeichen sind optional. Kommas, Zeilenumbrüche und Zeilenumbrüche werden automatisch maskiert, wenn sie in einer Zeichenfolge enthalten sind, die von doppelten Anführungszeichen („) umgeben ist. Beispiel: „Hello, World“.

  - Um Anführungszeichen in eine Zeichenfolge in Anführungszeichen einzufügen, können Sie das Anführungszeichen umgehen, indem Sie zwei in einer Reihe verwenden: Beispiel: „Hello „" World" ""“.
  - Zeichenkettenarrays sind zulässig, aber Zeichenketten in einem Array dürfen das Semikolon (;) nicht enthalten, es sei denn, es wird mit einem umgekehrten Schrägstrich (\;) maskiert).
  - Wenn Sie Zeichenfolgen in einem Array in Anführungszeichen setzen möchten, müssen Sie das gesamte Array mit einem Satz Anführungszeichen versehen. Beispiel: „Zeichenfolge eins; Zeichenfolge 2; Zeichenfolge 3".
- DATE, DATETIME — Die Datetime-Werte können entweder im XSD-Format oder in einem der folgenden Formate bereitgestellt werden:
  - yyyy- MM-dd
  - yyyy- MM-ddTHH:mm
  - yyyy- MM-ddTHH:mm:ss
  - yyyy- MM-ddTHH:mm:ssZ
  - yyyy- MM-ddTHH:mm:ss.SSSZ
  - yyyy- MM-ddTHH:mm:ss [+|-] hhmm
  - yyyy- MM-ddTHH:mm:ss.SSS [+|-] hhmm
- GANZZAHL MIT VORZEICHEN -
  - Byte: -128 bis 127
  - Kurz: -32768 bis 32767
  - Int: -2^31 bis 2^31-1
  - Lang: -2^63 bis 2^63-1

Neptune-specific Spaltentypen:

- In den Benutzerspalten wird der Spaltentyp Beliebig unterstützt. Ein Any-Typ ist ein Typ „syntaktischer Zucker“ für alle anderen Typen, die wir unterstützen. Es ist äußerst nützlich, wenn eine Benutzerspalte mehrere Typen enthält. Die Nutzlast eines Werts vom Typ Any ist eine Liste von JSON-Zeichenfolgen wie folgt:`{"value": "10", "type": "Int"};{"value": "1.0", "type": "Float"}`, die in jeder einzelnen JSON-Zeichenfolge ein Wertfeld und ein Typfeld enthält. Die Spaltenüberschrift eines beliebigen Typs lautet PropertyName:Any. Der Kardinalitätswert einer Any-Spalte ist festgelegt, was bedeutet, dass die Spalte mehrere Werte akzeptieren kann.

  - Neptune unterstützt die folgenden Typen in einem beliebigen Typ: Bool (oder Boolean), Byte, Short, Int, Long,,,, Float UnsignedByte UnsignedShort, Double UnsignedInt UnsignedLong, Date, dateTime, String und Geometry.
  - Der Vektortyp wird in Any Type nicht unterstützt.
  - Verschachtelt Jeder Typ wird nicht unterstützt. Beispiel, `{"value": {"value": "10", "type": "Int"}, "type": "Any"}`.
- Ein Geometriespaltentyp wird in den Benutzerspalten unterstützt. Die Nutzdaten dieser Spalten dürfen nur Geometrie-Primitive des Typs Point enthalten, die als Zeichenketten im Well-known Textformat (WKT) bereitgestellt werden. Beispielsweise wäre POINT (30 10) ein gültiger Geometriewert.

Beispiel für eine CSV-Ausgabe

Angesichts der folgenden CSV-Datei:

```
<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
```

Dieses Beispiel zeigt die Ausgabe, die von neptune.read mit der folgenden Abfrage zurückgegeben wurde:

```
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
      }
    }]
}
```

Derzeit gibt es keine Möglichkeit, einem Datenfeld, das aus einer CSV-Datei stammt, eine Knoten- oder Kantenbeschriftung zuzuweisen. Es wird empfohlen, die Abfragen in mehrere Abfragen zu partitionieren, eine für jede 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)
```

Verwaltung der Berechtigungen für neptune.read ()

Erforderliche IAM-Richtlinien

Um OpenCypher-Abfragen auszuführen, die verwendenneptune.read(), müssen Sie über die entsprechenden Berechtigungen für den Zugriff auf Daten in Ihrer Neptune-Datenbank verfügen. Read-only Abfragen erfordern die Aktion. ReadDataViaQuery Abfragen, die Daten ändern, erfordern WriteDataViaQuery Einfügungen oder DeleteDataViaQuery Löschungen. Im folgenden Beispiel werden alle drei Aktionen für den angegebenen Cluster gewährt.

Darüber hinaus benötigen Sie Berechtigungen für den Zugriff auf den S3-Bucket, der Ihre Datendateien enthält. Die Neptunes3Access-Richtlinienerklärung gewährt die erforderlichen S3-Berechtigungen:

- **s3:ListBucket**: Erforderlich, um das Vorhandensein eines Buckets und den Inhalt der Liste zu überprüfen.
- **s3:GetObject**: Erforderlich, um auf das angegebene Objekt zuzugreifen, damit sein Inhalt zur Integration in OpenCypher-Abfragen gelesen werden kann.

Wenn Ihr S3-Bucket serverseitige Verschlüsselung mit verwendet AWS KMS, müssen Sie auch KMS-Berechtigungen gewähren. Die Neptunes3KMSAccess-Richtlinienanweisung ermöglicht es Neptune, Daten zu entschlüsseln und Datenschlüssel zu generieren, wenn auf verschlüsselte S3-Objekte zugegriffen wird. Die Bedingung beschränkt den KMS-Betrieb auf Anfragen, die von S3- und RDS-Diensten in Ihrer Region stammen.

- **kms:Decrypt**: Erforderlich, um das verschlüsselte Objekt zu entschlüsseln, damit seine Daten von Neptune gelesen werden können.
- **kms:GenerateDataKey**: Wird auch von der S3-API benötigt, die zum Abrufen von zu lesenden Objekten verwendet wird.

```
{
```

```
  "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"
        ]
      }
  }
}
```

Wichtige Voraussetzungen

Diese Berechtigungen und Voraussetzungen gewährleisten die sichere und zuverlässige Integration von S3-Daten in OpenCypher-Abfragen bei gleichzeitiger Aufrechterhaltung angemessener Zugriffskontrollen und Datenschutzmaßnahmen.

- IAM-Authentifizierung: Diese Funktion wird nur für Neptune-Cluster mit aktivierter IAM-Authentifizierung unterstützt. Unter Sichern Ihrer Amazon Neptune Neptune-Datenbank finden Sie detaillierte Anweisungen zum Erstellen von Clustern mit IAM-Authentifizierung und zum Herstellen einer Verbindung zu diesen.
- VPC-Endpunkt:
  - Ein Gateway-type VPC-Endpunkt für Amazon S3 ist erforderlich, damit Neptune mit Amazon S3 kommunizieren kann.
  - Um eine benutzerdefinierte AWS KMS Verschlüsselung in der Abfrage zu verwenden, AWS KMS ist ein Interface-type VPC-Endpunkt für erforderlich, mit dem Neptune kommunizieren kann. AWS KMS
  - Detaillierte Anweisungen zur Konfiguration dieses Endpunkts finden Sie unter Erstellen des Amazon S3 S3-VPC-Endpunkts.

Räumliche Daten

Amazon Neptune unterstützt jetzt räumliche Abfragen, sodass Sie geometrische Daten in Ihrem Diagramm speichern und analysieren können. Räumliche Merkmale werden zwar häufig für geografische Standorte (wie Koordinaten auf einer Karte) verwendet, funktionieren aber mit allen zweidimensionalen Daten, bei denen Position und Nähe wichtig sind. Verwenden Sie diese Funktion, um Fragen wie „Welche Geschäfte befinden sich im Umkreis von 5 Meilen von diesem Kunden?“ zu beantworten. , „Finden Sie alle Lieferwege, die sich mit diesem Servicebereich überschneiden“ oder „Welche Komponenten in diesem Grundriss überschneiden sich mit der HLK-Zone?“ Neptune implementiert räumliche Unterstützung mithilfe von branchenüblichen Spatial Types-Funktionen, die mit Punkten, Polygonen und anderen geometrischen Formen funktionieren. Sie können räumliche Daten als Eigenschaften auf Knoten und Kanten speichern und dann räumliche Funktionen verwenden, um Entfernungen zu berechnen, zu überprüfen, ob Punkte innerhalb von Grenzen liegen, oder überlappende Regionen zu finden — alles innerhalb Ihrer OpenCypher-Abfragen.

Häufige Anwendungsfälle:

- Geografische Anwendungen: Location-based Empfehlungen, Geofencing, Routenplanung und Gebietsanalyse
- Anlagen- und Raummanagement: Grundrisse, Platzierung der Ausrüstung und Zonenabdeckung
- Netzwerktopologie: Kartierung der physischen Infrastruktur, Versorgungsbereiche und Dienstgrenzen
- Design und CAD: Positionierung von Komponenten, Kollisionserkennung und räumliche Beziehungen in 2D-Entwürfen
- Spieleentwicklung: Positionierung der Charaktere, Kollisionserkennung und Berechnungen des Wirkungsbereichs

Die Implementierung von Spatial Types in Amazon Neptune folgt wie andere Datenbanken den ISO/IEC 13249-3:2016-Richtlinien. Sie Geofunktionen sind in der OpenCypher-Abfragesprache verfügbar.

Koordinatensystem

Neptune hat einen Spatial Reference Identifier (SRID) für eine gesamte Datenbank. Die Homogenität des Koordinatensystems reduziert Benutzerfehler bei Abfragen und verbessert die Datenbankleistung. Die erste Version (1.4.7.0) unterstützt das kartesische Koordinatensystem, das auch als SRID 0 bezeichnet wird.

Die Neptun-Implementierung von SRID 0 ist mit Längen- und Breitengradwerten kompatibel. Wird verwendet`ST_DistanceSpheroid`, um Entfernungen auf der Grundlage von 4326 zu berechnen. WGS84/SRID

Die aktuelle Implementierung unterstützt das Speichern von dreidimensionalen Koordinaten. Die Spatial Functions unterstützen derzeit nur die Verwendung der Koordinaten der X- und Y-Achse (2-dimensional). Die Koordinaten der Z-Achse werden derzeit von den verfügbaren räumlichen Funktionen nicht unterstützt.

Speichern von Standortdaten

Speichern Sie Positionsdaten auf Knoten und Kanten mithilfe des Eigenschaftstyps Geometrie. Erstellen Sie Geometriewerte im Well-Known Textformat (WKT), einer Standardmethode zur Darstellung geografischer Formen als Text. Um beispielsweise eine Punktposition zu speichern:

```
CREATE (n:airport {code: 'ATL', location: ST_GeomFromText('POINT (-84.4281 33.6367)')})
```

Bei der Arbeit mit geographischen Koordinaten steht das erste Argument (x) für den Längengrad und das zweite Argument (y) für den Breitengrad. Dies entspricht der in Geodatenbanken verwendeten Standardkoordinatenreihenfolge und der Norm ISO 19125.

> **Note**
>
> Neptune unterstützt jetzt einen neuen Datentyp namens „Geometrie“. Die Geometrieeigenschaft eines Knotens oder einer Kante kann mithilfe der Funktion aus einer WKT-Zeichenfolge erstellt werden. `ST_GeomFromText`
> Neptune speichert Punktdaten automatisch in einem speziellen räumlichen Index, um die Leistung der Funktionen Räumliche Typen zu verbessern. Wird beispielsweise `ST_Contains` verwendet, um die Punkte innerhalb eines Polygons zu finden, und wird durch den speziellen räumlichen Index beschleunigt.
> Wikipedia-Seite zur Well-Known Textdarstellung von Geometrie

Räumliche Daten werden in großen Mengen geladen

Geben Sie beim Massenladen von Daten den Geometrietyp in Ihrem CSV-Header an. Neptune analysiert WKT-Strings und erstellt die entsprechenden Geometrieeigenschaften:

```
: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)
```

Vollständige Informationen zum CSV-Format finden Sie unter OpenCypher-Bulk-Load-Format.

Abfrage von Geodaten

In den folgenden Abfragebeispielen wird anhand des Datasets air-routes veranschaulicht, wie Spatial Functions in Neptune verwendet werden kann.

Wenn Ihre Daten nicht über eine Geometrieeigenschaft, sondern über separate Eigenschaften für Breitengrad und Längengrad verfügen, können Sie sie bei der Abfrage in Punkte konvertieren. Finden Sie die 10 Flughäfen, die einem bestimmten Ort am nächsten liegen:

```
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
```

Wenn Sie bereits Standorte gespeichert haben, können `ST_Point` Sie diese Standortwerte direkt verwenden:

1. Festlegen der -Eigenschaft

```
MATCH (a:airport)
SET a.location = ST_GeomFromText('POINT (' + a.lon + ' ' + a.lat + ')')
```

2. Abfrage mit 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
```

Den Bolt-Treiber verwenden

Die meisten Abfragemethoden geben Geometriewerte als WKT-Zeichenfolgen zurück, die für Menschen lesbar sind. Wenn Sie den Bolt-Treiber verwenden, werden Geometriewerte aus Effizienzgründen im WKB-Format (Well-Known Binär) zurückgegeben. Konvertieren Sie WKB in Ihrer Anwendung in ein Geometrieobjekt:

```
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);
}
```

Geofunktionen

Die folgenden räumlichen Funktionen sind in Neptune OpenCypher für die Arbeit mit Geometriedatentypen verfügbar:

- ST_Point
- ST_ GeomFromText
- ST_ AsText
- ST_ GeometryType
- ST_Equals
- ST_Contains
- ST_Intersects
- ST_Distance
- ST_ DistanceSpheroid
- ST_Envelope
- ST_Buffer

ST_Point

ST_Point gibt einen Punkt aus den eingegebenen Koordinatenwerten zurück.

Syntax

```
ST_Point(x, y, z)
```

Argumente

- x- Ein Wert vom Datentyp DOUBLE PRECISION, der eine erste Koordinate darstellt.
- y- Ein Wert vom Datentyp DOUBLE PRECISION, der eine zweite Koordinate darstellt.
- z- (optional)

Reihenfolge koordinieren

Bei der Arbeit mit geographischen Koordinaten steht das erste Argument (x) für den Längengrad und das zweite Argument (y) für den Breitengrad. Dies entspricht der in räumlichen Datenbanken verwendeten Standardkoordinatenreihenfolge und der Norm 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
```

Gültige Koordinatenbereiche

Stellen Sie bei geographischen Daten sicher, dass die Koordinaten innerhalb gültiger Bereiche liegen:

- Längengrad (x): -180 bis 180
- Breitengrad (y): -90 bis 90

Koordinaten außerhalb dieser Bereiche werden zurückgegeben `NaN` (keine Zahl), wenn sie mit Entfernungsberechnungsfunktionen wie `ST_DistanceSpheroid` verwendet werden.

Rückgabetyp

GEOMETRIE des Subtyps PUNKT

Wenn x oder y null ist, wird null zurückgegeben.

Beispiele

Im Folgenden wird eine Punktgeometrie aus den Eingabekoordinaten konstruiert.

```
RETURN ST_Point(5.0, 7.0);
POINT(5 7)
```

ST_ GeomFromText

ST_ GeomFromText konstruiert ein Geometrieobjekt aus einer WKT-Darstellung (Wellknown Text) einer Eingabegeometrie.

Syntax

```
ST_GeomFromText(wkt_string)
```

Argumente

- `wkt_string`— Ein Wert vom Datentyp STRING, bei dem es sich um eine WKT-Darstellung einer Geometrie handelt.

Rückgabetyp

GEOMETRY

Wenn wkt_string null ist, wird null zurückgegeben.

Wenn wkt_string nicht gültig ist, wird a zurückgegeben. BadRequestException

Beispiele

```
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 gibt die Well-Known-Text-Repräsentation (WKT) einer Eingabegeometrie zurück.

Syntax

```
ST_AsText(geo)
```

Argumente

- `geo`— Ein Wert vom Datentyp GEOMETRY oder ein Ausdruck, der als GEOMETRY ausgewertet wird.

Rückgabetyp

STRING

Wenn geo null ist, wird null zurückgegeben.

Wenn der Eingabeparameter keine Geometrie ist, BadRequestException wird a zurückgegeben.

Wenn das Ergebnis größer als eine 64-KB-ZEICHENFOLGE ist, wird ein Fehler zurückgegeben.

Beispiele

```
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 gibt den Typ der Geometrie als Zeichenfolge zurück.

Syntax

```
ST_GeometryType(geom)
```

Argumente

- `geom`- Ein Wert vom Datentyp GEOMETRY oder ein Ausdruck, der zu einem GEOMETRY-Typ ausgewertet wird.

Rückgabetyp

STRING

Wenn geom null ist, wird null zurückgegeben.

Wenn der Eingabeparameter keine Geometrie ist, BadRequestException wird a zurückgegeben.

Beispiele

```
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 gibt true zurück, wenn die 2D-Projektionen der Eingabegeometrien topologisch identisch sind. Geometrien gelten als topologisch gleich, wenn sie gleiche Punktsätze haben. Bei topologisch

gleichen Geometrien kann die Reihenfolge der Scheitelpunkte unter Beibehaltung dieser Gleichheit unterschiedlich sein.

Syntax

```
ST_Equals(geom1, geom2)
```

Argumente

- geom1— Ein Wert vom Datentyp GEOMETRY oder ein Ausdruck, der zu einem GEOMETRY-Typ ausgewertet wird.
- geom2- Ein Wert vom Datentyp GEOMETRY oder ein Ausdruck, der zu einem GEOMETRY-Typ ausgewertet wird. Dieser Wert wird mit geom1 verglichen, um festzustellen, ob er gleich geom1 ist.

Rückgabetyp

BOOLEAN

Wenn geom1 oder geom2 null ist, wird null zurückgegeben.

Wenn geom1 oder geom2 keine Geometrien sind, wird a zurückgegeben. BadRequestException

Beispiele

```
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
```

Im Folgenden wird geprüft, ob die beiden Linienfolgen geometrisch identisch sind.

```
RETURN ST_Equals(
    ST_GeomFromText('LINESTRING (1 0, 10 0)'),
    ST_GeomFromText('LINESTRING(1 0,5 0,10 0)'));
true
```

ST_Contains

ST_Contains gibt true zurück, wenn die 2D-Projektion der ersten Eingabegeometrie die 2D-Projektion der zweiten Eingabegeometrie enthält. Geometrie A enthält Geometrie B, wenn jeder Punkt in B

ein Punkt in A ist und ihre Innenräume einen nicht leeren Schnittpunkt haben. ST_Contains (A, B) entspricht ST_Within (B, A).

Syntax

```
ST_Contains(geom1, geom2)
```

Argumente

- geom1— Ein Wert vom Typ GEOMETRY oder ein Ausdruck, der zu einem GEOMETRY-Typ ausgewertet wird.
- geom2- Ein Wert vom Typ GEOMETRY oder ein Ausdruck, der zu einem GEOMETRY-Typ ausgewertet wird. Dieser Wert wird mit geom1 verglichen, um festzustellen, ob er in geom1 enthalten ist.

Rückgabetyp

BOOLEAN

Wenn geom1 oder geom2 null ist, wird null zurückgegeben.

Wenn der Eingabeparameter keine Geometrie ist, BadRequestException wird a zurückgegeben.

Beispiele

```
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 gibt true zurück, wenn die 2D-Projektionen der beiden Eingabegeometrien mindestens einen gemeinsamen Punkt haben.

Syntax

```
ST_Intersects(geom1, geom2)
```

Argumente

- `geom1`- Ein Wert vom Datentyp GEOMETRY oder ein Ausdruck, der zu einem GEOMETRY-Typ ausgewertet wird.
- `geom2`- Ein Wert vom Datentyp GEOMETRY oder ein Ausdruck, der zu einem GEOMETRY-Typ ausgewertet wird.

Rückgabetyp

BOOLEAN

Wenn geom1 oder geom2 null ist, wird null zurückgegeben.

Wenn der Eingabeparameter keine Geometrie ist, BadRequestException wird a zurückgegeben.

Beispiele

```
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

Bei Eingabegeometrien gibt ST_Distance den minimalen euklidischen Abstand zwischen den 2D-Projektionen der beiden Eingabegeometriewerte zurück.

Syntax

```
ST_Distance(geo1, geo2)
```

Argumente

- `geo1`- Ein Wert vom Datentyp GEOMETRY oder ein Ausdruck, der zu einem GEOMETRY-Typ ausgewertet wird.
- `geo2`- Ein Wert vom Datentyp GEOMETRY oder ein Ausdruck, der als GEOMETRY ausgewertet wird.

Rückgabetyp

DOPPELTE GENAUIGKEIT in denselben Einheiten wie die Eingabegeometrien.

Wenn geo1 oder geo2 Null ist, wird Null zurückgegeben.

Wenn der Eingabeparameter keine Geometrie ist, BadRequestException wird a zurückgegeben.

Beispiele

```
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

Gibt den Mindestabstand in Metern zwischen zwei lon/lat Geometrien zurück. Das Sphäroid ist 4326 WGS84/SRID .

Syntax

```
ST_DistanceSpheroid(geom1, geom2);
```

Argumente

- geom1— Ein Wert vom Datentyp GEOMETRY oder ein Ausdruck, der zu einem GEOMETRY-Typ ausgewertet wird.
- geom2- Ein Wert vom Datentyp GEOMETRY oder ein Ausdruck, der zu einem GEOMETRY-Typ ausgewertet wird.

Rückgabetyp

FLOAT

Wenn geom null ist, wird null zurückgegeben.

Beispiele

```
RETURN ST_DistanceSpheroid(
    ST_GeomFromText('POINT(-110 42)'),
    ST_GeomFromText('POINT(-118 38)'))
814278.77
```

ST_Envelope

ST_Envelope gibt den minimalen Begrenzungsrahmen der Eingabegeometrie wie folgt zurück:

- Wenn die Eingabegeometrie leer ist, ist die zurückgegebene Geometrie POINT EMPTY.
- Wenn der minimale Begrenzungsrahmen der Eingabegeometrie zu einem Punkt degeneriert wird, ist die zurückgegebene Geometrie ein Punkt.
- Wenn keine der obigen Bedingungen zutrifft, gibt die Funktion ein gegen den Uhrzeigersinn ausgerichtetes Polygon zurück, dessen Eckpunkte die Ecken des minimalen Begrenzungsrahmens sind.

Bei allen nicht leeren Eingaben wird die Funktion auf der 2D-Projektion der Eingabegeometrie ausgeführt.

Syntax

```
ST_Envelope(geom)
```

Argumente

- `geom`- Ein Wert vom Datentyp GEOMETRY oder ein Ausdruck, der zu einem GEOMETRY-Typ ausgewertet wird.

Rückgabetyp

GEOMETRY

Wenn geom null ist, wird null zurückgegeben.

Beispiele

```
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 gibt 2D-Geometrie zurück, die alle Punkte darstellt, deren Abstand von der auf die XY-kartesische Ebene projizierten Eingabegeometrie kleiner oder gleich dem Eingabeabstand ist.

Syntax

```
ST_Buffer(geom, distance, number_of_segments_per_quarter_circle)
```

Argumente

- `geom`- Ein Wert vom Datentyp GEOMETRY oder ein Ausdruck, der zu einem GEOMETRY-Typ ausgewertet wird.
- `distance`— Ein Wert vom Datentyp DOUBLE PRECISION, der die Entfernung (oder den Radius) des Puffers darstellt.
- `number_of_segments_per_quarter_circle`- Ein Wert vom Datentyp INTEGER (sollte größer oder gleich 0 sein). Dieser Wert bestimmt die Anzahl der Punkte, die einem Viertelkreis um jeden Scheitelpunkt der Eingabegeometrie annähern sollen. Negative Werte sind standardmäßig auf null. Der Standardwert ist 8.

Rückgabetyp

GEOMETRY

Die Funktion ST_Buffer gibt zweidimensionale (2D) Geometrie in der xy-kartesischen Ebene zurück.

Beispiele

```
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))
```

Im Folgenden wird der Puffer der Eingabepunktgeometrie zurückgegeben, der sich einem Kreis annähert. Da der Befehl 3 als Anzahl der Segmente pro Viertelkreis angibt, verwendet die Funktion drei Segmente, um den Viertelkreis anzunähern.

```
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))
```

Zugriff auf das Neptune-Diagramm mit SPARQL

SPARQL ist eine Abfragesprache für das Resource Description Framework (RDF), ein für das Web entwickelts Datenformat für Diagramme. Amazon Neptune ist mit SPARQL 1.1 kompatibel. Das bedeutet, dass Sie eine Verbindung zu einer Neptune-DB-Instance herstellen und das Diagramm mittels der in der Spezifikation SPARQL 1.1 Query Language beschriebenen Abfragesprache abfragen können.

Eine Abfrage in SPARQL besteht aus einer `SELECT`-Klausel zur Angabe der Variablen, die zurückgegeben werden sollen, und einer `WHERE`-Klausel, um anzugeben, welche Daten im Diagramm abgeglichen werden sollen. Wenn Sie noch keine Erfahrungen mit SPARQL-Abfragen haben, lesen Sie den Abschnitt Writing Simple Queries in SPARQL 1.1 Query Language.

> **Important**
>
> Zum Laden eines kleinen Datensatzes mag sich `SPARQL UPDATE INSERT` gut eignen, wenn Sie jedoch eine beträchtliche Menge an Daten aus einer Datei laden müssen, siehe Verwenden des Amazon Neptune Bulk Loaders zur Datenaufnahme.

Weitere Hinweise zu den Besonderheiten der Neptune-SPARQL-Implementierung finden Sie unter Einhaltung von SPARQL-Standards.

Sie benötigen Folgendes, um starten zu können:

- Eine Neptune-DB-Instance. Informationen zum Erstellen einer Neptune-DB-Instance finden Sie unter Einen Amazon Neptune Neptune-Cluster erstellen.
- Eine Amazon–EC2-Instance in derselben Virtual Private Cloud (VPC), in der sich auch Ihre Neptune-DB-Instance befindet.

Themen

- Herstellen einer Verbindung mit einer Neptune-DB-Instance über die RDF4J-Konsole
- Herstellen einer Verbindung mit einer Neptune-DB-Instance über RDF4J Workbench
- Herstellen von Verbindungen mit einer Neptune-DB-Instance über Java
- SPARQL HTTP-API
- SPARQL-Abfragehinweise
- Verhalten von SPARQL DESCRIBE in Bezug auf das Standarddiagramm
- SPARQL-Abfragestatus-API
- SPARQL-Abfrageabbruch
- Verwenden des SPARQL 1.1-Graph-Store-Protokolls (GSP) über HTTP in Amazon Neptune
- Analysieren der Ausführung Neptune Neptune-Abfragen mit SPARQL erklären
- SPARQL-Verbundabfragen in Neptune mit der SERVICE-Erweiterung

Herstellen einer Verbindung mit einer Neptune-DB-Instance über die RDF4J-Konsole

Über die RDF4J-Konsole können Sie mit Resource Description Framework (RDF)-Graphen und -Abfragen in einer REPL (Read-Eval-Print Loop)-Umgebung experimentieren.

Sie können eine Remote-Graph-Datenbank als Repository hinzufügen und dieses über die RDF4J-Konsole abfragen. Dieser Abschnitt führt Sie durch die Konfiguration der RDF4J-Konsole zum Herstellen einer Remote-Verbindung mit einer Neptune-DB-Instance.

Herstellen einer Verbindung mit Neptune über die RDF4J-Konsole

1. Laden Sie das RDF4J-SDK von der Download-Seite der RDF4J-Website herunter.

2. Entpacken Sie die ZIP-Datei des RDF4J-SDKs.
3. Navigieren Sie in einem Terminalfenster zum RDF4J-SDK-Verzeichnis und geben Sie den folgenden Befehl ein, um die RDF4J-Konsole auszuführen.

```
bin/console.sh
```

Die Ausgabe sollte folgendermaßen oder ähnlich aussehen:

```
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.
>
```

Sie sehen nun die >-Eingabeaufforderung. Dies ist die allgemeine Eingabeaufforderung der RDF4J-Konsole. Verwenden Sie diese zum Einrichten von Repositorys und anderen Vorgängen. Ein Repository verfügt über eine eigene Eingabeaufforderung zum Ausführen von Abfragen.

4. Geben Sie an der >-Eingabeaufforderung Folgendes ein, um ein SPARQL-Repository für Ihre Neptune-DB-Instance zu erstellen:

```
create sparql
```

5. Sie werden über die RDF4J-Konsole zur Angabe von Werten für Variablen gebeten, die für eine Verbindung zum SPARQL-Endpunkt erforderlich sind.

```
Please specify values for the following variables:
```

Geben Sie die folgenden Werte an:

Variablenbezeichnung	Wert
SPARQL-Abfrageendpunkt	`https://`*`your-neptune-endpoint`* `:`*`port`*`/sparql`

SPARQL-Aktualisierungsendpunkt	`https://your-neptune-endpoint :port/sparql`
ID des lokalen Repositorys [endpoint @localhost]	`neptune`
Repository-Titel [SPARQL endpoint repository @localhost]	`Neptune DB instance`

Informationen zum Ermitteln der Adresse Ihrer Neptune-DB-Instance finden Sie im Abschnitt Verbinden mit Amazo-Neptune-Endpunkten.

Wenn die Operation erfolgreich ausgeführt wurde, sehen Sie die folgende Meldung:

```
Repository created
```

6. Geben Sie an der >-Eingabeaufforderung Folgendes ein, um eine Verbindung zur Neptune-DB-Instance herzustellen:

```
open neptune
```

Wenn die Operation erfolgreich ausgeführt wurde, sehen Sie die folgende Meldung:

```
Opened repository 'neptune'
```

Sie sehen nun die `neptune`>-Eingabeaufforderung. Über diese Eingabeaufforderung können Sie das Neptune-Diagramm abfragen.

> **Note**
>
> Nach dem Hinzufügen des Repositorys können Sie bei der nächsten Ausführung von `bin/console.sh` sofort den Befehl `open neptune` ausführen, um eine Verbindung zur Neptune-DB-Instance herzustellen.

7. Geben Sie an der `neptune`>-Eingabeaufforderung Folgendes ein, um eine SPARQL-Abfrage auszuführen, die bis zu 10 der Triples (subject-predicate-object) im Diagramm zurückgibt, indem Sie die `?s ?p ?o`-Abfrage mit einem Grenzwert von 10 ausführen. Ersetzen Sie den Text nach dem Befehl `sparql` durch eine andere SPARQL-Abfrage, wenn Sie etwas anderes abfragen möchten.

```
sparql select ?s ?p ?o where {?s ?p ?o} limit 10
```

Herstellen einer Verbindung mit einer Neptune-DB-Instance über RDF4J Workbench

Dieser Abschnitt führt Sie durch die Herstellung einer Verbindung mit einer Amazon-Neptune-DB-Instance über RDF4J Workbench und RDF4J Server. RDF4J Server ist als Proxy zwischen dem Neptune-SPARQL-HTTP-REST-Endpunkt von Neptune und RDF4J Workbench erforderlich.

RDF4J Workbench stellt eine einfache Schnittstelle für das Experimentieren mit einem Diagramm bereit, einschließlich des Ladens lokaler Dateien. Weitere Informationen finden Sie unter Abschnitt hinzufügen in der RDF4J-Dokumentation.

Voraussetzungen

Bevor Sie beginnen, führen Sie die folgenden Schritte aus:

- Installieren Sie Java 1.8 oder höher.
- Installieren Sie RDF4J Server und RDF4J Workbench. Weitere Informationen finden Sie unter Installieren von RDF4J Server und RDF4J Workbench.

Verwenden von RDF4J Workbench zum Herstellen von Verbindungen mit Neptune

1. Navigieren Sie in einem Webbrowser zu der URL, über die die RDF4J Workbench-Web-App bereitgestellt wird. Wenn Sie beispielsweise Apache Tomcat verwenden, lautet die URL: https://*ec2_hostname*: 8080/rdf4j -workbench/.
2. Wenn die Aufforderung Connect to RDF4J Server (mit RDF4J Server verbinden) angezeigt wird, überprüfen Sie, ob RDF4J Server installiert ist, ausgeführt wird und ob die Server-URL korrekt ist. Fahren Sie dann mit dem nächsten Schritt fort.
3. Wählen Sie im linken Bereich New repository (Neues Repository) aus.

Unter New repository (Neues Repository):

- Wählen Sie aus der Dropdown-Liste Type (Typ) SPARQL endpoint proxy (SPARQL-Endpunkt-Proxy) aus.
- Geben Sie als ID neptune ein.
- Geben Sie Neptune-DB-Instance als Titel ein.

Wählen Sie Weiter aus.

4. Unter New repository (Neues Repository):

   - Geben Sie für SPARQL query endpoint URL (SPARQL-Abfrage-Endpunkt-URL) `https://`*`your-neptune-endpoint`*`:`*`port`*`/sparql` ein.
   - Geben Sie für SPARQL update endpoint URL (SPARQL-Aktualisierungs-Endpunkt-URL) `https://`*`your-neptune-endpoint`*`:`*`port`*`/sparql` ein.

   Informationen zum Ermitteln der Adresse Ihrer Neptune-DB-Instance finden Sie im Abschnitt Verbinden mit Amazo-Neptune-Endpunkten.

   Wählen Sie Erstellen aus.

5. Das neptune-Repository wird nun in der Liste der Repositorys aufgeführt. Es kann ein paar Minuten dauern, ehe Sie das neue Repository verwenden können.
6. Wählen Sie in der Spalte Id (ID) der Tabelle den Link neptune aus.
7. Wählen Sie im linken Bereich Query (Abfrage) aus.

> **Note**
>
> Wenn die Menüelemente unter Explore (Erkunden) deaktiviert sind, müssen Sie möglicherweise die Verbindung zu RDF4J Server wiederherstellen und das Repository neptune erneut auswählen.
> Sie können dies über die [change] [ändern]-Links [Ändern] in der oberen rechten Ecke durchführen.

8. Geben Sie im Abfragefeld die folgende SPARQL-Abfrage ein und wählen Sie dann Execute (Ausführen) aus.

```
select ?s ?p ?o where {?s ?p ?o} limit 10
```

Das vorherige Beispiel gibt bis zu 10 der Triples (subject-predicate-object) im Graph zurück, indem Sie die `?s ?p ?o`-Abfrage mit einem Grenzwert von 10 ausführen.

Herstellen von Verbindungen mit einer Neptune-DB-Instance über Java

Dieser Abschnitt führt Sie durch die Ausführung eines vollständigen Java-Beispiels, in dem eine Verbindung mit einer Amazon-Neptune-DB-Instance hergestellt wird und eine SPARQL-Abfrage ausgeführt wird.

Befolgen Sie diese Anweisungen für eine Amazon-EC2-Instance befolgen, die sich in derselben Virtual Private Cloud (VPC) wie Ihre Neptune-DB-Instance befindet.

Herstellen einer Verbindung mit Neptune über Java

1. Installieren Sie Apache Maven auf Ihrer EC2-Instance. Wenn Sie Amazon Linux 2023 (bevorzugt) verwenden, verwenden Sie:

   ```
   sudo dnf update -y
   sudo dnf install maven -y
   ```

   Wenn Sie Amazon Linux 2 verwenden, laden Sie die neueste Binärdatei herunter von 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. Dieses Beispiel wurde lediglich mit Java 8 getestet. Geben Sie Folgendes ein, um Java 8 auf Ihrer EC2-Instance zu installieren:

```
sudo yum install java-1.8.0-devel
```

3. Geben Sie Folgendes ein, um Java 8 als Standard-Laufzeit Ihrer EC2-Instance festzulegen:

```
sudo /usr/sbin/alternatives --config java
```

Wenn Sie dazu aufgefordert werden, geben Sie die Nummer für Java 8 ein.

4. Geben Sie Folgendes ein, um Java 8 als Standard-Compiler Ihrer EC2-Instance festzulegen:

```
sudo /usr/sbin/alternatives --config javac
```

Wenn Sie dazu aufgefordert werden, geben Sie die Nummer für Java 8 ein.

5. Erstellen Sie in einem neuen Verzeichnis eine `pom.xml`-Datei und öffnen Sie diese in einem Text-Editor.

6. Kopieren Sie Folgendes in die Datei `pom.xml` und speichern Sie diese. (Sie können die Versionsnummern in der Regel in die neueste stabile Version ändern):

```
<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**
>
> Wenn Sie ein vorhandenes Maven-Projekt ändern, ist die erforderliche Abhängigkeit im vorhergehenden Code hervorgehoben.

7. Zum Erstellen von Unterverzeichnissen für den Beispielquellcode (`src/main/java/com/amazonaws/`), geben Sie Folgendes in die Befehlszeile ein:

```
mkdir -p src/main/java/com/amazonaws/
```

8. Erstellen Sie im Verzeichnis `src/main/java/com/amazonaws/` eine Datei namens `App.java` und öffnen Sie diese dann in einem Text-Editor.
9. Kopieren Sie Folgendes in die `App.java`-Datei. *`your-neptune-endpoint`*Ersetzen Sie durch die Adresse Ihrer Neptune-DB-Instance.

> **Note**
>
> Informationen zum Ermitteln des Hostnamens Ihrer Neptune-DB-Instance finden Sie im Abschnitt Verbinden mit Amazo-Neptune-Endpunkten.

```
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. Kompilieren und führen Sie das Beispiel mit folgendem Maven-Befehl aus:

```
mvn compile exec:java
```

Das vorherige Beispiel gibt bis zu 10 der Triples (subject-predicate-object) im Graph zurück, indem Sie die `?s ?p ?o`-Abfrage mit einem Grenzwert von 10 ausführen. Um etwas anderes abzufragen, ersetzen Sie die Abfrage durch eine andere SPARQL-Abfrage.

Die Iteration der Ergebnisse des Beispiels gibt die Werte alle zurückgegebenen Variablen aus. Das `Value`-Objekt wird in einen `String` konvertiert und ausgegeben. Wenn Sie den `SELECT`-Teil der Abfrage ändern, müssen Sie den Code ändern.

SPARQL HTTP-API

SPARQL HTTP-Anforderungen werden am folgenden Endpunkt akzeptiert: `https://`*`your-neptune-endpoint`*`:`*`port`*`/sparql`

Weitere Informationen zum Herstellen einer Verbindung mit Amazon Neptune über SPARQL finden Sie unter Zugriff auf das Neptune-Diagramm mit SPARQL.

Weitere Informationen über das SPARQL-Protokoll und die SPARQL-Abfragesprache finden Sie in den Spezifikationen SPARQL 1.1 Protocol und SPARQL 1.1 Query Language.

In den folgenden Themen finden Sie Informationen zu SPARQL-RDF-Serialisierungsformaten und zur Verwendung der SPARQL-HTTP-API mit Neptune.

Inhalt

- Herstellen von Verbindungen mit einer Neptune-DB-Instance über den HTTP-REST-Endpunkt
- Optionale HTTP-Trailing-Header für mehrteilige SPARQL-Antworten
- Von SPARQL in Neptune verwendete RDF-Medientypen
  - Von Neptune SPARQL verwendete RDF-Serialisierungsformate
  - SPARQL-Ergebnisserialisierungsformate, die von Neptune SPARQL verwendet werden
  - Media-Types das Neptune verwenden kann, um RDF-Daten zu importieren
  - Media-Types das Neptune zum Exportieren von Abfrageergebnissen verwenden kann
- Verwenden von SPARQL UPDATE LOAD zum Importieren von Daten in Neptune

- Verwenden von SPARQL UPDATE UNLOAD zum Löschen von Daten aus Neptune

Herstellen von Verbindungen mit einer Neptune-DB-Instance über den HTTP-REST-Endpunkt

> **Note**
>
> Neptune unterstützt HTTP/2 derzeit keine REST-API-Anfragen. Clients müssen dies verwenden, HTTP/1.1 wenn sie eine Verbindung zu Endpunkten herstellen.

Die folgenden Anweisungen führen Sie durch das Herstellen einer Verbindung zum SPARQL-Endpunkt mittels des curl-Befehls und HTTPS. Dabei wird mit HTTP-Syntax eine Verbindung über HTTPS hergestellt. Befolgen Sie diese Anweisungen für eine Amazon-EC2-Instance befolgen, die sich in derselben Virtual Private Cloud (VPC) wie Ihre Neptune-DB-Instance befindet.

Der HTTP-Endpunkt für SPARQL-Abfragen für eine Neptune-DB-Instance ist `https://`*`your-neptune-endpoint`*`:`*`port`*`/sparql`.

> **Note**
>
> Informationen zum Ermitteln des Hostnamens Ihrer Neptune-DB-Instance finden Sie im Abschnitt Verbinden mit Amazo-Neptune-Endpunkten.

Amazon Neptune stellt einen HTTP-Endpunkt für SPARQL-Abfragen bereit. Die REST-Schnittstelle ist mit der SPARQL-Version 1.1. kompatibel.

ABFRAGE über HTTP POST

Im folgenden Beispiel wird curl zum Übermitteln einer SPARQL-**`QUERY`** mit HTTP POST verwendet.

```
curl -X POST --data-binary 'query=select ?s ?p ?o where {?s ?p ?o} limit 10'
 https://your-neptune-endpoint:port/sparql
```

Das vorherige Beispiel gibt bis zu 10 der Triples (subject-predicate-object) im Graph zurück, indem Sie die `?s ?p ?o`-Abfrage mit einem Grenzwert von 10 ausführen. Um etwas anderes abzufragen, ersetzen Sie diese durch eine andere SPARQL-Abfrage.

> **Note**
>
> Der Standard-MIME-Medientyp einer Antwort ist `application/sparql-results+json` für `SELECT`- und `ASK`-Abfragen.
> Der Standard-MIME-Typ einer Antwort ist `application/n-quads` für `CONSTRUCT`- und `DESCRIBE`-Abfragen.
> Eine Liste der von Neptune für die Serialisierung verwendeten Medientypen finden Sie unter Von Neptune SPARQL verwendete RDF-Serialisierungsformate.

UPDATE mit HTTP POST

Im folgenden Beispiel wird curl zum Übermitteln einer SPARQL-**`UPDATE`** mit HTTP POST verwendet.

```
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
```

Im vorherigen Beispiel wird das folgende Triple in das standardmäßige SPARQL-Diagramm eingefügt: `<https://test.com/s> <https://test.com/p> <https://test.com/o>`

Optionale HTTP-Trailing-Header für mehrteilige SPARQL-Antworten

Die HTTP-Antwort auf SPARQL-Abfragen und -Aktualisierungen wird häufig in mehr als einem Teil oder Block zurückgegeben. Es kann schwierig sein, einen Fehler zu diagnostizieren, der auftritt, nachdem eine Abfrage oder eine Aktualisierung mit dem Senden dieser Blöcke begonnen hat, vor allem, da der erste Block mit dem HTTP-Statuscode `200` eingeht.

Wenn Sie nicht ausdrücklich Trailing-Header anfordern, meldet Neptune einen solchen Fehler nur durch Anfügen einer Fehlermeldung an den Nachrichtentext, der in der Regel beschädigt ist.

Um die Erkennung und Diagnose solcher Probleme zu vereinfachen, können Sie Ihrer Anforderung einen Transfer-Encoding (TE)-Trailer-Header hinzufügen (siehe z. B. die MDN-Seite zu TE-Anforderungs-Headern). Anschließend fügt Neptune zwei neue Header-Felder in die Trailing-Header der Antwortblöcke ein:

- `X-Neptune-Status` – enthält den Antwortcode gefolgt von einem Kurznamen. Im Erfolgsfall wäre der nachgestellte Header beispielsweise: `X-Neptune-Status: 200 OK`. Bei einem Fehler wäre der Antwortcode ein Neptune-Engine-Fehlercode wie `X-Neptune-Status: 500 TimeLimitExceededException`.

- `X-Neptune-Detail` – ist bei erfolgreichen Anforderungen leer. Im Fehlerfall ist die JSON-Fehlermeldung enthalten. Da in HTTP-Header-Werten nur ASCII-Zeichen zulässig sind, ist die JSON-Zeichenfolge URL-codiert. Die Fehlermeldung wird weiter an die Antwortmeldung angefügt.

Von SPARQL in Neptune verwendete RDF-Medientypen

Resource Description Framework (RDF)-Daten können auf viele verschiedene Arten serialisiert werden, von denen SPARQL die meisten nutzen oder ausgeben kann:

Von Neptune SPARQL verwendete RDF-Serialisierungsformate

- RDF/XML— XML-Serialisierung von RDF, definiert in der RDF 1.1 XML-Syntax. Medienntyp: `application/rdf+xml`. Typische Dateierweiterung: `.rdf`.
- N-Triples— Ein zeilenbasiertes Klartextformat zur Kodierung eines RDF-Graphen, definiert in RDF 1.1. N-Triples Medientyp: `application/n-triples`, `text/turtle` oder `text/plain`. Typische Dateierweiterung: `.nt`.
- N-Quads— Ein zeilenbasiertes Klartextformat zur Kodierung eines RDF-Graphen, definiert in RDF 1.1. N-Quads Es ist eine Erweiterung von. N-Triples Medientyp:`application/n-quads`, oder `text/x-nquads` wenn mit US-ASCII 7-Bit codiert. Typische Dateierweiterung: `.nq`.
- Turtle – Eine Textsyntax für RDF, definiert in RDF 1.1 c Turtle, mit der ein RDF-Diagramm vollständig in einem kompakten und natürlichen Textformat geschrieben werden kann, einschließlich Abkürzungen für häufige Nutzungsmuster und Datentypen. Turtle bietet ein gewisses Maß an Kompatibilität mit dem N-Triples Format sowie der Dreifachmuster-Syntax von SPARQL. Medientyp: `text/turtle`Typische Dateierweiterung: `.ttl`.
- TriG – Eine Textsyntax für RDF, definiert in RDF 1.1 – TriG, mit der ein RDF-Diagramm vollständig in einem kompakten und natürlichen Textformat geschrieben werden kann, einschließlich Abkürzungen für häufige Nutzungsmuster und Datentypen. TriG ist eine Erweiterung des Turtle-Formats. Medienntyp: `application/trig`. Typische Dateierweiterung: `.trig`.
- N3 (Notation3) – Eine Zusicherungs- und Logiksprache, definiert in Notation3 (N3): Eine lesbare RDF-Syntax. N3 erweitert das RDF-Datenmodell um Formeln (Literale, die selbst Graphen sind), Variablen, logische Implikation und funktionale Prädikate und bietet eine textuelle Syntaxalternative zu. RDF/XML Medienntyp: `text/n3`. Typische Dateierweiterung: `.n3`.
- JSON-LD— Ein in 1.0 definiertes Datenserialisierungs- und Nachrichtenformat. Medientyp:. JSON-LD `application/ld+json` Typische Dateierweiterung: `.jsonld`.
- TriX – Eine Serialisierung von RDF in XML, definiert in TriX: RDF Triples in XML. Medienntyp: `application/trix`. Typische Dateierweiterung: `.trix`.

- SPARQL JSON Results – Eine Serialisierung von RDF im SPARQL 1.1-JSON-Format für Abfrageergebnisse. Medienntyp: `application/sparql-results+json`. Typische Dateierweiterung: `.srj`.
- RDF4J-Binärformat – Ein Binärformat zum Kodieren von RDF-Daten, dokumentiert in RDF4J-RDF-Binärformat. Medienntyp: `application/x-binary-rdf`.

SPARQL-Ergebnisserialisierungsformate, die von Neptune SPARQL verwendet werden

- SPARQL XML Results – Ein XML-Format für Variablenbindung und boolesche Ergebnisformate, bereitgestellt von der SPARQL-Abfragesprache, definiert in SPARQL-XML-Format für Abfrageergebnisse (zweite Ausgabe). Medienntyp: `application/sparql-results+xml`. Typische Dateierweiterung: `.srx`.
- SPARQL CSV and TSV Results – Verwendung von durch Komma und Tabulator getrennten Werten, um SPARQL-Abfrageergebnisse aus `SELECT`-Abfragen auszudrücken, definiert in SPARQL 1.1-CSV- und TSV-Formate für Abfrageergebnisse. Medientyp: `text/csv` für durch Komma getrennte Werte und `text/tab-separated-values` für durch Tabulator getrennte Werte. Typische Dateierweiterungen: `.csv` für durch Komma getrennte Werte und `.tsv` für durch Tabulator getrennte Werte.
- Binary Results Table – Ein Binärformat zum Kodieren der Ausgabe von SPARQL-Abfragen. Medienntyp: `application/x-binary-rdf-results-table`.
- SPARQL JSON Results – Eine Serialisierung von RDF im SPARQL 1.1-JSON-Format für Abfrageergebnisse. Medienntyp: `application/sparql-results+json`.

Media-Types das Neptune verwenden kann, um RDF-Daten zu importieren

Media-types unterstützt von der Neptune Massengutlader

- N-Triples
- N-Quads
- RDF/XML
- Turtle

Media-types das SPARQL UPDATE LOAD importieren kann

- N-Triples

- N-Quads
- RDF/XML
- Turtle
- TriG
- N3
- JSON-LD

Media-Types das Neptune zum Exportieren von Abfrageergebnissen verwenden kann

Um ein Ausgabeformat für eine SPARQL-Abfrageantwort anzugeben, senden Sie einen "`Accept: media-type`"-Header mit der Abfrageanforderung. Beispiel:

```
curl -H "Accept: application/nquads" ...
```

RDF-Medientypen, die SPARQL SELECT über Neptune ausgeben kann

- SPARQL-JSON-Ergebnisse (Standardwert)
- SPARQL-XML-Ergebnisse
- Binäre Ergebnistabelle (Medientyp: `application/x-binary-rdf-results-table`)
- Comma-Separated Werte (CSV)
- Tab-Separated Werte (TSV)

RDF-Medien, die SPARQL ASK über Neptune ausgeben kann

- SPARQL-JSON-Ergebnisse (Standardwert)
- SPARQL-XML-Ergebnisse
- Boolesch (Medientyp: `text/boolean`, mit der Bedeutung „true“ oder „false“)

RDF-Medientypen, die SPARQL CONSTRUCT über Neptune ausgeben kann

- N-Quads(Dies ist die Standardeinstellung)
- RDF/XML
- JSON-LD
- N-Triples

- Turtle
- N3
- TriX
- TriG
- SPARQL-JSON-Ergebnisse
- Binäres RDF4J-RDF-Format

RDF-Medientypen, die SPARQL DESCRIBE über Neptune ausgeben kann

- N-Quads(Dies ist die Standardeinstellung)
- RDF/XML
- JSON-LD
- N-Triples
- Turtle
- N3
- TriX
- TriG
- SPARQL-JSON-Ergebnisse
- Binäres RDF4J-RDF-Format

Verwenden von SPARQL UPDATE LOAD zum Importieren von Daten in Neptune

Die Syntax des SPARQL UPDATE LOAD-Befehls ist in der SPARQL 1.1-Aktualisierungsempfehlung angegeben:

```
LOAD SILENT (URL of data to be loaded) INTO GRAPH (named graph into which to load the
 data)
```

- **`SILENT`** – (Optional) Bewirkt, dass die Operation auch dann erfolgreich ist, wenn bei der Verarbeitung ein Fehler aufgetreten ist.

  Dies kann nützlich sein, wenn eine einzelne Transaktion mehrere Anweisungen enthält, z. B. `"LOAD ...; LOAD ...; UNLOAD ...; LOAD ...;"`, und die Transaktion abgeschlossen werden soll, auch wenn einige der entfernten Daten nicht verarbeitet werden konnten.

- *`URL of data to be loaded`*— (Erforderlich) Gibt eine Remote-Datendatei an, die Daten enthält, die in ein Diagramm geladen werden sollen.

  Die Remote-Datei muss eine der folgenden Erweiterungen haben:

  - `.nt` für NTriples.
  - `.nq` für NQuads.
  - `.trig` für Trig.
  - `.rdf`für RDF/XML.
  - `.ttl` für Turtle.
  - `.n3` für N3.
  - `.jsonld`für JSON-LD.
- **`INTO GRAPH`***`(named graph into which to load the data)`*— (Optional) Gibt den Graphen an, in den die Daten geladen werden sollen.

  Neptune ordnet jedem Triple ein benanntes Diagramm zu. Sie können das benannte Standarddiagramm mithilfe des Fallback-URI (`http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph`) für das benannte Diagramm wie folgt angeben:

  ```
  INTO GRAPH <http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph>
  ```

> **Note**
>
> Wenn Sie viele Daten laden müssen, sollten Sie den Neptune-Massen-Loader anstelle von UPDATE LOAD verwenden. Weitere Informationen zum Massen-Loader finden Sie unter Verwenden des Amazon Neptune Bulk Loaders zur Datenaufnahme.

Sie können mit `SPARQL UPDATE LOAD` Daten direkt aus Amazon S3 oder aus Dateien laden, die von einem selbst gehosteten Webserver abgerufen wurden. Die Ressourcen, die geladen werden sollen, müssen sich in derselben Region wie der Neptune-Server befinden. Außerdem muss der Endpunkt für die Ressourcen in der VPC zugelassen worden sein. Informationen zum Erstellen eines Amazon-S3-Endpunkts finden Sie unter Erstellen eines Amazon-S3-VPC-Endpunkts.

Alle `SPARQL UPDATE LOAD`-URIs müssen mit `https://` beginnen. Dies umfasst Amazon-S3-URLs.

Im Gegensatz zum Neptune-Massen-Loader ist ein Aufruf an `SPARQL UPDATE LOAD` vollständig transaktional.

Laden von Dateien direkt aus Amazon S3 in Neptune über SPARQL UPDATE LOAD

Da Neptune die Übergabe einer IAM-Rolle an Amazon S3 bei Verwendung von SPARQL UPDATE LOAD nicht erlaubt, muss der betreffende Amazon-S3-Bucket öffentlich sein oder Sie müssen eine vorsignierte Amazon-S3-URL in der LOAD-Abfrage verwenden.

Um eine vorsignierte URL für eine Amazon S3 S3-Datei zu generieren, können Sie einen AWS CLI Befehl wie diesen verwenden:

```
aws s3 presign --expires-in (number of seconds) s3://(bucket name)/(path to file of
 data to load)
```

Anschließend können Sie die resultierende vorsignierte URL in Ihrem `LOAD`-Befehl verwenden:

```
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)'
```

Weitere Informationen finden Sie auf der Seite zum Thema Authentifizieren von Anforderungen: Verwenden von Abfrageparametern. Die Boto3-Dokumentation zeigt, wie ein Python-Skript verwendet wird, um eine vorsignierte URL zu generieren.

Außerdem muss der Inhaltstyp der Dateien, die geladen werden sollen, korrekt eingestellt sein.

1. Sie legen den Inhaltstyp von Dateien beim Hochladen in Amazon S3 mithilfe des Parameters `-metadata` wie folgt fest:

   ```
   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. Vergewissern Sie sich, dass die Medientyp-Informationen tatsächlich vorhanden sind. Führen Sie Folgendes aus:

   ```
   curl -v bucket-name/folder-name
   ```

Die Ausgabe dieses Befehls sollte die Medientyp-Informationen anzeigen, die Sie beim Hochladen der Dateien festlegen.

3. Anschließend können Sie diese Dateien mit dem Befehl `SPARQL UPDATE LOAD` in Neptune importieren:

```
curl https://your-neptune-endpoint:port/sparql \
  -d "update=LOAD <https://s3.amazonaws.com/bucket-name/my-rdf-input/test.rdf>"
```

Die Schritte oben funktionieren nur für einen öffentlichen Amazon-S3-Bucket oder für einen Bucket, auf den Sie mit einer vorsignierten Amazon-S3-URL in der LOAD-Abfrage zugreifen.

Sie können auch einen Web-Proxy-Server einrichten, um Dateien aus einem privaten Amazon-S3-Bucket zu laden, wie unten gezeigt:

Verwenden eines Webservers zum Laden von Dateien in Neptune über SPARQL UPDATE LOAD

1. Installieren Sie einen Webserver auf einem in der VPC ausgeführten Computer, auf dem Neptune gehostet wird. Die Dateien werden dann geladen. Bei Verwendung von Amazon Linux könnten Sie Apache beispielsweise wie folgt installieren:

```
sudo yum install httpd mod_ssl
sudo /usr/sbin/apachectl start
```

2. Definieren Sie den/die MIME-Typ(en) des RDF-Dateinhalts, den Sie laden möchten. SPARQL bestimmt das Eingabeformat des Inhalts anhand des vom Webserver gesendeten `Content-type`-Headers. Sie müssen daher die relevanten MIME-Typen für den Webserver definieren.

   Angenommen, Sie verwenden die folgenden Dateierweiterungen zum Identifizieren von Dateiformaten:

   - `.nt` für NTriples.
   - `.nq` für NQuads.
   - `.trig` für Trig.
   - `.rdf`für RDF/XML.
   - `.ttl` für Turtle.
   - `.n3` für N3.

- `.jsonld`für JSON-LD.

Wenn Sie Apache 2 als Webserver verwenden, würden Sie die Datei `/etc/mime.types` bearbeiten und die folgenden Typen hinzufügen:

```
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. Vergewissern Sie sich, dass das MIME-type Mapping funktioniert. Sobald ein Webserver in Betrieb ist und RDF-Dateien in einem Format/in Formaten Ihrer Wahl hostet, können Sie die Konfiguration testen, indem Sie von Ihrem lokalen Host eine Anfrage an den Webserver senden.

   Sie können beispielsweise eine Anfrage wie z. B. die Folgende senden:

   ```
   curl -v http://localhost:80/test.rdf
   ```

   Anschließend sollten Sie in einer detaillierten Ausgabe von `curl` eine Zeile ähnlich der Folgenden sehen:

   ```
   Content-Type: application/rdf+xml
   ```

   Daran ist zu erkennen, dass die Zuweisung des Inhaltstyps erfolgreich definiert wurde.

4. Sie sind nun zum Laden von Daten über den Befehl SPARQL UPDATE bereit:

   ```
   curl https://your-neptune-endpoint:port/sparql \
       -d "update=LOAD <http://web_server_private_ip:80/test.rdf>"
   ```

> **Note**
>
> Die Verwendung von `SPARQL UPDATE LOAD` kann eine Zeitüberschreitung auf dem Webserver auslösen, wenn die zu ladende Quelldatei groß ist. Neptune verarbeitet die Daten der Datei, während sie hinein gestreamt werden. Für eine große Datei kann dies den auf

dem Server konfigurierten Zeitüberschreitungswert überschreiten. Dies kann wiederum dazu führen, dass der Server die Verbindung schließt, was zu der folgenden Fehlermeldung führen kann, wenn Neptune auf eine unerwartete EOF-Meldung im Stream trifft:

```
{
  "detailedMessage":"Invalid syntax in the specified file",
  "code":"InvalidParameterException"
}
```

Wenn Sie diese Meldung erhalten und nicht annehmen, dass Ihre Quelldatei ungültige Syntax enthält, versuchen Sie, die Zeitüberschreitungseinstellungen auf dem Webserver zu erhöhen. Sie können das Problem auch diagnostizieren, indem Sie Debug-Protokolle auf dem Server aktivieren und nach Zeitüberschreitungen suchen.

Verwenden von SPARQL UPDATE UNLOAD zum Löschen von Daten aus Neptune

Neptune stellt auch eine benutzerdefinierte SPARQL-Operation (`UNLOAD`) zum Entfernen von Daten bereit, die in einer Remote-Quelle angegeben sind. `UNLOAD` kann als Gegenstück zur Operation `LOAD` betrachtet werden. Die Syntax ist wie folgt:

```
UNLOAD SILENT (URL of the remote data to be unloaded) FROM GRAPH (named graph from
 which to remove the data)
```

- **`SILENT`** – (Optional) Bewirkt, dass die Operation auch dann erfolgreich ist, wenn bei der Verarbeitung der Daten ein Fehler aufgetreten ist.

  Dies kann nützlich sein, wenn eine einzelne Transaktion mehrere Anweisungen enthält, z. B. `"LOAD ...; LOAD ...; UNLOAD ...; LOAD ...;"`, und die Transaktion abgeschlossen werden soll, auch wenn einige der entfernten Daten nicht verarbeitet werden konnten.
- *`URL of the remote data to be unloaded`*— (Erforderlich) Gibt eine Remote-Datendatei an, die Daten enthält, die aus einem Diagramm entladen werden sollen.

  Die Remote-Datei muss eine der folgenden Erweiterungen haben (dies sind dieselben Formate, die UPDATE-LOAD unterstützt werden):
  - `.nt` für NTriples.
  - `.nq` für NQuads.
  - `.trig` für Trig.

- `.rdf`für RDF/XML.
- `.ttl` für Turtle.
- `.n3` für N3.
- `.jsonld`für JSON-LD.

Alle Daten, die diese Datei enthält, werden durch die Operation `UNLOAD` aus Ihrem DB-Cluster entfernt.

Jede Amazon-S3-Authentifizierung muss in der URL enthalten sein, damit die Daten entladen werden können. Sie können eine Amazon-S3-Datei vorsignieren und dann über die resultierende URL sicher auf sie zugreifen. Beispiel:

```
aws s3 presign --expires-in (number of seconds) s3://(bucket name)/(path to file of
 data to unload)
```

Dann:

```
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)'
```

Weitere Informationen finden Sie auf der Seite zum Thema Authentifizieren von Anforderungen: Verwenden von Abfrageparametern.

- **`FROM GRAPH`** *`(named graph from which to remove the data)`*— (Optional) Gibt den benannten Graphen an, aus dem die entfernten Daten entladen werden sollen.

  Neptune ordnet jedem Triple ein benanntes Diagramm zu. Sie können das benannte Standarddiagramm mithilfe des Fallback-URI (`http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph`) für das benannte Diagramm wie folgt angeben:

  ```
  FROM GRAPH <http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph>
  ```

Auf die gleiche Weise, wie `LOAD INSERT DATA { (inline data) }` entspricht, entspricht `UNLOAD DELETE DATA { (inline data) }`. Wie `DELETE DATA`, funktioniert `UNLOAD` nicht für Daten, die leere Knoten enthalten.

Wenn ein lokaler Webserver beispielsweise eine Datei mit dem Namen `data.nt` bereitstellt, die die folgenden 2 Tripel enthält:

```
<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> .
```

Der folgende Befehl `UNLOAD` würde diese beiden Tripel aus dem benannten Diagramm (`<http://example.org/graph1>`) löschen.

```
UNLOAD <http://localhost:80/data.nt> FROM GRAPH <http://example.org/graph1>
```

Dies hätte den gleichen Effekt wie die Verwendung des folgenden Befehls `DELETE DATA`:

```
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> .
  }
}
```

Ausnahmen, die durch den Befehl **`UNLOAD`** ausgelöst wurden

- **`InvalidParameterException`** – Die Daten enthielten leere Knoten. HTTP-Status: 400 – Ungültige Anforderung.

  Meldung: `Blank nodes are not allowed for UNLOAD`

- **`InvalidParameterException`** – Die Daten enthielten eine fehlerhafte Syntax. HTTP-Status: 400 – Ungültige Anforderung.

  Meldung: `Invalid syntax in the specified file.`

- **`UnloadUrlAccessDeniedException`** – Zugriff wurde abgelehnt. HTTP-Status: 400 – Ungültige Anforderung.

Meldung: `Update failure: Endpoint (Neptune endpoint) reported access denied error. Please verify access.`

- **`BadRequestException`** – Die Remote-Daten können nicht abgerufen werden. HTTP-Status: 400 – Ungültige Anforderung.

  Meldung: (von der HTTP-Antwort abhängig).

SPARQL-Abfragehinweise

Sie können mit Abfragehinweisen Optimierungs- und Auswertungsstrategien für eine bestimmte SPARQL-Abfrage in Amazon Neptune angeben.

Abfragehinweise werden mittels zusätzlicher Triple-Muster ausgedrückt, die in die SPARQL-Abfrage eingebettet sind und folgende Teile haben:

```
scope hint value
```

- scope – Bestimmt den Teil der Abfrage, auf den der Abfragehinweis angewendet wird, wie eine bestimmte Gruppe in der Abfrage oder die gesamte Abfrage.
- hint – Identifiziert die Art des anzuwendenden Hinweises.
- value – Legt das Verhalten des jeweiligen Systemaspekts fest.

Die Abfragehinweise und Bereiche werden als vordefinierte Begriffe im Amazon–Neptune-Namespace `http://aws.amazon.com/neptune/vocab/v01/QueryHints#` angezeigt. Die Beispiele in diesem Abschnitt enthalten den Namespace als ein `hint`-Präfix, das in der Abfrage definiert und enthalten ist:

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
```

Das folgende Beispiel zeigt beispielsweise, wie Sie einen `joinOrder`-Hinweis in einer `SELECT`-Abfrage einschließen:

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
SELECT ... {
```

```
  hint:Query hint:joinOrder "Ordered" .
  ...
}
```

Die vorherige Abfrage weist die Neptune-Engine an, Joins in der Abfrage in der angegebenen Reihenfolge auszuwerten, und deaktiviert jede automatische Neuanordnung.

Sie sollten bei der Verwendung von Abfragehinweisen folgende Punkte beachten:

- Sie können in einer einzigen Abfrage verschiedene Abfragehinweise kombinieren. Beispielsweise können Sie den Abfragehinweis `bottomUp` verwenden, um eine Unterabfrage für die Auswertung von unten nach oben zu markieren, und den Abfragehinweis `joinOrder`, um die Join-Anordnung innerhalb der Unterabfrage festzulegen.
- Sie können den gleichen Abfragehinweis mehrmals in verschiedenen, sich nicht überschneidenden Bereichen verwenden.
- Abfragehinweise sind Hinweise. Obwohl die Abfrage-Engine in der Regel die Berücksichtigung der angegebenen Abfragehinweise anstrebt, kann sie diese auch ignorieren.
- Abfragehinweise bewahren die Semantik. Das Hinzufügen eines Abfragehinweises ändert die Ausgabe der Abfrage nicht (ausgenommen die potenzielle Ergebnisreihenfolge, wenn keine Anordnungsgarantien angegeben werden; d. h., wenn die Ergebnisreihenfolge nicht explizit mit ORDER BY durchgesetzt wird).

Die folgenden Abschnitte enthalten weitere Informationen zu den verfügbaren Abfragehinweisen und ihre Nutzung in Neptune.

Themen

- Bereich der SPARQL-Abfragehinweise in Neptune
- Der JoinOrder SPARQL-Abfragehinweis
- Der SPARQL-Abfragehinweis von EvaluationStrategy
- Der QueryTimeout SPARQL-Abfragehinweis
- Der RangeSafe SPARQL-Abfragehinweis
- Der queryId SPARQL-Abfragehinweis
- Der UserDFE SPARQL-Abfragehinweis
- SPARQL-Abfragehinweise, die mit DESCRIBE verwendet werden

Bereich der SPARQL-Abfragehinweise in Neptune

Die folgende Tabelle zeigt die verfügbaren Bereiche, zugehörigen Hinweise und Beschreibungen für SPARQL-Abfragehinweise in Amazon Neptune. Das Präfix `hint` in diesen Einträgen stellt den Neptune-Namespace für Hinweise dar:

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
```

Scope	Unterstützter Hinweis	Description
`hint:Query`	joinOrder	Der Abfragehinweis gilt für die gesamte Abfrage.
`hint:Query`	queryTimeout	Der Timeout-Wert gilt für die gesamte Abfrage.
`hint:Query`	rangeSafe	Die Typheraufstufung ist für die gesamte Abfrage deaktivie rt.
`hint:Query`	queryId	Der Abfrage-ID-Wert gilt für die gesamte Abfrage.
`hint:Query`	useDFE	Die DFE ist für die gesamte Abfrage aktiviert (oder deaktiviert).
`hint:Group`	joinOrder	Der Abfragehinweis gilt für die Elemente auf der obersten Ebene in der angegeben en Gruppe, jedoch nicht für verschachtelte Elemente (z. B. Unterabfragen) oder übergeordnete Elemente.
`hint:SubQuery`	evaluationStrategy	Der Hinweis angegeben ist und gilt für eine verschachtelte SELECT-Unterabfrage. Die Unterabfrage wird unabhängi

Scope	Unterstützter Hinweis	Description
		g ohne Berücksichtigung von Lösungen ausgewertet, die vor der Unterabfrage berechnet wurden.

Der **`JoinOrder`** SPARQL-Abfragehinweis

Wenn Sie eine SPARQL-Abfrage senden, untersucht die Abfrage-Engine von Amazon Neptune die Struktur der Abfrage. Sie ordnet Teile der Abfrage neu an und versucht, den für die Auswertung erforderlichen Aufwand und die Reaktionszeit der Abfrage zu reduzieren.

Beispielsweise wird eine Folge verbundener Triple-Muster in der Regel nicht in der angegebenen Reihenfolge ausgewertet. Sie wird mittels Heuristik und Statistik neu angeordnet, beispielweise der Selektivität der einzelnen Muster und der Art ihrer Verbindung mittels gemeinsamer Variablen. Wenn Ihre Abfrage außerdem komplexere Muster wie Unterabfragen, FILTER oder komplexe OPTIONAL- oder MINUS-Blöcke enthält, ordnet die Abfrage-Engine von Neptune diese nach Möglichkeit neu an und strebt dabei eine effiziente Auswertungsreihenfolge an.

Für komplexere Abfragen ist die Reihenfolge, in der Neptune die Abfrage bewertet, möglicherweise nicht immer optimal. Beispielsweise übersieht Neptune möglicherweise spezifische Instance-Dateneigenschaften (z. B. Power-Knoten im Diagramm), die während der Auswertung der Abfrage auftauchen.

Wenn Sie die genauen Merkmale der Daten kennen und die Reihenfolge der Abfrageausführung manuell festlegen möchten, können Sie den Neptune-Abfragehinweis `joinOrder` verwenden, um anzugeben, dass die Abfrage in der angegebenen Reihenfolge ausgewertet werden soll.

`JoinOrder` SPARQL-Hinweissyntax

Der Abfragehinweis `joinOrder` ist als Triple-Muster angegeben, das in einer SPARQL-Abfrage enthalten ist.

Die folgende Syntax verwendet das Präfix `hint`, das in der Abfrage definiert und eingefügt ist, zur Angabe des Neptune-Abfragehinweis-Namespace:

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
scope hint:joinOrder "Ordered" .
```

Verfügbare Bereiche

- `hint:Query`
- `hint:Group`

Weitere Informationen zu Bereichen von Abfragehinweisen finden Sie unter Bereich der SPARQL-Abfragehinweise in Neptune.

Beispiel für einen **`JoinOrder`** SPARQL-Hinweis

Dieser Abschnitt zeigt eine mit und ohne den Abfragehinweis `joinOrder` geschriebene Abfrage und die entsprechenden Optimierungen.

In diesem Beispiel wird davon ausgegangen, dass der Datensatz die folgenden Elemente enthält:

- Eine einzelne Person mit dem Namen `John`, die 1.000 Personen mit `:likes` markiert hat, darunter `Jane`.
- Eine einzelne Person mit dem Namen `Jane`, die 10 Personen mit `:likes` markiert hat, darunter `John`.

Kein Abfragehinweis

Die folgende SPARQL-Abfrage extrahiert alle Paare von Personen mit dem Namen `John` und `Jane` aus einem Satz von Daten aus sozialen Netzwerken, die sich gegenseitig mit „Like“ (Gefällt) markiert haben.

```
PREFIX : <https://example.com/>
SELECT ?john ?jane {
  ?person1 :name "Jane" .
  ?person1 :likes ?person2 .
  ?person2 :name "John" .
  ?person2 :likes ?person1 .
}
```

Die Abfrage-Engine von Neptune wertet die Anweisungen möglicherweise in einer anderen Reihenfolge als geschrieben aus. Sie könnte sie beispielsweise in der folgenden Reihenfolge auswerten:

1. Suche nach allen Personen mit dem Namen `John`.

2. Suche nach allen mit `John` über das Edge `:likes` verbundenen Personen.
3. Filtern dieses Satzes nach Personen mit dem Namen `Jane`.
4. Filtern dieses Satzes nach mit `John` über das Edge `:likes` verbundenen Personen.

Entsprechend dem Datensatz ergibt die Auswertung in der Reihenfolge 1.000 Entitäten, die im zweiten Schritt extrahiert werden. Der dritte Schritt engt dies auf den einzelnen Knoten `Jane` ein. Im letzten Schritt wird ermittelt, dass `Jane` den Knoten `John` auch mit `:likes` markiert hat.

Abfragehinweis

Es wäre günstiger, mit dem Knoten `Jane` zu beginnen, da dieser nur 10 ausgehende `:likes` Edges besitzt. Dies reduziert den Aufwand während der Auswertung der Abfrage, da das Extrahieren der 1.000 Entitäten im zweiten Schritt vermieden wird.

Das folgende Beispiel verwendet den Abfragehinweis joinOrder, um sicherzustellen, dass der Knoten `Jane` und dessen ausgehenden Kanten zuerst verarbeitet werden, indem die automatische Neuanordnung der Verbindung für die Abfrage deaktiviert ist:

```
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 .
}
```

Ein reales Szenario wäre etwa eine Social-Network-Anwendung, bei der Personen im Netzwerk als Influencer mit zahlreichen Verbindungen oder als normale Nutzer mit wenigen Verbindungen klassifiziert würden. In einem solchen Szenario könnten Sie dafür sorgen, dass in einem Beispiel wie dem obigen der normale Benutzer (`Jane`) vor dem Influencer (`John`) verarbeitet würde.

Abfragehinweis und Neuanordnung

Sie können mit diesem Beispiel einen Schritt weiter gehen. Wenn Sie wissen, dass das `:name`-Attribut für einen Knoten eindeutig ist, könnten Sie die Abfrage durch Neuanordnung und Verwendung des Abfragehinweises `joinOrder` beschleunigen Dieser Schritt stellt sicher, dass die eindeutigen Knoten zuerst extrahiert werden.

```
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 .
}
```

In diesem Fall können Sie die Abfrage auf die folgenden einzelnen Aktionen in jedem Schritt reduzieren:

1. Suchen des Einzelpersonknotens mit `:name Jane`.
2. Suchen des Einzelpersonknotens mit `:name John`.
3. Prüfen, ob der erste Knoten über eine `:likes`-Edge mit dem zweiten verbunden ist.
4. Prüfen, ob der zweite Knoten über eine `:likes`-Edge mit dem ersten verbunden ist.

> **Important**
>
> Wenn Sie die falsche Reihenfolge wählen, kann der Abfragehinweis `joinOrder` zu erheblichen Leistungseinbußen führen. Das vorangehende Beispiel wäre beispielsweise ineffizient, wenn die `:name`-Attribute nicht eindeutig wären. Hätten alle 100-Knoten den Namen `Jane` und alle 1.000-Knoten den Namen `John`, würde die Abfrage 1.000 * 100 (100.000) Paare für `:likes`-Edges prüfen.

Der **`SPARQL-Abfragehinweis von EvaluationStrategy`**

Der Abfragehinweis `evaluationStrategy` teilt der Amazon–Neptune-Abfrageengine mit, dass das markierte Abfragefragment von unten nach oben als unabhängige Einheit evaluiert werden soll. Dies bedeutet, dass keine Lösungen aus vorherigen Evaluierungsschritten für die Berechnung des Abfragefragments verwendet werden. Das Abfragefragment wird als selbstständige Einheit evaluiert, und seine produzierten Lösungen werden nach der Berechnung mit dem Rest der Abfrage verbunden.

Die Verwendung des `evaluationStrategy`-Abfragehinweises impliziert die Blockierung eines Abfrageplans (ohne Pipeline); dies bedeutet, dass die Lösungen auf dem mit dem Abfragehinweis versehenen Fragment im Hauptspeicher materialisiert und zwischengespeichert werden. Die Verwendung dieses Abfragehinweises kann die Menge des Hauptspeichers deutlich erhöhen, die für die Evaluierung der Abfrage erforderlich ist, besonders wenn das markierte Abfragefragment eine große Zahl von Ergebnissen berechnet.

Syntax des SPARQL-Hinweises **`von EvaluationStrategy`**

Der Abfragehinweis `evaluationStrategy` ist als Triple-Muster angegeben, das in einer SPARQL-Abfrage enthalten ist.

Die folgende Syntax verwendet das Präfix `hint`, das in der Abfrage definiert und eingefügt ist, zur Angabe des Neptune-Abfragehinweis-Namespace:

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
hint:SubQuery hint:evaluationStrategy "BottomUp" .
```

Verfügbare Bereiche

- `hint:SubQuery`

> **Note**
>
> Dieser Abfragehinweis wird nur in verschachtelten Unterabfragen unterstützt.

Weitere Informationen zu Bereichen von Abfragehinweisen finden Sie unter Bereich der SPARQL-Abfragehinweise in Neptune.

Beispiel für einen **`SPARQL-Hinweis von EvaluationStrategy`**

Dieser Abschnitt zeigt eine mit und ohne den Abfragehinweis `evaluationStrategy` geschriebene Abfrage und die entsprechenden Optimierungen.

In diesem Beispiel wird davon ausgegangen, dass der Datensatz über die folgenden Eigenschaften verfügt:

- Er enthält 1.000 Edges mit der Beschriftung `:connectedTo`.

- Jeder `component`-Knoten ist mit durchschnittlich 100 anderen `component`-Knoten verbunden.
- Die typische Anzahl von vier-hop-zyklischen Verbindungen zwischen Knoten liegt bei etwa 100.

Als ein typisches Beispiel kann der `evaluationStrategy`-Hinweis hilfreich sein, um Abfragemuster mit Zyklen zu optimieren.

Kein Abfragehinweis

Die folgende SPARQL-Abfrage extrahiert alle `component`-Knoten, die zyklisch miteinander über vier Hops verbunden sind:

```
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 .
}
```

Das Konzept der Neptune-Abfrageengine besteht darin, diese Abfrage mit den folgenden Schritten auszuwerten:

- Extrahieren aller 1.000 `connectedTo`-Edges in dem Graph.
- Erweitern Sie dies um das Hundertfache (die Anzahl der ausgehenden `connectedTo`-Edges von Component2).

  Zwischenergebnisse: 100.000 Knoten.
- Erweitern Sie dies um das Hundertfache (die Anzahl der ausgehenden `connectedTo`-Edges von Component3).

  Zwischenergebnisse: 10.000.000 Knoten.
- Scannen Sie die 10.000.000 Knoten für den Zyklusabschluss.

Dies führt zu einem Streaming-Abfrageplan, der eine konstante Menge an Arbeitsspeicher verwendet.

Abfragehinweis und Unterabfragen

Möglicherweise möchten Sie Kompromisse beim Hauptspeicherplatz machen, um die Berechnungen zu beschleunigen. Indem Sie die Abfrage mithilfe eines `evaluationStrategy`-Abfragehinweises umschreiben, können Sie erzwingen, dass die Engine eine Verbindung zwischen kleineren, materialisierten Teilmengen berechnet.

```
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 .
    }
  }
}
```

Anstatt die Triple-Muster nacheinander bei iterativer Verwendung der Ergebnisse vom vorherigen Triple-Muster als Eingabe für die folgenden Muster zu evaluieren, sorgt der `evaluationStrategy`-Hinweis dafür, dass die beiden Unterabfragen unabhängig evaluiert werden. Beide Unterabfragen produzieren 100.000 Knoten für Zwischenergebnisse, die dann miteinander verbunden werden, um die endgültige Ausgabe zu erstellen.

Wenn Sie Neptune auf den größeren Instance-Typen ausführen, erhöht die temporäre Speicherung dieser beiden Teilmengen von 100 000 die Speichernutzung, ermöglicht damit jedoch deutlich schnellere Auswertungen.

Der **`QueryTimeout` SPARQL-Abfragehinweis**

Der `queryTimeout`-Abfragehinweis gibt eine Zeitüberschreitung an, die kürzer ist als der `neptune_query_timeout`-Wert, der in der DB-Parameter-Gruppe festlegt ist.

Wenn die Abfrage aufgrund dieses Hinweises beendet wird, wird ein`TimeLimitExceededException` mit einer `Operation terminated (deadline`

`exceeded)`- Nachricht ausgelöst. Ob Sie eine Abfrage mit Timeout erneut versuchen sollten, hängt von der Art des Fehlers und Ihrer Arbeitslast ab. Anleitungen finden Sie unter the section called “Ausnahmen und Wiederholungen”.

`QueryTimeout`: SPARQL-Hinweissyntax

```
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 .
 ...
}
```

Der Timeout-Wert wird in Millisekunden angegeben.

Der Timeout-Wert muss kleiner sein als der in der DB-Parametergruppe festgelegte `neptune_query_timeout`-Wert. Andernfalls wird eine `MalformedQueryException`-Ausnahme mit einer `Malformed query: Query hint 'queryTimeout' must be less than neptune_query_timeout DB Parameter Group`-Nachricht ausgelöst.

Der `queryTimeout`-Abfragehinweis sollte in der `WHERE`-Klausel der Hauptabfrage oder in der `WHERE`-Klausel einer der Unterabfragen, wie im nachfolgenden Beispiel dargestellt, angegeben werden.

Sie darf in allen Abschnitten queries/subqueries und SPARQL-Updates (wie INSERT und DELETE) nur einmal gesetzt werden. Andernfalls wird eine `MalformedQueryException`-Ausnahme mit einer `Malformed query: Query hint 'queryTimeout' must be set only once`-Nachricht ausgelöst.

Verfügbare Bereiche

Der `queryTimeout`-Hinweis kann sowohl auf SPARQL-Abfragen als auch auf Updates angewendet werden.

- In einer SPARQL-Abfrage kann sie in der WHERE-Klausel der Hauptabfrage oder einer Unterabfrage erscheinen.
- In einem SPARQL-Update kann er in der INSERT-, DELETE- oder WHERE-Klausel festgelegt werden. Wenn es mehrere Update-Klauseln gibt, kann er nur in einer davon festgelegt werden.

Weitere Informationen zu Bereichen von Abfragehinweisen finden Sie unter Bereich der SPARQL-Abfragehinweise in Neptune.

`QueryTimeout`, Beispiel für einen SPARQL-Hinweis

Hier finden Sie ein Beispiel für die Verwendung von `hint:queryTimeout` in der `WHERE`-Hauptklausel einer `UPDATE`-Abfrage:

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
INSERT {
    ?s ?p ?o
} WHERE {
    hint:Query hint:queryTimeout 100 .
    ?s ?p ?o .
}
```

Hier ist `hint:queryTimeout` eine Unterabfrage in der `WHERE`-Klausel:

```
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 .
      }
   }
}
```

Der RangeSafe SPARQL-Abfragehinweis

Mit diesem Abfragehinweis können Sie die Typheraufstufung für eine SPARQL-Abfrage deaktivieren.

Wenn Sie eine SPARQL-Abfrage absenden, die einen `FILTER` für einen numerischen Wert oder Bereich enthält, muss die Neptune-Abfrage-Engine normalerweise den Typ heraufstufen, wenn sie die Abfrage ausführt. Das bedeutet, dass sie die Werte aller Typen untersuchen muss, die den Wert enthalten könnten, nach dem Sie filtern.

Wenn Sie beispielsweise nach Werten filtern, die gleich 55 sind, muss die Engine nach Ganzzahlen gleich 55, langen Ganzzahlen gleich 55L, Gleitkommazahlen gleich 55,0 usw. suchen. Jede

Typheraufstufung erfordert eine zusätzliche Suche im Speicher. Dies kann dazu führen, dass eine scheinbar einfache Abfrage unerwartet lange dauert, bis sie abgeschlossen ist.

Häufig ist eine Heraufstufung des Typs nicht notwendig, da Sie bereits wissen, dass Sie nur Werte eines bestimmten Typs suchen müssen. In diesem Fall können Sie Ihre Abfragen erheblich beschleunigen, indem Sie die Typheraufstufung mit dem Abfragehinweis `rangeSafe` deaktivieren.

Syntax des **`RangeSafe-SPARQL-Hinweises`**

Der Abfragehinweis `rangeSafe` nimmt den Wert `true` an, um die Typheraufstufung zu deaktivieren. Er akzeptiert auch den Wert `false` (Standard).

Beispiel. Das folgende Beispiel zeigt, wie die Typheraufstufung deaktiviert wird, wenn nach einem Ganzzahlwert von o größer als 1 gefiltert wird:

```
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>)
```

Der **`queryId SPARQL-Abfragehinweis`**

Verwenden Sie diesen Abfragehinweis, um einer SPARQL-Abfrage einen eigenen queryId-Wert zuzuweisen.

Beispiel:

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
SELECT * WHERE {
  hint:Query hint:queryId "4d5c4fae-aa30-41cf-9e1f-91e6b7dd6f47"
  {?s ?p ?o}}
```

Der Wert, den Sie zuweisen, muss für alle Abfragen in der Neptune-DB eindeutig sein.

Der UserDFE **`SPARQL-Abfragehinweis`**

Mit diesem Abfragehinweis können Sie die Verwendung der DFE zur Ausführung der Abfrage aktivieren. Standardmäßig verwendet Neptune die DFE nicht, wenn dieser Abfragehinweis nicht

auf `true` festgelegt ist, da der Instance-Parameter neptune_dfe_query_engine standardmäßig auf `viaQueryHint` festgelegt ist. Wenn Sie diesen Instance-Parameter auf `enabled` festlegen, wird die DFE-Engine für alle Abfragen verwendet, außer Abfragen, bei denen der Abfragehinweis `useDFE` auf `false` festgelegt ist.

Beispiel für die Aktivierung der Verwendung der DFE für eine Abfrage:

```
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 .
}
```

SPARQL-Abfragehinweise, die mit DESCRIBE verwendet werden

Eine SPARQL-`DESCRIBE`Abfrage stellt einen flexiblen Mechanismus zum Anfordern von Ressourcenbeschreibungen bereit. Die SPARQL-Spezifikationen definieren jedoch nicht die genaue Semantik von `DESCRIBE`.

Ab Engine-Version 1.2.0.2 unterstützt Neptune mehrere verschiedene `DESCRIBE`-Modi und -Algorithmen, die für unterschiedliche Situationen geeignet sind.

Dieser Beispieldatensatz kann helfen, die verschiedenen Modi zu veranschaulichen:

```
@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 .
```

Die folgenden Beispiele gehen davon aus, dass eine Beschreibung der Ressource `:JaneDoe` mit einer SPARQL-Abfrage wie dieser angefordert wird:

```
DESCRIBE <https://example.com/JaneDoe>
```

Der **`DescribeMode` SPARQL-Abfragehinweis**

Der SPARQL-Abfragehinweis `hint:describeMode` wird verwendet, um einen der folgenden, von Neptune unterstützten SPARQL-`DESCRIBE`-Modi auszuwählen:

Der DESCRIBE-Modus **`ForwardOneStep`**

Sie rufen den Modus `ForwardOneStep` wie folgt mit dem Abfragehinweis `describeMode` auf:

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
DESCRIBE <https://example.com/JaneDoe>
{
  hint:Query hint:describeMode "ForwardOneStep"
}
```

Der Modus `ForwardOneStep` gibt nur die Attribute und Forward-Links der Ressource zurück, die beschrieben werden soll. Im Beispiel gibt der Modus die Tripel zurück, deren Subjekt `:JaneDoe` ist, die Ressource, die beschrieben werden soll.

```
:JaneDoe :firstName "Jane" .
:JaneDoe :knows :JohnDoe .
:JaneDoe :knows _:b301990159 .
```

Beachten Sie, dass die DESCRIBE-Abfrage Tripel mit leeren Knoten zurückgeben kann, z. B. `_:b301990159`, die im Vergleich zum Eingabedatensatz jeweils unterschiedliche IDs haben.

Der DESCRIBE-Modus **`SymmetricOneStep`**

`SymmetricOneStep` ist der DESCRIBE-Standardmodus, wenn Sie keinen Abfragehinweis angeben. Sie können ihn auch explizit mit dem Abfragehinweis `describeMode` wie folgt aufrufen:

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
DESCRIBE <https://example.com/JaneDoe>
{
  hint:Query hint:describeMode "SymmetricOneStep"
}
```

In der Semantik `SymmetricOneStep` gibt `DESCRIBE` die Attribute, Forward- und Reverse-Links der Ressource zurück, die beschrieben werden soll:

```
:JaneDoe :firstName "Jane" .
:JaneDoe :knows :JohnDoe .
:JaneDoe :knows _:b318767375 .

_:b318767631 rdf:subject :JaneDoe .

:RichardRoe :knows :JaneDoe .

:ref_s2 rdf:subject :JaneDoe .
```

Der DESCRIBE-Modus (Concise Bounded Description)

Der Modus Concise Bounded Description (`CBD`) wird mit dem Abfragehinweis `describeMode` wie folgt aufgerufen:

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
DESCRIBE <https://example.com/JaneDoe>
{
  hint:Query hint:describeMode "CBD"
}
```

In der Semantik `CBD` gibt `DESCRIBE` die Concise Bounded Description (wie vom W3C definiert) der zu beschreibenden Ressource zurück:

```
: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 .
```

Die Concise Bounded Description einer RDF-Ressource (d. h. eines Knotens in einem RDF-Diagramm) ist das kleinste Unterdiagramm um diesen Knoten herum, das eigenständig sein kann. Wenn Sie sich dieses Diagramm als Baum vorstellen, bei dem der angegebene Knoten die Wurzel ist, dann gibt es keine leeren Knoten (bnodes) als Blätter dieses Baums. Da bnodes nicht extern adressiert oder in nachfolgenden Abfragen verwendet werden können, reicht es nicht aus, das Diagramm zu durchsuchen, nur um den oder die nächsten einzelnen Hop(s) vom aktuellen Knoten aus zu finden. Sie müssen auch weit genug gehen, um etwas zu finden, das in nachfolgenden Abfragen verwendet werden kann (d. h. etwas anderes als ein bnode).

Berechnen der CBD

Für einen bestimmten Knoten (den Startknoten oder die Wurzel) im RDF-Quelldiagramm wird der CBD-Wert dieses Knotens wie folgt berechnet:

1. Fügen Sie in das Unterdiagramm alle Anweisungen im Quelldiagramm ein, deren Subjekt der Anweisung der Startknoten ist.
2. Fügen Sie rekursiv für alle vorhandenen Anweisungen im Unterdiagramm, die ein leeres Knotenobjekt haben, in das Quelldiagramm alle Anweisungen ein, in denen das Subjekt der Anweisung dieser leere Knoten ist und die noch nicht im Unterdiagramm enthalten sind.
3. Fügen Sie rekursiv für alle bisher eingefügten Anweisungen im Unterdiagramm, für alle Konkretisierungen dieser Anweisungen im Quelldiagramm die CBD ab dem `rdf:Statement`-Knoten jeder Konkretisierung ein.

Dies führt zu einem Unterdiagramm, in dem die Objektknoten entweder IRI-Referenzen oder Literale sind oder leere Knoten, die nicht als Subjekt einer Anweisung im Diagramm dienen. Beachten Sie,

dass die CBD nicht mit einzelnen SPARQL SELECT- oder CONSTRUCT-Abfrage berechnet werden kann.

Der DESCRIBE-Modus (Symmetric Concise Bounded Description)

Der Modus Symmetric Concise Bounded Description (`SCBD`) wird mit dem Abfragehinweis `describeMode` wie folgt aufgerufen:

```
PREFIX hint: <http://aws.amazon.com/neptune/vocab/v01/QueryHints#>
DESCRIBE <https://example.com/JaneDoe>
{
  hint:Query hint:describeMode "SCBD"
}
```

In der `SCBD`-Semantik gibt `DESCRIBE` die Symmetric Concise Bounded Description der Ressource zurück (wie vom W3C in Describing Linked Datasets with the VoID Vocabulary definiert:

```
: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 .
```

Der Vorteil von CBD und SCBD gegenüber den Modi `ForwardOneStep` und `SymmetricOneStep` besteht darin, dass leere Knoten stets um ihre Darstellung erweitert werden. Dies kann ein wichtiger Vorteil sein, da Sie einen leeren Knoten nicht über SPARQL abfragen können. Darüber hinaus berücksichtigen die Modi CBD und SCBD auch Konkretisierungen.

Beachten Sie, dass der Abfragehinweis `describeMode` auch Teil einer `WHERE`-Klausel sein kann:

```
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>
}
```

Der SPARQL-Abfragehinweis beschreiben IterationLimit

Der SPARQL-Abfragehinweis `hint:describeIterationLimit` stellt eine optionale Einschränkung der maximalen Anzahl iterativer Erweiterungen bereit, die für iterative DESCRIBE-Algorithmen wie CBD und SCBD ausgeführt werden sollen.

Die DESCRIBE-Einschränkungen sind durch UND verknüpft. Wenn also sowohl die Iterationseinschränkung als auch die Befehlseinschränkung angegeben sind, müssen beide Einschränkungen eingehalten werden, bevor die DESCRIBE-Abfrage abgeschnitten wird.

Der Standardwert für diesen Wert ist 5. Sie können ihn auf NULL (0) festlegen, um anzugeben, dass die Anzahl der iterativen Erweiterungen KEINEN Einschränkungen unterliegt.

Der StatementLimitDescribe-SPARQL-Abfragehinweis

Der SPARQL-Abfragehinweis `hint:describeStatementLimit` stellt eine optionale Einschränkung der maximalen Anzahl von Anweisungen dar, die in einer DESCRIBE-Abfrageantwort enthalten sein dürfen. Er wird nur auf iterative DESCRIBE-Algorithmen wie CBD und SCBD angewendet.

Die DESCRIBE-Einschränkungen sind durch UND verknüpft. Wenn also sowohl die Iterationseinschränkung als auch die Befehlseinschränkung angegeben sind, müssen beide Einschränkungen eingehalten werden, bevor die DESCRIBE-Abfrage abgeschnitten wird.

Der Standardwert für diesen Wert ist 5 000. Sie können ihn auf NULL (0) festlegen, um anzugeben, dass die Anzahl der zurückgegebenen Anweisungen KEINEN Einschränkungen unterliegt.

Verhalten von SPARQL DESCRIBE in Bezug auf das Standarddiagramm

Über das SPARQL-Abfrageformular `DESCRIBE` können Sie Informationen zu Ressourcen abrufen, ohne die Struktur der Daten zu kennen und ohne eine Abfrage schreiben zu müssen. Wie diese Informationen zusammengestellt werden, ist der SPARQL-Implementierung überlassen. Neptune stellt mehrere Abfragehinweise bereit, die verschiedene Modi und Algorithmen zur Verwendung durch `DESCRIBE` aufrufen.

In der Neptune-Implementierung verwendet `DESCRIBE` unabhängig vom Modus ausschließlich Daten im SPARQL-Standarddiagramm. Dies entspricht der Art, wie SPARQL Datensätze behandelt (siehe Specifying RDF Datasets in der SPARQL-Spezifikation).

In Neptune enthält das Standarddiagramm alle eindeutigen Tripel in der Vereinigung aller benannten Graphen in der Datenbank, es sei denn, bestimmte benannte Graphen werden mithilfe von Klauseln spezifiziert. `FROM` and/or `FROM NAMED` Alle RDF-Daten in Neptune werden in einem benannten Diagramm gespeichert. Wenn ein Tripel außerhalb eines Kontexts mit einem benannten Diagramm eingefügt wird, speichert Neptune es in einem benannten Graphen mit der Bezeichnung `http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph`.

Wenn ein oder mehrere benannte Diagramme mittels der `FROM`-Klausel angegeben werden, ist das Standarddiagramm die Vereinigung aller eindeutigen Tripel in diesen benannten Diagrammen. Wenn es keine `FROM`-Klausel gibt, jedoch eine oder mehrere `FROM NAMED`-Klauseln, ist das Standarddiagramm leer.

SPARQL DESCRIBE-Beispiele

Betrachten Sie die folgende Daten:

```
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" .
```

Für diese Abfrage:

```
PREFIX ex: <https://example.com/>
DESCRIBE ?s
FROM ex:g1
FROM NAMED ex:g2
WHERE {
```

```
  GRAPH ex:g2 { ?s ?p "b" . }
}
```

Neptune würde Folgendes zurück:

```
ex:s ex:p1 "a" .
ex:s ex:p2 "c" .
```

Hier wird zuerst das Diagrammmuster `GRAPH ex:g2 { ?s ?p "b" }` ausgewertet, was zu Bindungen für `?s` führt. Anschließend wird der Teil `DESCRIBE` anhand des Standarddiagramms ausgewertet, das jetzt `ex:g1` ist.

Für diese Abfrage gilt jedoch Folgendes:

```
PREFIX ex: <https://example.com/>
DESCRIBE ?s
FROM NAMED ex:g1
WHERE {
  GRAPH ex:g1 { ?s ?p "a" . }
}
```

Neptune würde nichts zurückgeben, da das Standarddiagramm leer ist, wenn eine `FROM NAMED`-Klausel, jedoch keine `FROM`-Klausel vorhanden ist.

In der folgenden Abfrage wird `DESCRIBE` verwendet, ohne dass eine `FROM`- oder `FROM NAMED`-Klausel vorhanden ist:

```
PREFIX ex: <https://example.com/>
DESCRIBE ?s
WHERE {
  GRAPH ex:g1 { ?s ?p "a" . }
}
```

In dieser Situation besteht das Standarddiagramm aus allen eindeutigen Tripeln in der Vereinigung aller benannten Diagramme in der Datenbank (formal die RDF-Zusammenführung). Neptune würde daher Folgendes zurückgeben:

```
ex:s ex:p1 "a" .
ex:s ex:p2 "c" .
ex:s ex:p3 "b" .
```

```
ex:s ex:p3 "d" .
```

SPARQL-Abfragestatus-API

Zum Abrufen des Status von SPARQL-Abfragen verwenden Sie HTTP `GET` oder `POST`, um eine Anforderung an den `https://`*`your-neptune-endpoint`*`:`*`port`*`/sparql/status`-Endpunkt zu senden.

SPARQL-Abfragestatus-Anforderungsparameter

queryId (optional)

Die ID einer laufenden SPARQL-Abfrage Zeigt nur den Status der angegebenen Abfrage an.

SPARQL-Abfragestatus-Antwortsyntax

```
{
    "acceptedQueryCount": integer,
    "runningQueryCount": integer,
    "queries": [
      {
        "queryId":"guid",
        "queryEvalStats":
          {
            "subqueries": integer,
            "elapsed": integer,
            "cancelled": boolean
          },
        "queryString": "string"
      }
    ]
}
```

SPARQL-Abfragestatus-Antwortwerte

akzeptiert QueryCount

Die Anzahl der Abfragen, die seit dem letzten Neustart der Neptune-Engine akzeptiert wurden.

läuft QueryCount

Die Anzahl der zurzeit laufenden SPARQL-Abfragen

queries

Eine Liste der aktuellen SPARQL-Abfragen

queryId

Eine GUID-ID für die Abfrage Neptune weist diesen ID-Wert automatisch jeder Abfrage zu. Sie können auch eine eigene ID zuweisen (siehe Einfügen einer benutzerdefinierten ID in eine Neptune-Gremlin- oder -SPARQL-Abfrage).

abfragen EvalStats

Statistiken für diese Abfrage

Unterabfragen

Die Anzahl der Unterabfragen in dieser Abfrage

elapsed

Die Dauer in Mikrosekunden, die die Abfrage bis jetzt ausgeführt wurde

cancelled

„True“ gibt an, dass die Abfrage abgebrochen wurde.

queryString

Die gesendete Abfrage

SPARQL-Abfragestatus-Beispiel

Das folgende Beispiel zeigt den Statusbefehl mit Verwendung von `curl` und HTTP `GET`.

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

Diese Ausgabe zeigt eine einzelne ausgeführte Abfrage.

```
{
    "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}"
        }
    ]
}
```

SPARQL-Abfrageabbruch

Zum Abrufen des Status von SPARQL-Abfragen verwenden Sie HTTP `GET` oder `POST`, um eine Anforderung an den `https://`*`your-neptune-endpoint`*`:`*`port`*`/sparql/status`-Endpunkt zu senden.

SPARQL-Abfrageabruch-Anforderungsparameter

cancelQuery

(Erforderlich) Weist den Statusbefehl an, eine Abfrage abzubrechen. Dieser Parameter akzeptiert keinen Wert.

queryId

(Erforderlich) Die ID der laufenden SPARQL-Abfrage, die abgebrochen werden soll.

Still

(Optional) Bei `silent=true` wird die laufende Abfrage abgebrochen, und der HTTP-Antwortcode lautet 200. Wenn `silent` nicht vorhanden ist oder bei `silent=false` wird die Abfrage mit einem HTTP 500-Statuscode abgebrochen.

SPARQL-Abfrageabbruch-Beispiele

Beispiel 1: Kündigung mit **`silent=false`**

Es folgt ein Beispiel für den Statusbefehl, der `curl` verwendet, um eine Abfrage abzubrechen, wobei der `silent`-Parameter auf `false` gesetzt ist:

```
curl https://your-neptune-endpoint:port/sparql/status \
  -d "cancelQuery" \
  -d "queryId=4d5c4fae-aa30-41cf-9e1f-91e6b7dd6f47" \
  -d "silent=false"
```

Wenn die Abfrage nicht bereits mit dem Streaming der Ergebnisse begonnen hat, würde die abgebrochene Abfrage dann einen HTTP-500-Code mit einer Antwort wie folgt zurückgeben:

```
{
  "code": "CancelledByUserException",
  "requestId": "4d5c4fae-aa30-41cf-9e1f-91e6b7dd6f47",
  "detailedMessage": "Operation terminated (cancelled by user)"
}
```

Wenn die Abfrage bereits einen HTTP-200-Code (OK) zurückgegeben hat und mit dem Streaming der Ergebnisse begonnen hat, bevor sie abgebrochen wurde, werden die Informationen zur Zeitbeschränkungsausnahme an den regulären Ausgabe-Stream gesendet.

Beispiel 2: Kündigung mit silent=true

Nachfolgend finden Sie ein Beispiel für denselben Statusbefehl wie oben mit Ausnahme des `silent`-Parameters, der jetzt auf `true` gesetzt ist:

```
curl https://your-neptune-endpoint:port/sparql/status \
  -d "cancelQuery" \
  -d "queryId=4d5c4fae-aa30-41cf-9e1f-91e6b7dd6f47" \
  -d "silent=true"
```

Dieser Befehl würde die gleiche Antwort wie bei `silent=false` zurückgeben, aber die abgebrochene Abfrage würde jetzt einen HTTP 200-Code mit einer Antwort wie folgt zurückgeben:

```
{
  "head" : {
    "vars" : [ "s", "p", "o" ]
  },
  "results" : {
    "bindings" : [ ]
  }
}
```

Verwenden des SPARQL 1.1-Graph-Store-Protokolls (GSP) über HTTP in Amazon Neptune

In der Empfehlung SPARQL 1.1 Graph Store HTTP Protocol hat das W3C ein HTTP-Protokoll für die Verwaltung von RDF-Diagrammen definiert. Es definiert Operationen für die Entfernung, Erstellung und Ersetzung von RDF-Diagramminhalten und die Hinzufügung von RDF-Anweisungen zu vorhandenem Inhalten.

Das Graph-Store-Protokoll (GSP) ermöglicht die Bearbeitung eines vollständigen Diagramms, ohne komplexe SPARQL-Abfragen schreiben zu müssen.

Neptune unterstützt dieses Protokoll vollständig.

Der Endpunkt für das Graph-Store-Protokoll (GSP) ist:

```
https://your-neptune-cluster:port/sparql/gsp/
```

Um über GSP auf das Standarddiagramm zuzugreifen, verwenden Sie:

```
https://your-neptune-cluster:port/sparql/gsp/?default
```

Um über GSP auf ein benanntes Diagramm zuzugreifen, verwenden Sie:

```
https://your-neptune-cluster:port/sparql/gsp/?graph=named-graph-URI
```

Besondere Details der Neptune-GSP-Implementierung

Neptune setzt die W3C-Empfehlung mit der GSP-Definition vollständig um. Es gibt jedoch einige Situationen, die die Spezifikation nicht abdeckt.

Ein solcher Fall tritt ein, wenn eine `PUT`- oder `POST`-Anforderung ein oder mehrere benannte Diagramme im Anforderungstext spezifiziert, die sich vom in der Anforderungs-URL angegebenen Diagramm unterscheiden. Dies ist nur möglich, wenn das RDF-Format des Anforderungstexts benannte Graphen unterstützt, z. B. die Verwendung von `Content-Type: application/n-quads` oder`Content-Type: application/trig`.

In diesem Fall fügt Neptune alle im Text benannten Diagramme und das in der URL angegebene benannte Diagramm hinzu oder aktualisiert sie.

Angenommen, Sie beginnen mit einer leeren Datenbank und senden eine PUT-Anforderung, um ein Upsert für Stimmen in drei Diagramme auszuführen. Das Diagramm mit dem Namen `urn:votes` enthält alle Stimmen aus allen Wahljahren. Die beiden anderen Diagramme mit den Namen `urn:votes:2005` und `urn:votes:2019` enthalten Stimmen aus bestimmten Wahljahren. Die Anforderung und ihre Nutzlast sehen wie folgt aus:

```
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>
```

Nach der Ausführung der Anforderung sehen die Daten in der Datenbank wie folgt aus:

```
<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>
```

Eine weitere mehrdeutige Situation entsteht, wenn in der Anforderungs-URL (mit `PUT`, `POST`, `GET` oder `DELETE`) mehr als ein Diagramm angegeben ist. Beispiel:

```
POST "http://your-Neptune-cluster:port/sparql/gsp/?
graph=urn:votes:2005&graph=urn:votes:2019"
```

Oder:

```
GET "http://your-Neptune-cluster:port/sparql/gsp/?default&graph=urn:votes:2019"
```

In diesem Fall gibt Neptune HTTP 400 mit der Meldung zurück, dass in der Anforderungs-URL nur ein Diagramm angegeben werden darf.

Analysieren der Ausführung Neptune Neptune-Abfragen mit SPARQL erklären

Amazon Neptune hat das SPARQL -Feature explain hinzugefügt. Dieses Feature ist ein Self-Service-Tool, um den Ausführungsansatz der Neptune-Engine zu verstehen. Sie rufen diese Funktion auf, indem Sie einem HTTP-Aufruf, der eine SPARQL-Abfrage sendet, einen `explain`-Parameter hinzufügen.

Die `explain`-Funktion bietet Informationen zur logischen Struktur von Abfrageausführungsplänen. Mit diesen Informationen können Sie potenzielle Engpässe bei der Bewertung und Ausführung identifizieren. Anschließend können Sie Ihre Abfrageausführungspläne mit Abfragetipps verbessern.

Themen

- Funktionsweise der SPARQL-Abfrage-Engine in Neptune
- Wie man SPARQL benutzt, um die Ausführung von Neptune-Abfragen zu analysieren
- Beispiele für den Aufruf von SPARQL erklären in Neptune
- Neptune SPARQL erklärt Operatoren
- Einschränkungen von SPARQL werden in Neptune erklärt

Funktionsweise der SPARQL-Abfrage-Engine in Neptune

Um die Informationen zu verwenden, die das SPARQL-Feature `explain` bereitstellt, müssen Sie einige Details zur Funktionsweise der SPARQL-Abfrage-Engine von Amazon Neptune kennen.

Die Engine übersetzt jede SPARQL-Abfrage in eine Operatoren-Pipeline. Ab dem ersten Operator durchlaufen Zwischenlösungen, die als Bindungslisten bekannt sind, diese Operator Pipeline. Sie können sich eine Bindungsliste als eine Tabelle vorstellen, in der die Tabellenüberschriften eine Teilmenge der in der Abfrage verwendeten Variablen darstellen. Jede Zeile in der Tabelle steht für ein Ergebnis bis zur kompletten Bewertung.

Angenommen, zwei Namespace-Präfixe wurden für unsere Daten definiert,

```
@prefix ex:   <http://example.com> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
```

dann ergäbe sich in diesem Kontext das folgende Beispiel einer einfachen Bindungsliste:

```
?person       | ?firstName
-----------------------------------------------------
ex:JaneDoe    | "Jane"
ex:JohnDoe    | "John"
ex:RichardRoe | "Richard"
```

Für alle drei Personen wird die `?person`-Variable von der Liste an einen Bezeichner der Person und die `?firstName`-Variable an den Vornamen der Person gebunden.

Im Allgemeinen können Variablen ungebunden bleiben, wenn beispielsweise eine `OPTIONAL`-Auswahl einer Variable in einer Abfrage vorliegt, für die kein Wert in den Daten vorhanden ist.

Der `PipelineJoin`-Operator ist ein Beispiel für einen Operator der Neptune-Abfrage-Engine in der `explain`-Ausgabe. Dabei wird als Eingabe ein eingehender Bindungssatz aus dem vorherigen Operator übernommen und einem Dreifachmuster, z. B. `(?person, foaf:lastName, ?lastName)` angefügt. Bei diesem Vorgang werden die Bindungen für die `?person`-Variable im Eingabe-Stream verwendet und im Dreifachmuster ersetzt. Außerdem werden die Triplets in der Datenbank nachgeschlagen.

Bei der Ausführung im Kontext der eingehenden Bindungen aus der vorherigen Tabelle würde `PipelineJoin` die folgenden drei Suchvorgänge bewerten:

```
(ex:JaneDoe,    foaf:lastName, ?lastName)
(ex:JohnDoe,    foaf:lastName, ?lastName)
(ex:RichardRoe, foaf:lastName, ?lastName)
```

Dieser Ansatz wird als as-bound-Bewertung bezeichnet. Die Lösungen dieses Bewertungsprozess werden den eingehenden Lösungen zugeführt, sodass das erkannte `?lastName`-Objekt in den eingehenden Lösungen aufgefüllt wird. Falls Sie einen Nachnamen für alle drei Personen finden, erstellt der Operator eine ausgehende Bindungsliste, die etwa wie folgt aussieht:

```
?person       | ?firstName | ?lastName
---------------------------------------
ex:JaneDoe    | "Jane"     | "Doe"
ex:JohnDoe    | "John"     | "Doe"
ex:RichardRoe | "Richard"  | "Roe"
```

Diese ausgehende Bindungsliste dient anschließend als Eingabe für den nächsten Operator in der Pipeline. Am Ende definiert die Ausgabe des letzten Operators in der Pipeline das Abfrageergebnis.

Operator-Pipelines sind häufig linear. Da bedeutet, jeder Operator gibt Lösungen für einen einzelnen verbundenen Operator weiter. In einigen Fällen kann es jedoch zu komplexen Strukturen kommen. Beispiel: Ein `UNION`-Operator in einer SPARQL-Abfrage wird einer `Copy`-Operation zugeordnet. Dieser Vorgang dupliziert die Bindungen und leitet die Kopien an zwei Unterpläne weiter; einer für die linke Seite und der andere für die rechte Seite der `UNION`.

Weitere Informationen zu Operatoren finden Sie unter `Neptune SPARQL erklärt Operatoren`.

Wie man SPARQL benutzt, um die **`Ausführung`** von Neptune-Abfragen zu analysieren

Das SPARQL-Feature `explain` ist ein Self-Service-Tool in Amazon Neptune, das Ihnen hilft, den Ausführungsansatz der Neptune-Engine besser zu verstehen. Wenn Sie `explain` aufrufen, übergeben Sie einen Parameter in der Form `explain=`*`mode`* an eine HTTP- oder HTTPS-Anforderung.

Der Moduswert kann `static`, `dynamic` oder `details` sein.

- Im statischen Modus gibt `explain` nur die statische Struktur des Abfrageplans aus.
- Im dynamischen Modus enthält `explain` auch dynamische Aspekte des Abfrageplans. Diese Aspekte können ggf. die Anzahl der durch die Operatoren fließenden Zwischenbindungen, das Verhältnis der eingehenden zu den ausgehenden Bindungen sowie die von den Operatoren insgesamt in Anspruch genommene Zeit enthalten.
- Im Detailmodus werden die im `dynamic`-Modus angezeigten Informationen sowie zusätzliche Details wie die tatsächliche SPARQL-Abfragezeichenfolge und die geschätzte Bereichsanzahl für das Muster , das einem Join-Operator zugrunde liegt, von `explain` gedruckt.

Neptune unterstützt die Verwendung von `explain` mit allen drei SPARQL-Abfragezugriffsprotokollen, die in der Spezifikation W3C SPARQL 1.1 Protocol aufgeführt sind:

1. HTTP GET
2. HTTP POST unter Verwendung von Parametern URL-encoded
3. HTTP POST mit Textparametern

Allgemeine Informationen zur Abfrage-Engine von SPARQL finden Sie unter Funktionsweise der SPARQL-Abfrage-Engine in Neptune.

Weitere Informationen über die Art der durch das Aufrufen von SPARQL `explain` erstellten Ausgabe finden Sie unter Beispiele für den Aufruf von SPARQL `erklären` in Neptune.

Beispiele für den Aufruf von SPARQL **`erklären`** in Neptune

Die Beispiele in diesem Abschnitt zeigen die Arten von Ausgaben, die Sie durch das Aufrufen des SPARQL-Features `explain` erzeugen können, um die Abfrageausführung in Amazon Neptune zu analysieren.

Themen

- Informationen zur Explain-Ausgabe
- Beispiel für die Ausgabe im Detailmodus
- Beispiel für die Ausgabe im statischen Modus
- Verschiedene Möglichkeiten für die Parameterverschlüsselung
- Daneben gibt es noch andere Ausgabetypen text/plain
- Beispiel für die SPARQL Explain-Ausgabe, wenn DFE aktiviert ist

Informationen zur Explain-Ausgabe

In diesem Beispiel kennt Jane Doe die beiden Personen John Doe und 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" .
.
```

Wenn Sie den Vornamen aller Personen ermitteln möchten, die Jane Doe kennt, können Sie die folgende Abfrage schreiben:

```
 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"
```

Diese einfache Abfrage gibt Folgendes zurück:

```
firstName
John
Richard
```

Als Nächstes ändern Sie den `curl`-Befehl, um `explain` aufzurufen, indem Sie `-d "explain=dynamic"` hinzufügen und den Standardausgabetyp anstelle von `text/csv` verwenden:

```
 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"
```

Die Abfrage gibt jetzt eine Ausgabe in gut lesbarem ASCII-Format (HTTP-Inhaltstyp `text/plain`) zurück. Dabei handelt es sich um die Standardausgabe:

```
########################################################################################
# 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         #
#########################################################################################
```

Weitere Informationen zu den Operationen in der `Name`-Spalte und ihren Argumenten finden Sie unter explain-Operatoren.

In der folgenden Tabelle wird die Ausgabe Zeile für Zeile beschrieben:

1. Beim ersten Schritt in der Hauptabfrage wird immer der `SolutionInjection`-Operator verwendet, um eine Lösung einzubringen. Die Lösung wird dann durch den Bewertungsprozess zum Endergebnis erweitert.

   In diesem Fall bringt er die sogenannte universelle `{ }`-Lösung ein. Bei `VALUES`-Klauseln oder `BIND` bringt dieser Schritt zu Beginn möglicherweise auch komplexere Variablenbindungen ein.

   Die `Units Out`-Spalte zeigt an, dass diese Einzellösung aus dem Operator eingeht. Die `Out #1`-Spalte gibt den Operator an, in den dieser Operator das Ergebnis einbringt. In diesem Beispiel sind alle Operatoren mit dem folgenden Operator in der Tabelle verbunden.
2. Die zweite Schritt ist ein `PipelineJoin`-Element. Es empfängt als Eingabe die einzelne universelle (vollständig uneingeschränkte) Lösung, die durch den vorherigen Operator (`Units In := 1`) erstellt wurde. Es verbindet sich im Tupel-Muster, das vom `pattern`-Argument definiert wird. Dies entspricht einer einfachen Suche nach dem Muster. In diesem Fall wird die dreifache Muster wie folgt definiert:

   ```
   distinct( ex:JaneDoe, foaf:knows, ?person )
   ```

   Das `joinType := join`-Argument gibt an, dass es sich um eine normale Verbindung handelt (andere Typen umfassen `optional`-Joins, `existence check`-Joins usw.).

   Das `distinct := true`-Argument gibt an, dass Sie nur eindeutige Treffer (keine Duplikate) aus der Datenbank extrahieren, und Sie binden die unterschiedlichen Treffer auf deduplizierte Weise an die `joinProjectionVars := ?person`-Variable.

Die Tatsache, dass der `Units Out`-Spaltenwert 2 lautet, gibt an, dass zwei Lösungen ausgehen. Genauer gesagt handelt es sich dabei um die Bindungen für die `?person`-Variable, mit der die zwei Personen widergespiegelt werden, die Jane Doe laut der Daten kennt:

```
?person
-------------
ex:JohnDoe
ex:RichardRoe
```

3. Die beiden Lösungen aus Stufe 2 gehen als Eingabe (`Units In := 2`) in das zweite `PipelineJoin`-Element ein. Dieser Operator verbindet die beiden vorherigen Lösungen mit dem folgenden Dreifachmuster:

   ```
   distinct(?person, foaf:firstName, ?firstName)
   ```

   Die `?person`-Variable wird mithilfe der eingehenden Lösung des Operators entweder an `ex:JohnDoe` oder `ex:RichardRoe` gebunden. Da `PipelineJoin` den Vornamen John und Richard extrahiert, lauten die beiden ausgehenden Lösungen (Units Out: = 2) wie folgt:

   ```
   ?person       | ?firstName
   --------------------------
   ex:JohnDoe    | John
   ex:RichardRoe | Richard
   ```

4. Der nächste Projection-Operator übernimmt als Eingabe die beiden Lösungen aus Stufe 3 (`Units In := 2`) und projiziert sie zur `?firstName`-Variable. Dadurch entfallen alle anderen Variablenbindungen in den Zuweisungen und die beiden Bindungen (`Units Out := 2`) werden übergeben:

   ```
   ?firstName
   ----------
   John
   Richard
   ```

5. Zur Leistungsverbesserung wird Neptune (wenn möglich) auf internen IDs ausgeführt, die Bedingungen wie URIs und Zeichenfolgenliteralen zugewiesen werden, statt auf den Zeichenfolgen selbst ausgeführt zu werden. Der letzte Operator, `TermResolution`, führt eine Zuweisung aus diesen internen IDs zurück an die entsprechenden Begriffszeichenfolgen durch.

In regelmäßigen (nicht Explain-)Abfragen wird das vom letzten Operator berechnete Ergebnis dann in das angeforderte Serialisierungsformat serialisiert und an den Client gestreamt.

Beispiel für die Ausgabe im Detailmodus

Angenommen, Sie führen die gleiche Abfrage wie zuvor im Detailmodus statt im dynamischen Modus aus:

```
 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"
```

Wie dieses Beispiel zeigt, ist die Ausgabe bis auf einige zusätzliche Details wie der Abfragezeichenfolge am Anfang der Ausgabe und der `patternEstimate`-Anzahl für den `PipelineJoin`-Operator identisch:

```
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         #
################################################################################################
```

Beispiel für die Ausgabe im statischen Modus

Nehmen wir an, Sie führen die gleiche Abfrage wie zuvor im statischen Modus (Standard) statt im dynamischen Modus aus:

```
 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"
```

Wie dieses Beispiel zeigt, ist die Ausgabe identisch, abgesehen davon, dass die letzten drei Spalten ausgelassen werden:

```
################################################################################################
# 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 #
#############################################################################################
```

Verschiedene Möglichkeiten für die Parameterverschlüsselung

Die folgenden Beispielabfragen demonstrieren zwei verschiedene Möglichkeiten beim Aufrufen von SPARQL `explain` Parameter zu verschlüsseln.

Verwenden der URL-Verschlüsselung – Dieses Beispiel verwendet die URL-Parameterverschlüsselung und gibt eine dynamische Ausgabe an:

```
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"
```

Direkte Parameterangabe – Dies entspricht der vorherigen Abfrage, die Parameter werden jedoch direkt über POST übergeben:

```
curl http(s)://your_server:your_port/sparql \
  -d "query=SELECT * WHERE { ?s ?p ?o } LIMIT 1" \
  -d "explain=dynamic"
```

Daneben gibt es noch andere Ausgabetypen text/plain

In den vorherigen Beispielen wird der Standardausgabetyp `text/plain` verwendet. Neptune kann die `explain` SPARQL-Ausgabe auch in zwei anderen MIME-type Formaten formatieren, nämlich und. `text/csv` `text/html` Sie können sie aufrufen, indem Sie den HTTP-`Accept`-Header aufrufen. Dies ist wie folgt mit dem `-H`-Flag in `curl` möglich:

```
  -H "Accept: output type"
```

Hier sind einige Beispiele:

`text/csv`-Ausgabe

Diese Abfrage fordert eine MIME-type CSV-Ausgabe auf, indem Folgendes angegeben wird: `-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"
```

Das CSV-Format ist beim Importieren von Dateien in eine Tabelle oder Datenbank hilfreich, da es die Felder in jeder `explain`-Zeile durch Semikolons (`;`) trennt, wie im folgenden Beispiel dargestellt:

```
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
```

`text/html`-Ausgabe

Wenn Sie `-H "Accept: text/html"` angeben, generiert `explain` eine HTML-Tabelle:

```
<!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>
```

Das HTML wird in einem Browser etwa wie folgt gerendert:

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

Beispiel für die SPARQL **`Explain-Ausgabe`**, wenn DFE aktiviert ist

Dies ist ein Beispiel für eine SPARQL-Ausgabe für `explain` bei Aktivierung der alternativen DFE-Abfrage-Engine:

```
#################################################################################################
# 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 erklärt Operatoren

Die folgenden Abschnitte beschreiben die Operatoren und Parameter für das SPARQL-Feature `explain`, die zurzeit in Amazon Neptune verfügbar sind.

> **Important**
>
> Das SPARQL–Feature `explain` wird zurzeit weiter optimiert. Die hier dokumentierten Operatoren und Parameter unterliegen möglicherweise Änderungen in zukünftigen Versionen.

Themen

- Aggregationsoperator
- ConditionalRouting-Operator
- Operator kopieren
- DefNode-Operator
- Eindeutiger Operator
- Föderationsbetreiber
- Operator filtern

- HashIndexBuild-Operator
- HashIndexJoin-Operator
- MergeJoin-Operator
- NamedSubquery-Operator
- PipelineJoin-Operator
- PipelineCountJoin-Operator
- PipelinedHashIndexJoin-Operator
- Projektionsoperator
- PropertyPath-Operator
- TermResolution-Operator
- Slice-Operator
- SolutionInjection-Operator
- Operator sortieren
- VariableAlignment-Operator

`Aggregationsoperator`

Führt eine oder mehrere Aggregationen durch, mit denen die Semantik der SPARQL-Aggregationsoperatoren wie `count`, `max`, `min`, `sum` und so weiter implementiert wird.

`Aggregation` verfügt über optionale Gruppierung mithilfe von `groupBy`-Klauseln sowie über optionale `having`-Einschränkungen.

Argumente

- `groupBy` – (Optional) Stellt die `groupBy`-Klausel zur Angabe der Abfolge von Ausdrücken je nach Gruppierung der eingehenden Lösungen bereit.
- `aggregates` – (Erforderlich) Gibt eine geordnete Liste von Aggregationsausdrücke an.
- `having` – (Optional) Fügt Filtern für Gruppen Einschränkungen hinzu wie durch die `having`-Klausel in der SPARQL-Abfrage angegeben.

`ConditionalRouting`-Operator

Leitet eingehende Lösungen auf Grundlage einer bestimmten Bedingung weiter. Lösungen, die die Bedingung erfüllen, werden an die Operator-ID weitergeleitet, die von `Out #1` verwiesen wurde.

Dagegen werden Lösungen, die die Bedingung nicht erfüllen, an den Operator weitergeleitet, der von `Out #2` verwiesen wird.

Argumente

- `condition` – (Erforderlich) Die Weiterleitungsbedingung.

Operator kopieren

Delegiert den Lösungs-Stream auf die vom angegebenen Modus bestimmte Weise.

Modi

- `forward` – Leitet die Lösungen an den Downstream-Operator weiter, wie von `Out #1` identifiziert.
- `duplicate` – Dupliziert die Lösungen und leitet sie an beide Operatoren weiter, die von `Out #1` und `Out #2` identifiziert werden.

`Copy` hat keine Argumente.

DefNode-Operator

Dieser Operator ist eine Abstraktion des Plans, der von der alternativen DFE-Abfrage-Engine ausgeführt wird. Der detaillierte DFE-Plan wird in den Argumenten für diesen Operator beschrieben. Das Argument ist zurzeit überladen, indem es die detaillierten Laufzeitstatistiken des DFE-Plans enthält. Es enthält die Zeit, die DFE für die einzelnen Schritte der Abfrageausführung benötigt hat.

Der logisch optimierte abstrakte Syntaxbaum (AST) für den DFE-Abfrageplan wird mit Informationen zu den Operatortypen gedruckt, die bei der Planung berücksichtigt wurden, und zu den Kosten für die Ausführung der Operatoren im besten und schlechtesten Fall. Der AST besteht zurzeit aus den folgenden Knotentypen:

- `DFEJoinGroupNode` – Stellt die Vereinigung eines oder mehrerer `DFEPatternNodes` dar.
- `DFEPatternNode` – Kapselt ein zugrundeliegendes Muster mit den übereinstimmenden Tupeln, die aus der zugrunde liegenden Datenbank projiziert werden.

Der Unterabschnitt (`Statistics & Operator histogram`) enthält Details zur Ausführungszeit des `DataflowOp`-Plans und zur Aufschlüsselung der von den einzelnen Operatoren genutzten CPU-Zeit. Unterhalb befindet sich eine Tabelle, in der detaillierte Laufzeitstatistiken des von der DFE ausgeführten Plans ausgegeben werden.

> **Note**
>
> Da es sich bei der DFE-Unterstützung für SPARQL um eine experimentelle Funktion handelt, kann sich das genaue Format der Ausgabe ändern. `explain`

`Eindeutiger Operator`

Berechnet die unterschiedliche Projektion bei einer Teilmenge der Variablen, sodass Duplikate vermieden werden. Dies hat zur Folge, dass die Anzahl eingehender Lösungen größer oder gleich der Anzahl der ausgehenden Lösungen ist.

Argumente

- `vars` – (Erforderlich) Die Variablen, auf die die `Distinct`-Projektion angewendet werden soll.

`Föderationsbetreiber`

Übergibt eine angegebene Abfrage an einen angegebenen Remote-SPARQL-Endpunkt

Argumente

- `endpoint` – (Erforderlich) Die Endpunkt-URL in der SPARQL-Anweisung `SERVICE`. Dies kann eine konstante Zeichenfolge sein, oder, wenn der Abfrageendpunkt basierend auf einer Variablen innerhalb derselben Abfrage bestimmt wird, kann es sich um den Variablennamen handeln.
- `query` – (Erforderlich) Die rekonstruierte Abfragezeichenfolge, die an den Remote-Endpunkt gesendet werden soll. Die Engine fügt dieser Abfrage Standardpräfixe hinzu, auch wenn der Client keine angibt.
- `silent` – (Erforderlich) Ein boolescher Wert, der angibt, ob das Schlüsselwort `SILENT` nach dem Schlüsselwort angezeigt wurde. `SILENT` weist die Engine an, bei einem Fehlschlag von `SERVICE` nicht die gesamte Abfrage fehlschlagen zu lassen.

Operator `filtern`

Filtert die eingehenden Lösungen. Nur Lösungen, die die Filterbedingung erfüllen, werden an die Upstream-Operator weitergeleitet. Alle anderen werden gelöscht.

Argumente

- `condition` – (Erforderlich) Die Filterbedingung.

`HashIndexBuild`-Operator

Übernimmt eine Liste der Bindungen und spult sie in einem Hash-Index, dessen Name vom `solutionSet`-Argument definiert wird. In der Regel führen nachfolgende Operatoren Joins für diesen Lösungssatz aus und verweisen ihn mit diesem Namen.

Argumente

- `solutionSet` – (Erforderlich) Der Name des Hash-Index-Lösungssatzes.
- `sourceType` – (Required) Der Typ der Quelle, aus der die Bindungen für die Speicherung im Hash-Index abgerufen werden:
  - `pipeline` – Spult die eingehenden Lösungen aus dem Downstream-Operator in der Operator-Pipeline in den Hash-Index.
  - `binding set` – Spult den vom `sourceBindingSet`-Argument angegebenen festen Bindungssatz in den Hash-Index.
- `sourceBindingSet` – (Optional) Wenn das Argument `sourceType` den Wert `binding set` hat, gibt dieses Argument den statischen Bindungssatz an, der in den Hash-Index gespult werden soll.

`HashIndexJoin`-Operator

Verbindet die eingehenden Lösungen für den Lösungssatz des Hash-Index, der vom `solutionSet`-Argument identifiziert wird.

Argumente

- `solutionSet` – (Erfoderlich) Name des Lösungssatzes, mit dem der Join durchgeführt werden soll. Dies muss ein Hash-Index sein, der in einem vorherigen Schritt mithilfe des Operators `HashIndexBuild` konstruiert wurde.
- `joinType` – (Erforderlich) Der Typ des Join, der ausgeführt werden soll:
  - `join` – Ein normales Join, für das eine exakte Übereinstimmung aller geteilten Variablen erforderlich ist.
  - `optional` – Ein `optional`-Join, für das die SPARQL-`OPTIONAL`-Operatorsemantik verwendet wird.

  - `minus` – Eine `minus`-Operation behält eine Zuordnung, für die kein Join-Partner vorhanden ist, mithilfe der SPARQL-`MINUS`-Operatorsemantik bei.
  - `existence check` – Prüft, ob ein Join-Partner vorhanden ist, und bindet die Variable `existenceCheckResultVar`an das Ergebnis dieser Prüfung.
- `constraints` – (Optional) Zusätzliche Join-Einschränkungen, die während des Join in Betracht gezogen werden. Joins, die diese Einschränkungen nicht erfüllen, werden verworfen.
- `existenceCheckResultVar` – (Optional) Wird nur für Joins verwendet, in denen `joinType` gleich `existence check` ist (siehe zuvor das Argument `joinType`).

`MergeJoin`-Operator

Ein Zusammenführungs-Join über mehrere Lösungen, wie es vom `solutionSets`-Argument identifiziert wird.

Argumente

- `solutionSets` – (Erforderlich) Die Lösungssätze, für die ein Join ausgeführt werden soll.

`NamedSubquery`-Operator

Löst eine Bewertung der Unterabfrage aus, die vom `subQuery`-Argument identifiziert wird, und spult das Ergebnis in den vom `solutionSet`-Argument angegebenen Lösungssatz. Die eingehenden Lösungen für den Operator werden an die Unterabfrage und dann an den nächsten Operator weitergeleitet.

Argumente

- `subQuery` – (Erforderlich) Der Name der Unterabfrage, die ausgewertet werden soll. Die Unterabfrage wird in der Ausgabe explizit gerendert.
- `solutionSet` – (Erforderlich) Der Name des Lösungssatzes, in dem das Ergebnis der Unterabfrage gespeichert werden soll.

`PipelineJoin`-Operator

Erhält die Ausgabe des vorherigen Operators als Eingabe und verbindet sie mit dem Tupel-Muster, das vom `pattern`-Argument definiert wird.

Argumente

- `pattern` – (Erforderlich) Das Muster, das die Form eines Subjekt-Prädikat-Objekts annimmt, und optional das -graph-Tupel, das dem Join zugrunde liegt. Wenn `distinct` für das Muster angegeben wird, extrahiert der Join nur unterschiedliche Lösungen aus den Projektionsvariablen, die vom `projectionVars`-Argument angegeben werden, anstatt alle passenden Lösungen zu extrahieren.
- `inlineFilters` – (Optional) Eine Reihe von Filtern, die auf die Variablen im Muster angewendet werden sollen. Das Muster wird in Verbindung mit diesen Filtern ausgewertet.
- `joinType` – (Erforderlich) Der Typ des Join, der ausgeführt werden soll:
  - `join` – Ein normales Join, für das eine exakte Übereinstimmung aller geteilten Variablen erforderlich ist.
  - `optional` – Ein `optional`-Join, für das die SPARQL-`OPTIONAL`-Operatorsemantik verwendet wird.
  - `minus` – Eine `minus`-Operation behält eine Zuordnung, für die kein Join-Partner vorhanden ist, mithilfe der SPARQL-`MINUS`-Operatorsemantik bei.
  - `existence check` – Prüft, ob ein Join-Partner vorhanden ist, und bindet die Variable `existenceCheckResultVar`an das Ergebnis dieser Prüfung.
- `constraints` – (Optional) Zusätzliche Join-Einschränkungen, die während des Join in Betracht gezogen werden. Joins, die diese Einschränkungen nicht erfüllen, werden verworfen.
- `projectionVars` – (Optional) Die Projektionsvariablen. Werden in Verbindung mit `distinct := true` verwendet, um das Extrahieren unterschiedlicher Projektionen aus einem bestimmten Variablensatz zu erzwingen.
- `cutoffLimit` – (Optional) Ein Abschnittslimit für die Anzahl der extrahierten Join-Partner. Obwohl standardmäßig kein Limit vorliegt, können Sie diesen Wert bei Ausführen von Joins auf 1 festlegen, um `FILTER (NOT) EXISTS`-Klauseln zu implementieren, sofern es ausreicht zu beweisen oder zu wiederlegen, dass ein Join-Partner vorhanden ist.

`PipelineCountJoin`-Operator

Eine Variante von `PipelineJoin`. Anstatt einen Join durchzuführen, werden einfach die übereinstimmenden Join-Partner gezählt und die Anzahl wird an die Variable gebunden, die vom `countVar`-Argument angegeben wird.

Argumente

- `countVar` – (Erforderlich) Die Variable, über die das Ergebnis der Zählung, d. h. die Anzahl der Join-Partner, gebunden werden soll.
- `pattern` – (Erforderlich) Das Muster, das die Form eines Subjekt-Prädikat-Objekts annimmt, und optional das -graph-Tupel, das dem Join zugrunde liegt. Wenn `distinct` für das Muster angegeben wird, extrahiert der Join nur unterschiedliche Lösungen aus den Projektionsvariablen, die vom `projectionVars`-Argument angegeben werden, anstatt alle passenden Lösungen zu extrahieren.
- `inlineFilters` – (Optional) Eine Reihe von Filtern, die auf die Variablen im Muster angewendet werden sollen. Das Muster wird in Verbindung mit diesen Filtern ausgewertet.
- `joinType` – (Erforderlich) Der Typ des Join, der ausgeführt werden soll:
  - `join` – Ein normales Join, für das eine exakte Übereinstimmung aller geteilten Variablen erforderlich ist.
  - `optional` – Ein `optional`-Join, für das die SPARQL-`OPTIONAL`-Operatorsemantik verwendet wird.
  - `minus` – Eine `minus`-Operation behält eine Zuordnung, für die kein Join-Partner vorhanden ist, mithilfe der SPARQL-`MINUS`-Operatorsemantik bei.
  - `existence check` – Prüft, ob ein Join-Partner vorhanden ist, und bindet die Variable `existenceCheckResultVar`an das Ergebnis dieser Prüfung.
- `constraints` – (Optional) Zusätzliche Join-Einschränkungen, die während des Join in Betracht gezogen werden. Joins, die diese Einschränkungen nicht erfüllen, werden verworfen.
- `projectionVars` – (Optional) Die Projektionsvariablen. Werden in Verbindung mit `distinct := true` verwendet, um das Extrahieren unterschiedlicher Projektionen aus einem bestimmten Variablensatz zu erzwingen.
- `cutoffLimit` – (Optional) Ein Abschnittslimit für die Anzahl der extrahierten Join-Partner. Obwohl standardmäßig kein Limit vorliegt, können Sie diesen Wert bei Ausführen von Joins auf 1 festlegen, um `FILTER (NOT) EXISTS`-Klauseln zu implementieren, sofern es ausreicht zu beweisen oder zu wiederlegen, dass ein Join-Partner vorhanden ist.

`PipelinedHashIndexJoin`-Operator

Dies ist ein All-in-One-Build-Hash-Index und Join-Operator. Er nimmt eine Liste von Bindungen, spult sie in einen Hash-Index und verknüpft dann die eingehenden Lösungen anhand des Hash-Indexes.

Argumente

- `sourceType` – (Required) Der Typ der Quelle, aus der die Bindungen für die Speicherung im Hash-Index abgerufen werden:
  - `pipeline` – Lässt `PipelinedHashIndexJoin` die eingehenden Lösungen aus dem Downstream-Operator in der Operator-Pipeline in den Hash-Index spulen.
  - `binding set` – Lässt `PipelinedHashIndexJoin` den vom `sourceBindingSet`-Argument angegebenen festen Bindungssatz in den Hash-Index spulen.
- `sourceSubQuery` – (Optional) Wenn das Argument `sourceType` den Wert `pipeline` hat, gibt dieses Argument die Unterabfrage an, die ausgewertet und in den Hash-Index gespoolt wird.
- `sourceBindingSet` – (Optional) Wenn das Argument `sourceType` den Wert `binding set` hat, gibt dieses Argument den statischen Bindungssatz an, der in den Hash-Index gespult werden soll.
- `joinType` – (Erforfderlich) Der Typ des Join, der ausgeführt werden soll:
  - `join` – Ein normales Join, für das eine exakte Übereinstimmung aller geteilten Variablen erforderlich ist.
  - `optional` – Ein `optional`-Join, für das die SPARQL-`OPTIONAL`-Operatorsemantik verwendet wird.
  - `minus` – Eine `minus`-Operation behält eine Zuordnung, für die kein Join-Partner vorhanden ist, mithilfe der SPARQL-`MINUS`-Operatorsemantik bei.
  - `existence check` – Prüft, ob ein Join-Partner vorhanden ist, und bindet die Variable `existenceCheckResultVar`an das Ergebnis dieser Prüfung.
- `existenceCheckResultVar` – (Optional) Wird nur für Joins verwendet, in denen `joinType` gleich `existence check` ist (siehe zuvor das Argument joinType).

`Projektionsoperator`

Projiziert über eine Teilmenge der Variablen. Die Anzahl der eingehenden Lösungen entspricht der Anzahl der ausgehenden Lösungen, aber die Form der Lösung ist je nach Moduseinstellung unterschiedlich.

Modi

- `retain` – In Lösungen werden nur die Variablen beibehalten, die vom Argument `vars` angegeben werden.

- `drop` – Alle Variablen, die vom Argument `vars` angegeben werden, werden entfernt.

Argumente

- `vars` – (Erforderlich) Die Variablen, die je nach Moduseinstellung beibehalten oder entfernt werden sollen.

`PropertyPath`-Operator

Aktiviert rekursive Eigenschaftspfade wie `+` oder `*`. Neptune implementiert einen Festkomma-Iterationsansatz, der auf einer durch das Argument `iterationTemplate` angegebenen Vorlage basiert. Bekannte Variablen der linken oder rechten Seite werden in der Vorlage solange für jede Iteration mit festem Punkt gebunden, bis keine neuen Lösungen gefunden werden können.

Argumente

- `iterationTemplate` – (Erforderlich) Der Name der Vorlage für die Unterabfrage, die für die Implementierung der Iteration mit festem Punkt verwendet wird.
- `leftTerm` – (Erforderlich) Der Begriff (Variable oder Konstante) auf der linken Seite des Eigenschaftspfads.
- `rightTerm` – (Erforderlich) Der Begriff (Variable oder Konstante) auf der rechten Seite des Eigenschaftspfads.
- `lowerBound` – (Erforderlich) Die untere Grenze für die Iteration mit festem Punkt (entweder `0` für `*`-Abfragen oder 1 für +-Abfragen).

`TermResolution`-Operator

Übersetzt interne Zeichenfolgekennungswerte zurück in ihre entsprechenden externen Zeichenfolgen oder übersetzt externe Zeichenfolgen in internen Kennungswerte. Dies ist vom Modus abhängig.

Modi

- `value2id` – Ordnet Begriffe wie Literale und URIs den entsprechenden internen ID-Werten zu (Kodierung zu internen Werten).
- `id2value` – Ordnet interne ID-Werte den entsprechenden Begriffen wie Literalen und URIs zu (Dekodierung interner Werte).

Argumente

- `vars` – (Erforderlich) Gibt die Variablen an, deren Zeichenfolgen oder interne Zeichenfolgen-IDs zugeordnet werden sollen.

`Slice-Operator`

Implementiert ein Slice mithilfe der Semantik der `LIMIT`- und `OFFSET`-Klauseln von SPARQL über den eingehenden Lösungs-Stream.

Argumente

- `limit` – (Optional) Eine Einschränkung für die Lösungen, die weitergeleitet werden sollen.
- `offset` – (Optional) Der Versatz, mit dem Lösungen für die Weiterleitung ausgewertet werden.

`SolutionInjection`-Operator

Empfängt keine Eingabe. Bringt statische Lösungen in den Abfrageplan ein und nimmt sie in das `solutions`-Argument auf.

Anfragepläne beginnen immer mit dieser statischen Injektion. Wenn einzubringende statische Lösungen aus der Abfrage abgeleitet werden können, indem verschiedene Quellen statischer Bindungen (z. B. aus `VALUES`- oder `BIND`-Klauseln) abgeleitet werden können, bringt der `SolutionInjection`-Operator diese abgeleiteten statischen Lösungen ein. Im einfachsten Fall spiegelt dies Bindungen wider, die von einer äußeren `VALUES`-Klausel angedeutet werden.

Wenn keine statischen Lösungen aus der Abfrage abgeleitet werden können, bringt `SolutionInjection` die leere so genannte universelle Lösung ein, die erweitert und im Abfragebewertungsprozess multipliziert wird.

Argumente

- `solutions` – (Erforderlich) Die vom Operator eingefügte Lösungsabfolge.

Operator `sortieren`

Sortiert den Lösungssatz mithilfe bestimmter Sortierbedingungen.

Argumente

- `sortOrder` – (Erforderlich) Eine geordnete Liste von Variablen mit jeweils einer `ASC`-ID (aufsteigenden) oder einer `DESC`-ID (absteigend), die zum Sortieren des Lösungssatzes verwendet wird.

`VariableAlignment`-Operator

Prüft Lösungen einzeln und führt bei jeder davon eine Ausrichtung über zwei Variablen durch: ein angegebenes `sourceVar`-Element und ein angegebenes `targetVar`-Element.

Wenn `sourceVar` und `targetVar` in einer Lösung denselben Wert haben, gilt die Variablen als ausgerichtet und die Lösung wird weitergeleitet. Dabei wird das redundante `sourceVar`-Element heraus projiziert.

Wenn die Variablen an unterschiedliche Werte gebunden werden, wird die Lösung vollständig herausgefiltert.

Argumente

- `sourceVar` – (Erforderlich) Die Quellvariable, die mit der Zielvariablen verglichen werden soll. Wenn die Ausrichtung in einer Lösung erfolgreich ist, die beiden Variablen also den gleichen Wert haben, wird die Quellvariable heraus projiziert.
- `targetVar` – (Erforderlich) Die Zielvariable, mit der die Quellvariable verglichen werden soll. Wird auch beibehalten, wenn die Ausrichtung erfolgreich ist.

Einschränkungen von SPARQL werden in **`Neptune erklärt`**

Die Version der des Neptune-SPARQL–Features `explain` unterliegt den folgenden Einschränkungen.

Neptune unterstützt explain zurzeit nur in SPARQL SELECT-Abfragen

Weitere Informationen zum Bewertungsprozess für andere Abfrageformulare, z. B. `ASK`-, `CONSTRUCT`-, `DESCRIBE`-, und `SPARQL UPDATE`-Abfragen, erhalten Sie, wenn Sie diese Abfragen in eine SELECT-Abfrage umwandeln. Überprüfen die entsprechende SELECT-Abfrage anschließend mit `explain`.

Beispiel: Wenn Sie `explain`-Informationen zu einer `ASK WHERE {...}`-Abfrage erhalten möchten, führen Sie die entsprechende `SELECT WHERE {...} LIMIT 1`-Abfrage mit `explain` aus.

Bei einer `CONSTRUCT {...} WHERE {...}`-Abfrage löschen Sie den `CONSTRUCT {...}`-Teil und führen eine `SELECT`-Abfrage mit `explain` in der zweiten `WHERE {...}`-Klausel durch. Die Auswertung der zweiten `WHERE`-Klausel zeigt in der Regel die wichtigsten Herausforderungen bei der Verarbeitung der `CONSTRUCT`-Abfrage, da Lösungen, die aus dem zweiten `WHERE` in die `CONSTRUCT`-Vorlage eingehen in der Regel einfach nur ersetzt werden müssen.

Explain-Operatoren können in zukünftigen Versionen geändert werden

Die SPARQL `explain`-Operatoren und ihre Parameter können in zukünftigen Versionen geändert werden.

Die Explain-Ausgabe kann in zukünftigen Versionen geändert werden

Beispiel: Spaltenüberschriften können geändert werden und den Tabellen werden möglicherweise weitere Spalten hinzugefügt.

SPARQL-Verbundabfragen in Neptune mit der SERVICE-Erweiterung

Amazon Neptune unterstützt die SPARQL-Verbundabfrageerweiterung vollständig, die das Schlüsselwort `SERVICE` verwendet. (Weitere Informationen finden Sie unter SPARQL 1.1-Verbundabfrage.)

Das `SERVICE`-Schlüsselwort weist die SPARQL-Abfrage-Engine an, einen Teil der Abfrage für einen Remote-SPARQL-Endpunkt auszuführen und das endgültige Abfrageergebnis zu erstellen. Es sind nur `READ`-Operationen möglich. Die Operationen `WRITE` und `DELETE` werden nicht unterstützt. Neptune kann Verbundabfragen nur für SPARQL-Endpunkte ausführen, auf die in dessen Virtual Private Cloud (VPC) zugegriffen werden kann. Sie können jedoch auch einen Reverse-Proxy in der VPC verwenden, damit auf eine externe Datenquelle innerhalb der VPC zugegriffen werden kann.

> **Note**
>
> Wenn SPARQL `SERVICE` für die Verbindung einer Abfrage mit zwei oder mehr Neptune-Clustern in derselben VPC verwendet wird, müssen die Sicherheitsgruppen so konfiguriert werden, dass alle diese Neptune-Cluster miteinander kommunizieren können.

> **Important**
>
> SPARQL 1.1 Federation sendet Serviceanfragen in Ihrem Namen, wenn Abfragen und Parameter an externe SPARQL-Endpunkte übergeben werden. Es liegt in Ihrer Verantwortung, zu überprüfen, ob die externen SPARQL-Endpunkte die Datenverarbeitungs- und Sicherheitsanforderungen Ihrer Anwendung erfüllen.

Beispiel für eine Neptune-Verbundabfrage

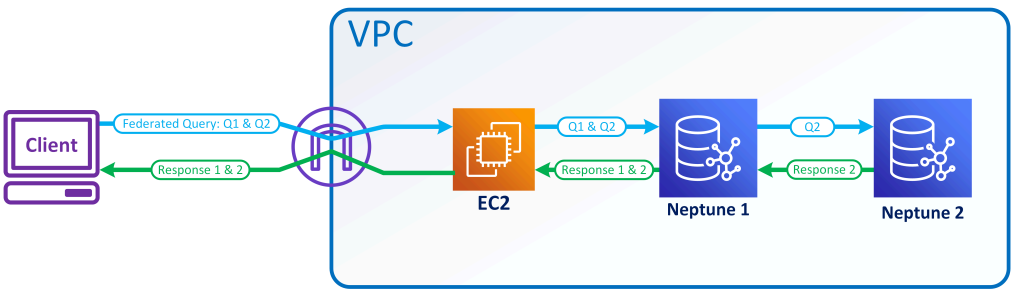
Das folgende einfache Beispiel zeigt, wie SPARQL-Verbundabfragen funktionieren.

Angenommen, ein Kunde sendet die folgende Anfrage an: Neptune-1`http://neptune-1:8182/sparql`

```
SELECT * WHERE {
   ?person rdf:type foaf:Person .
   SERVICE <http://neptune-2:8182/sparql> {
       ?person foaf:knows ?friend .
    }
}
```

1. Neptune-1wertet das erste Abfragemuster (Q-1) aus`?person rdf:type foaf:Person`, verwendet die Ergebnisse zur Auflösung von `?person` in Q-2(`?person foaf:knows ?friend`) und leitet das resultierende Muster an Neptune-2at weiter. `http://neptune-2:8182/sparql`
2. Neptune-2wertet die Ergebnisse aus Q-2und sendet sie zurück an. Neptune-1
3. Neptune-1verbindet die Lösungen für beide Muster und sendet die Ergebnisse an den Kunden zurück.

Dies wird im folgenden Diagramm illustriert:



> **Note**
>
> Standardmäßig legt der Optimierer den Punkt in der Abfrageausführung fest, an dem die Anweisung `SERVICE` ausgeführt wird. Sie können diese Platzierung mittels des Abfragehinweises joinOrder überschreiben.

Zugriffskontrolle für Verbundabfragen in Neptune

Neptune verwendet AWS Identity and Access Management (IAM) für die Authentifizierung und Autorisierung. Die Zugriffskontrolle für eine Verbundabfrage kann mehr als eine Neptune-DB-Instance umfassen. Diese Instances haben möglicherweise unterschiedliche Anforderungen für die Zugriffskontrolle. Unter bestimmten Umständen kann dies Ihre Fähigkeit zum Erstellen einer Verbundabfrage einschränken.

Betrachten Sie das einfache Beispiel aus dem vorherigen Abschnitt. Neptune-1ruft Neptune-2mit denselben Anmeldeinformationen auf, mit denen sie aufgerufen wurden.

- Wenn Neptune-1eine IAM-Authentifizierung und -Autorisierung erforderlich ist, Neptune-2dies aber nicht der Fall ist, benötigen Sie lediglich die entsprechenden IAM-Berechtigungen, Neptune-1um die Verbundabfrage durchzuführen.
- Wenn Neptune-2beide eine IAM-Authentifizierung Neptune-1und -Autorisierung erfordern, müssen Sie IAM-Berechtigungen für beide Datenbanken anhängen, um die Verbundabfrage durchzuführen. Beide Cluster müssen sich außerdem im selben AWS Konto und in derselben Region befinden. Cross-region and/or Kontenübergreifende Verbundabfragearchitekturen werden derzeit nicht unterstützt.
- In dem Fall, wo Neptune-1es nicht ist, IAM-enabled aber Neptune-2ist, können Sie jedoch keine Verbundabfrage stellen. Der Grund dafür ist, dass Neptune-1Sie Ihre IAM-Anmeldeinformationen nicht abrufen und an die Autorisierung des zweiten Teils der Abfrage weitergeben können. Neptune-2

Tools für die Visualisierung von Diagrammen für Neptune

Zusätzlich zu den Visualisierungsfunktionen, die in Neptune Graph-Notebooks integriert sind, können Sie auch Lösungen von AWS Partnern und Drittanbietern zur Visualisierung von in Neptune gespeicherten Daten verwenden.

Komplexe Diagrammvisualisierungen können Datenwissenschaftlern, Managern und anderen Rollen in Organisationen helfen, Diagrammdaten interaktiv zu untersuchen, ohne komplexe Abfragen zu schreiben.

Themen

- Open-Source-basierter Graph-Explorer
- Tom Sawyer Software
- Cambridge Intelligence
- Graphistry
- metaphacts
- G.V() Graph-Datenbank-Client
- Linkurious
- Graph.Build

Open-Source-basierter Graph-Explorer

Graph-explorerist ein Open-Source-Low-Code-Tool zur visuellen Erkundung von Grafikdaten, das unter der Lizenz erhältlich ist. Apache-2.0 Damit können Sie entweder LPG-Daten (Labeled Property Graphs) oder Resource Description Framework (RDF) in einer Graphdatenbank durchsuchen, ohne Graphabfragen schreiben zu müssen. Graph-explorer soll Datenwissenschaftlern, Geschäftsanalysten und anderen Rollen in einem Unternehmen helfen, Grafikdaten interaktiv zu untersuchen, ohne eine Graphabfragesprache lernen zu müssen.

Graph-explorer stellt eine React-based Webanwendung bereit, die als Container zur Visualisierung von Grafikdaten eingesetzt werden kann. Sie können eine Verbindung zu Amazon Neptune oder zu anderen Graphdatenbanken herstellen, die einen Apache TinkerPop Gremlin- oder SPARQL 1.1-Endpunkt bereitstellen.

- Mithilfe der facettierten Filter können Sie schnell eine Übersicht über die Daten anzeigen oder die Daten durchsuchen, indem Sie Text in die Suchleiste eingeben.
- Sie können außerdem Knoten- und Edge-Verbindungen interaktiv untersuchen. Sie können Knotennachbarn anzeigen, um Beziehungen zwischen Objekten zu untersuchen. Anschließend können Sie einen Drilldown zu Kanten und Eigenschaften ausführen.
- Sie können auch das Layout, die Farben und die Symbole des Diagramms anpassen und festlegen, welche Standardeigenschaften für Knoten und Kanten angezeigt werden sollen. Für RDF-Diagramme können Sie auch Namespaces für Ressourcen-URIs anpassen.
- Für Berichte und Präsentationen mit Diagrammdaten können Sie Ansichten konfigurieren und speichern, die Sie in einem hochauflösenden PNG-Format erstellt haben. Sie können die zugehörigen Daten zur weiteren Verarbeitung auch in eine CSV- oder JSON-Datei herunterladen.

Verwenden von Graph-Explorer in einem Neptun-Graph-Notebook

Der einfachste Weg, Graph-Explorer mit Neptune zu verwenden, ist in einem Neptun-Graph-Notebook.

Wenn Sie die Neptune-Workbench verwenden, um ein Neptune-Notebook zu hosten, wird Graph-Explorer automatisch mit dem Notebook bereitgestellt und mit Neptune verbunden.

Nachdem Sie ein Notebook erstellt haben, rufen Sie die Neptune-Konsole auf, um Graph-Explorer zu starten:

1. Gehen Sie zu Neptune.
2. Wählen Sie unter Notebooks Ihr Notebook aus.
3. Wählen Sie unter „Aktionen“ die Option Graph Explorer öffnen aus.

So führen Sie Graph-Explorer in Amazon ECS auf AWS Fargate und verbinde dich mit Neptune

Sie können auch das Graph-Explorer-Docker-Image erstellen und es auf einem lokalen Computer oder einem gehosteten Service wie Amazon Elastic Compute Cloud (Amazon EC2) oder Amazon Elastic Container Service (Amazon ECS) ausführen, wie im Abschnitt Erste Schritte der Read-Me-Datei im Graph-Explorer-Projekt erklärt. GitHub

Als Beispiel stellt der folgende Abschnitt eine Schritt-für-Schritt-Anleitung für die Ausführung von Graph-Explorer in Amazon ECS auf AWS Fargate bereit:

1. Erstellen Sie eine neue IAM-Rolle und fügen Sie diese Richtlinien an:

   - AmazonECSTaskExecutionRolePolicy
   - CloudWatchLogsFullAccess

   Halten Sie den Rollennamen griffbereit, da Sie ihn in Kürze benötigen.

2. Erstellen Sie einen Amazon-ECS-Cluster, dessen Infrastruktur auf FARGATE festgelegt ist und der die folgenden Netzwerkoptionen verwendet:

   - `VPC`: Legen Sie diese Option auf die VPC fest, in der sich Ihre Neptune-Datenbank befindet.
   - `Subnets`: Legen Sie diese Option auf die öffentlichen Subnetze dieser VPC fest (entfernen Sie alle anderen Subnetze).

3. Erstellen Sie eine neue JSON-Aufgabendefinition wie folgt:

```
{
  "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. Starten Sie unter Verwendung der Standardeinstellungen eine neue Aufgabe, ausgenommen die folgenden Felder:

   - Umgebung
     - Berechnungsoptionen => Starttyp
   - Bereitstellungskonfiguration
     - Anwendungstyp => Aufgabe
     - Familie => *(your new JSON task definition)*

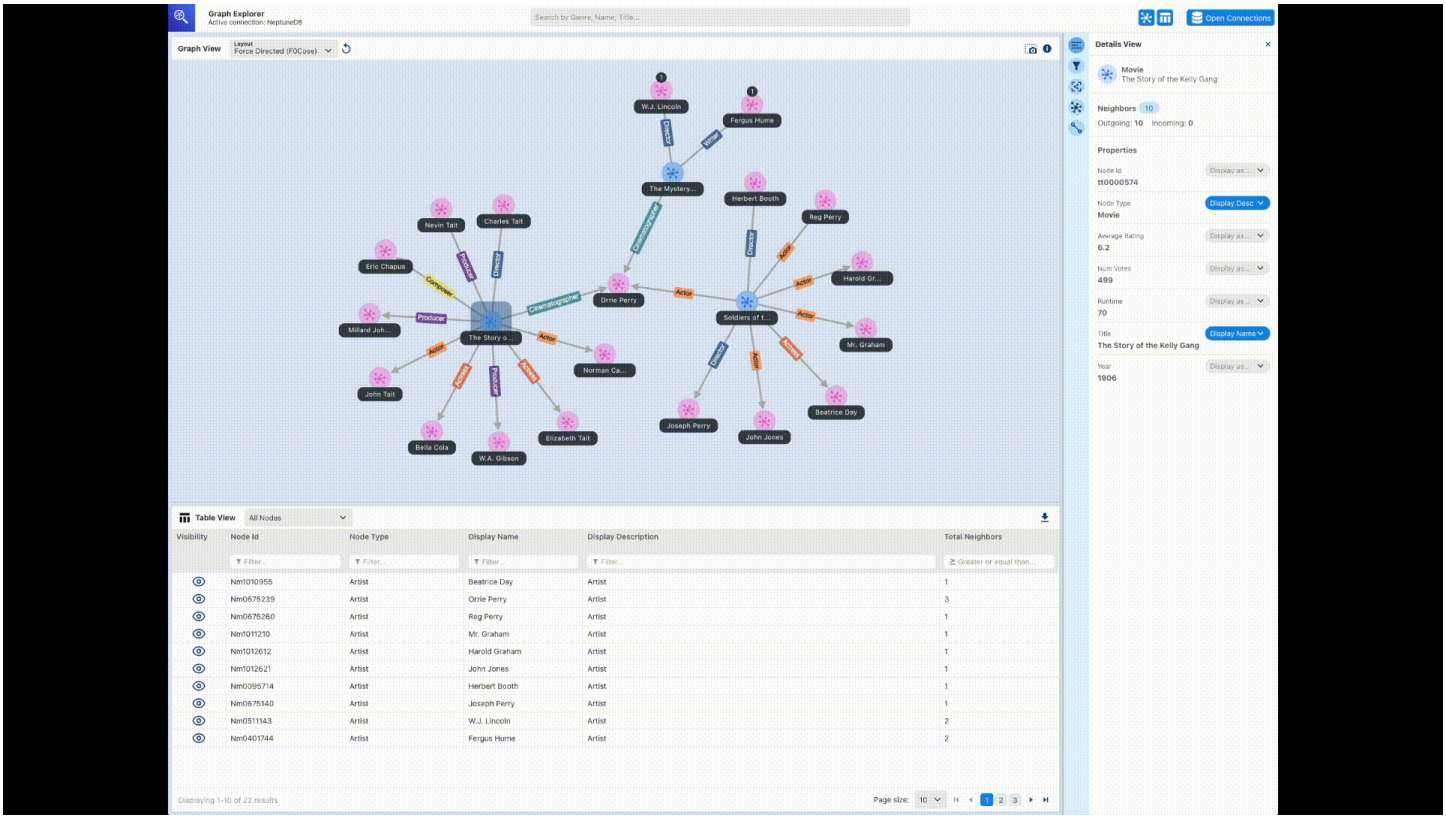
    - Überarbeitung => *(latest)*
  - Netzwerkfunktionen
    - VPC => *(the Neptune VPC you want to connect to)*
    - Subnetze => *(ONLY the public subnets of the VPC– remove all others)*
    - Sicherheitsgruppe => Errstellen Sie eine neue Sicherheitsgruppe
    - Name der Sicherheitsgruppe => Graph-Explorer
    - Beschreibung der Sicherheitsgruppe = Sicherheitsgruppe für den Zugriff auf Graph-Explorer
    - Regeln für den eingehenden Datenverkehr für Sicherheitsgruppen =>
      1. 80 Anywhere
      2. 443 Anywhere
5. Wählen Sie Erstellen aus.
6. Kopieren Sie nach dem Start der Aufgabe die öffentliche IP der ausgeführten Aufgabe und navigieren Sie zu: `https://(your public IP)/explorer`.
7. Akzeptieren Sie das Risiko, das generierte unbekannte Zertifikat zu verwenden, oder fügen Sie es Ihrer Schlüsselkette hinzu.
8. Jetzt können Sie eine Verbindung mit Neptune hinzufügen. Erstellen Sie eine neue Verbindung (für ein Eigenschaftsdiagramm (LPG) oder für RDF) und legen Sie die folgenden Felder fest:

```
Using proxy server => true
Public or Proxy Endpoint => https://(your public IP address)
Graph connection URL => https://(your Neptune endpoint):8182
```

Die Verbindung sollte jetzt hergestellt sein.

Graph-explorer Vorführung

Dieses kurze Video zeigt, wie Sie Ihre Diagrammdaten mit Graph-Explorer einfach visualisieren können:



Tom Sawyer Software

Tom Sawyer Perspectives ist eine Low-Code-Entwicklungsplattform für die Visualisierung und Analyse von Diagrammen und Daten für in Amazon Neptune gespeicherte Daten. Mit den integrierten Design- und Vorschauschnittstellen und den umfangreichen API-Bibliotheken können Sie schnell benutzerdefinierte Visualisierungsanwendungen in Produktionsqualität erstellen. Mittels einer Point-and-Click-Designeroberfläche und 30 integrierten Analysealgorithmen können Sie Anwendungen entwerfen und entwickeln, um Einblicke in Daten aus Dutzenden von Quellen zu erhalten.

Der Tom Sawyer Graph Database Browser vereinfacht die Visualisierung und Analyse von Daten in Amazon Neptune. Sie können Verbindungen in Ihren Daten ohne umfassende Kenntnisse der Abfragesprache oder des Abfrageschemas anzeigen und verstehen. Sie können ohne technisches Wissen mit den Daten interagieren, indem Sie einfach die Nachbarn der ausgewählten Knoten laden und die Visualisierung in die gewünschte Richtung erstellen. Sie können auch fünf verschiedene Diagrammlayouts nutzen, um das Diagramm auf eine möglichst relevante Weise darzustellen. Sie können Zentralitäts-, Clustering- und Pfadfindungsanalysen anwenden, um bisher unerkannte Muster aufzudecken. Ein Beispiel für die Integration von Graph Database Browser mit Neptune finden Sie in diesem Blogbeitrag. Im AWS Marketplace finden Sie eine kostenlose Testversion von Graph Database Browser.

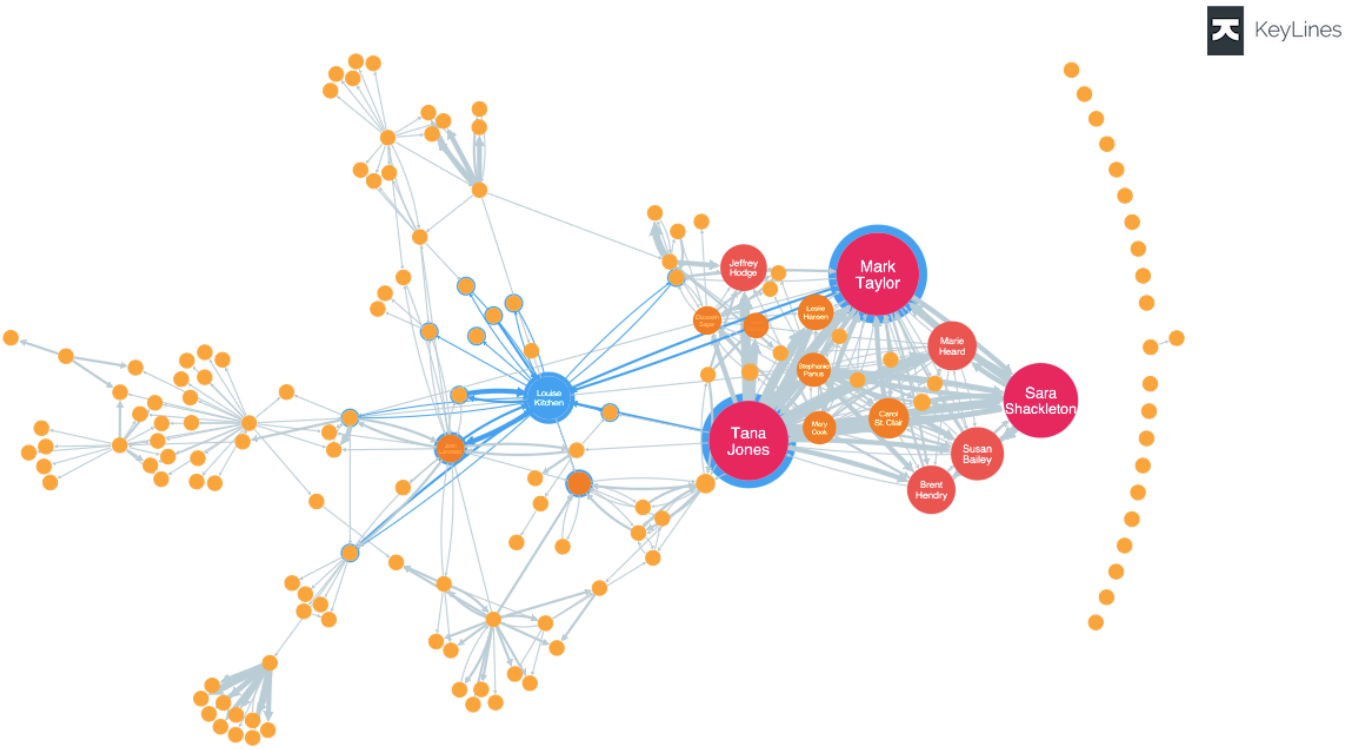


Cambridge Intelligence

Cambridge Intelligence stellt Datenvisualisierungstechnologien für die Untersuchung und Interpretation von Amazon-Neptune-Daten bereit. Die Toolkits zur Grafikvisualisierung (KeyLinesfür JavaScript Entwickler und ReGraphfür React-Entwickler) bieten eine einfache Möglichkeit, hochgradig interaktive und anpassbare Tools für Webanwendungen zu erstellen. Diese Toolkits nutzen WebGL und HTML5 Canvas für eine hohe Leistung, unterstützen erweiterte Diagrammanalysefunktionen und kombinieren Flexibilität und Skalierbarkeit mit einer sicheren, robusten Architektur. Diese SDKs funktionieren mit Neptune-Gremlin- und RDF-Daten.

Nutzen Sie diese Tutorials für die Integration von Gremlin-Daten, SPARQL-Daten und Neptune-Architektur.

Hier ist ein Beispiel für eine KeyLines Visualisierung:

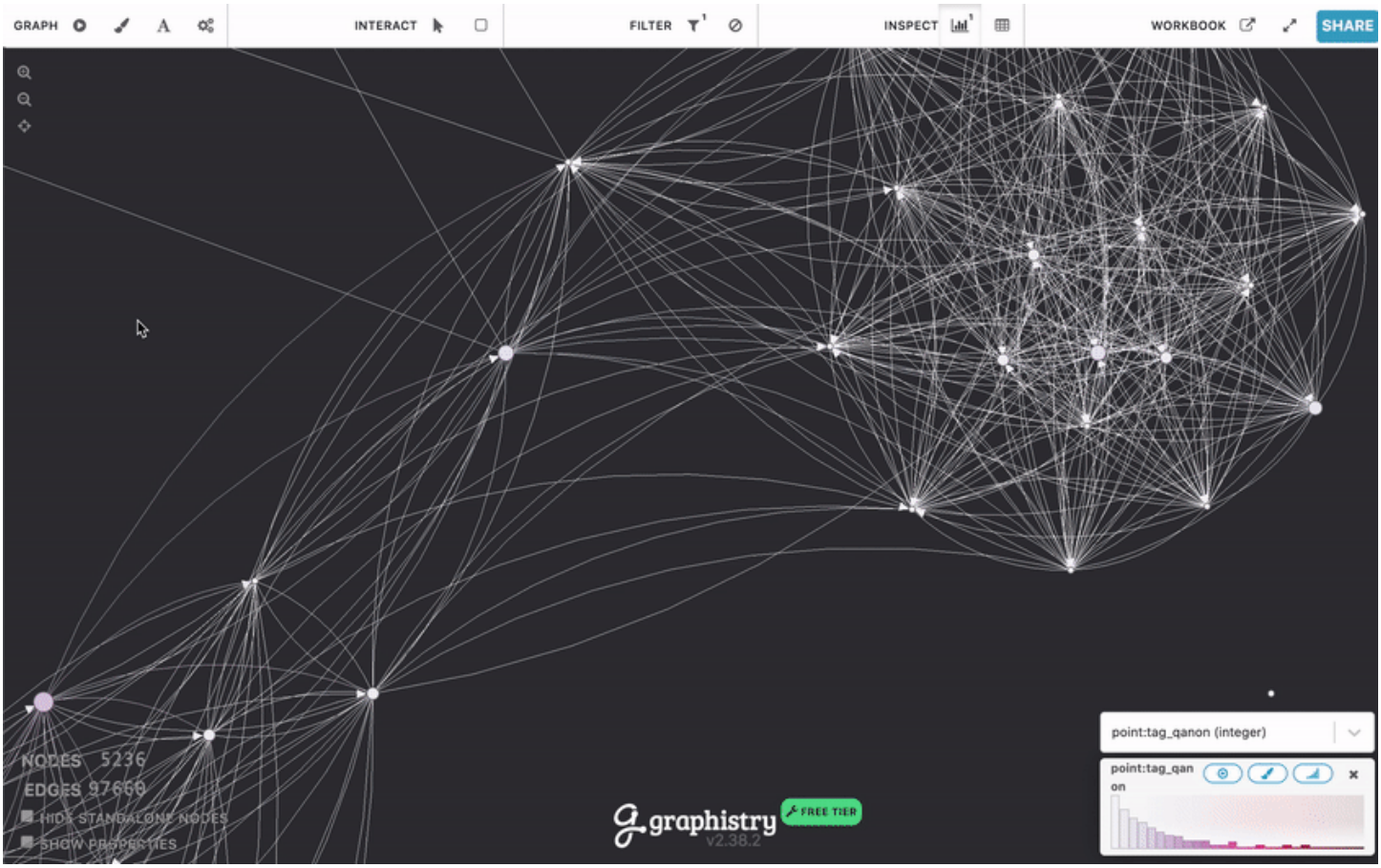


Graphistry

Graphistry ist eine Plattform für visuelle Diagrammintelligenz, die mittels GPU-Beschleunigung eine umfassende visuelle Erfahrung bietet. Teams können über verschiedene Features in Graphistry zusammenarbeiten, von der codefreien Untersuchung von Dateien und Datenbanken über das Teilen von Jupyter-Notebooks und Streamlit-Dashboards bis zur Nutzung der Einbettungs-API in Ihren eigenen Apps.

Sie können mit vollständig interaktiven Low-Code-Dashboards beginnen, indem Sie einfach das Graph-App-Kit konfigurieren und starten und nur wenige Codezeilen ändern. In diesem Blogbeitrag finden Sie eine Anleitung für die Erstellung Ihres ersten Dashboards mit Graphistry und Neptune. Sie können auch die PyGraphistryNeptune-Demo ausprobieren. PyGraphistry ist eine Python-Bibliothek für visuelle Graphanalysen für Notebooks. In diesem Tutorial-Notizbuch finden Sie eine Neptune-Demo PyGraphistry .

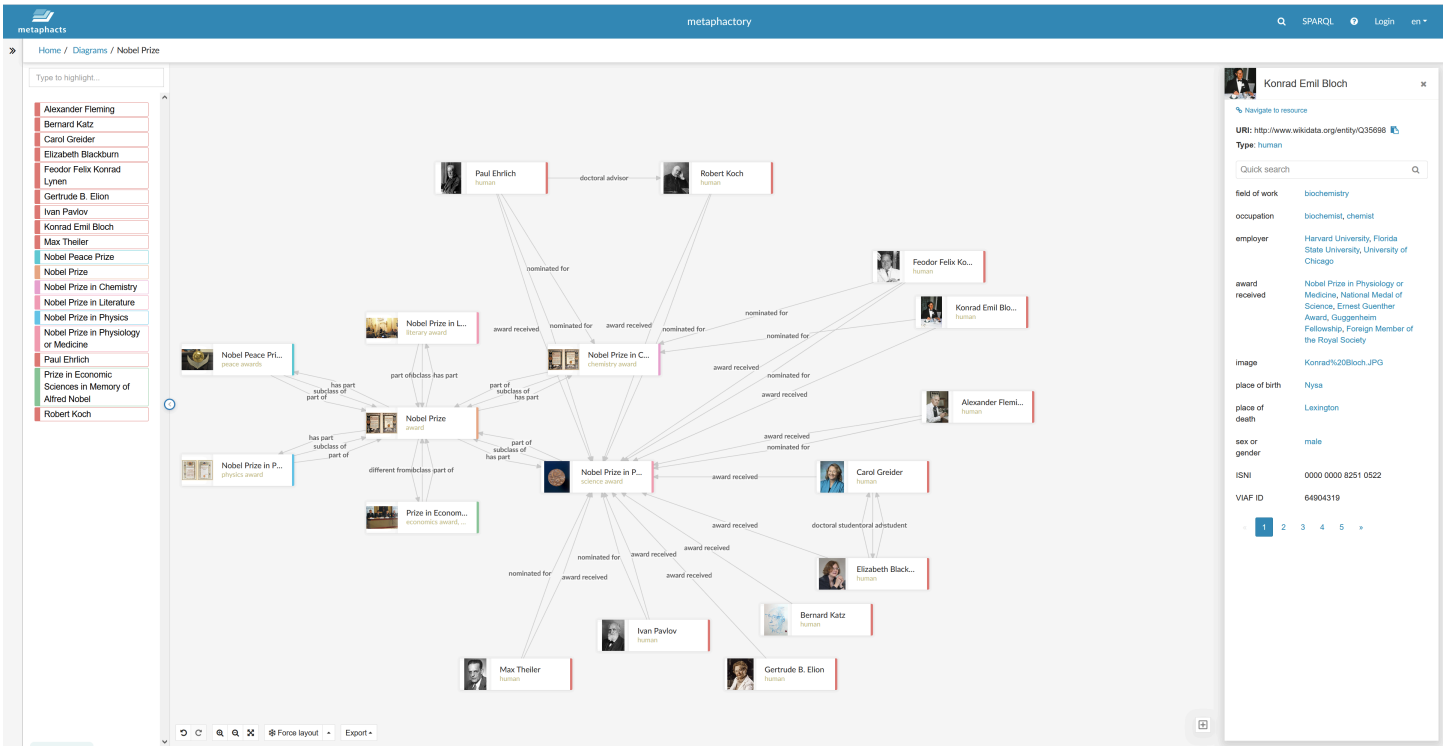
Besuchen Sie zunächst Graphistry in the AWS Marketplace.



metaphacts

metaphacts ist eine flexible, offene Plattform für die Beschreibung und Abfrage von Diagrammdaten und die Visualisierung von Wissensdiagrammen und die Interaktion mit diesen. Mit metaphactory können Sie interaktive Webanwendungen wie Visualisierungen und Dashboards mittels Wissensdiagrammen in Neptune anhand des RDF-Datenmodells erstellen. Die metaphactory-Plattform unterstützt eine Low-Code-Entwicklungserfahrung mit einer Benutzeroberfläche für das Laden von Daten, einer visuellen Ontologie-Modellierungsschnittstelle mit OWL- und SHACL-Unterstützung, einer SPARQL-Abfrageoberfläche und einem SPARQL-Abfragekatalog sowie verschiedenen Webkomponenten zum Untersuchen, Visualisieren, Durchsuchen und Erstellen von Diagrammen.

Dies ist ein Beispiel für eine metaphactory-Visualisierung:



Die Plattform wurde für Maschinenbau, Fertigung, Pharma, Biowissenschaften, Finanzen, Versicherungen und weitere Bereiche entwickelt und wird in diesen Bereichen produktiv eingesetzt. Ein Beispiel für eine Lösungsarchitektur finden Sie in diesem Blogbeitrag.

Im AWS Marketplace finden Sie eine kostenlose Testversion von metaphactory.

G.V() Graph-Datenbank-Client

G.V() ist ein All-in-One-Graphdatenbank-Client, der für Entwickler und Datenexperten entwickelt wurde, um Amazon Neptune Neptune-Grafikdaten zu untersuchen und mit ihnen zu interagieren.

Mit G.V () können Sie Ihre Grafikprojekte beschleunigen und alltägliche Graphdatenbankaufgaben effizienter ausführen. Es ist über AWS Marketplace oder als eigenständige Desktop-Anwendung für Windows, macOS und Linux erhältlich.

Schlüssel-Features

Abfrage-Editoren

Die integrierten Gremlin- und Cypher-Abfrage-Editoren bieten automatische Vervollständigung, Vorschläge und Dokumentation in Echtzeit, während Sie Ihre Abfrage eingeben. Diese Funktionen basieren auf Ihrem Graphdatenschema, sodass Sie schnell genaue Abfragen schreiben können.

Sie können Abfragen parametrisieren und als Berichte mit einem Klick speichern, sodass Sie in wenigen Minuten aussagekräftige, zentrale Berichte für Ihre Graphdatenbank erstellen können.

Formate zur Datenvisualisierung

Um Ihre Abfrageergebnisse zu analysieren, bietet G.V () mehrere Datenvisualisierungsformate:

- Graphische Visualisierung zur Untersuchung von Zusammenhängen und Mustern in Ihren Daten, mit optionaler Unterlage für geografische Kartendaten
- Tabellenansichten, die Scheitelpunkte, Kanten oder tabellarische Abfrageergebnisse als Zeilen für einen schnellen Vergleich von Eigenschaftswerten organisieren
- JSON-Datenformat für entwicklerfreundliche Ausgabe
- Objektbrowser zum Navigieren in Ihren Daten auf der Grundlage ihrer hierarchischen Struktur, der sich am besten für komplexe Informationsaggregationen eignet

Grafikdaten-Explorer

G.VMit dem Graph Data Explorer ohne Code können Sie Ihre Grafikdaten mithilfe von Eigenschaftsfiltern mühelos durchsuchen und Kanten von Scheitelpunkten interaktiv über die Diagrammansicht durchqueren. Sie können auch Scheitelpunkte, Kanten und deren Eigenschaften mit wenigen Klicks erstellen, aktualisieren und löschen, um Ihre Grafikdaten zu ändern, ohne vollständige Abfragen schreiben zu müssen.

Ansicht des Grafikdatenmodells

Die Ansicht „Datenmodell grafisch“ bietet eine detaillierte visuelle Darstellung Ihres Diagramms für beschriftete Immobilien, damit Ihr Team Ihr Datenmodell besser verstehen kann.

Amazon Neptune Neptune-Integration

G.V() bietet eine umfassende Integration mit Amazon Neptune, einschließlich:

- IAM-Authentifizierung
- Neptune-API-Unterstützung
- Langsame Query-Analyse
- Einblicke in das Auditprotokoll

Weitere Informationen finden Sie in der Dokumentation G.V ().

Verfügbarkeit und Updates

G.V() wird ständig weiterentwickelt und es werden monatlich neue Funktionen veröffentlicht. Starten Sie noch heute mit einer kostenlosen Testversion der Desktop - oder AWS Marketplace-Version.

Vorführung des Produkts

Sehen Sie G.V () in Aktion:

Weitere Informationen

Weitere Informationen zu den Funktionen und Fähigkeiten von G.V () finden Sie in der G.V() - Dokumentation.

Linkurious

Linkurious bietet technischen und nicht-technischen Anwendern eine unternehmenstaugliche kontextuelle Entscheidungsinformationsplattform der nächsten Generation, die auf nativer Graphtechnologie basiert.

Linkurious wird von Behörden, Global-2000-Unternehmen und gemeinnützigen Organisationen eingesetzt und hilft datengestützten Forschern, Analysten und Ermittlern, versteckte Muster in komplexen vernetzten Daten zu erkennen — und schnellere, kontextbasierte Entscheidungen zu treffen.

Für Neptune-Benutzer bietet Linkurious mehrere fortschrittliche Softwarelösungen für Graphenanalyse und Visualisierung, mit denen Unternehmen verbundene Daten, die in der Neptune-Datenbank gespeichert sind, visualisieren, untersuchen und analysieren können.

Linkurious Enterprise

Linkurious Enterprise ist ein eigenständiges Modul der Linkurious Decision Intelligence Platform. Es handelt sich um eine handelsübliche Grafikvisualisierungs- und Analysesoftware, mit der Benutzer in Neptune gespeicherte Daten visuell und intuitiv nutzen können. Mit seiner benutzerfreundlichen Oberfläche, fortschrittlichen Analysefunktionen und Sicherheitsfunktionen für Unternehmen ermöglicht Linkurious Enterprise Teams, komplexe Muster zu erkennen, versteckte Zusammenhänge aufzudecken und Entscheidungen auf der Grundlage von Neptun-Graphdaten zu beschleunigen.

- Visualisieren Sie Neptundaten: Stellen Sie eine nahtlose Verbindung zu Neptune her und greifen Sie auf eine 360°-Ansicht Ihrer Grafikdaten für eine dynamische, interaktive Erkundung zu.
- Beschleunigen Sie die Datenanalyse: Filtern Sie Ergebnisse, untersuchen Sie Zeitleisten, wechseln Sie in den Geomodus oder verwenden Sie den Abfragegenerator ohne Code und voreinstellen Sie spezifische Abfragen für alle Benutzer, um schnell aussagekräftige Erkenntnisse zu gewinnen.
- Automatische Erkennung komplexer Muster: Verwenden Sie grafikbasierte, modellübergreifende Warnmeldungen und die Deduplizierung von Warnmeldungen, um automatisch interessante Muster im Diagramm aufzudecken und den Aufwand für die Triage zu reduzieren.
- Effiziente Zusammenarbeit: Teilen Sie Diagrammansichten und Untersuchungsergebnisse teamübergreifend oder auf Organisationsebene, um eine konsistente, datengestützte Entscheidungsfindung zu unterstützen.
- Sichere Bereitstellung im Unternehmen: Stellen Sie die Einhaltung der Vorschriften mit rollenbasierter Zugriffskontrolle, LDAP/SSO Integration und Auditprotokollierung sicher.

Technische Informationen finden Sie in der Dokumentation von Linkurious Enterprise.

Interessiert daran, Linkurious Enterprise zu testen? Fordern Sie eine kostenlose 30-Tage-Testversion an.

Preise und Abonnementoptionen finden Sie auf AWS Marketplace.

Ogma

Ogma ist eine von Linkurious entwickelte kommerzielle JavaScript Bibliothek, die für Entwickler konzipiert wurde, die leistungsstarke, groß angelegte interaktive Grafikvisualisierungen in Anwendungen erstellen, die Neptun-Daten nutzen.

Zu den wichtigsten Features gehören:

- High-performance Rendern: Verwenden Sie WebGL und proprietäre Algorithmen, um Zehntausende von Knoten fast sofort zu visualisieren und dabei Präzision und Qualität beizubehalten, selbst mit einer 5 Jahre alten GPU, um ein reibungsloses Benutzererlebnis zu gewährleisten.
- Anpassung: Maßgeschneiderte Visualisierungen mit einer Vielzahl von Layouts, Stilen und interaktiven Funktionen. Bereichern Sie Visualisierungen mit eingebetteten Graphalgorithmen oder externen Klassifikatoren für maschinelles Lernen. Fügen Sie visuelle Überlagerungen und Bildanmerkungen hinzu, um menschliche Einblicke und Zusammenarbeit zu ermöglichen.

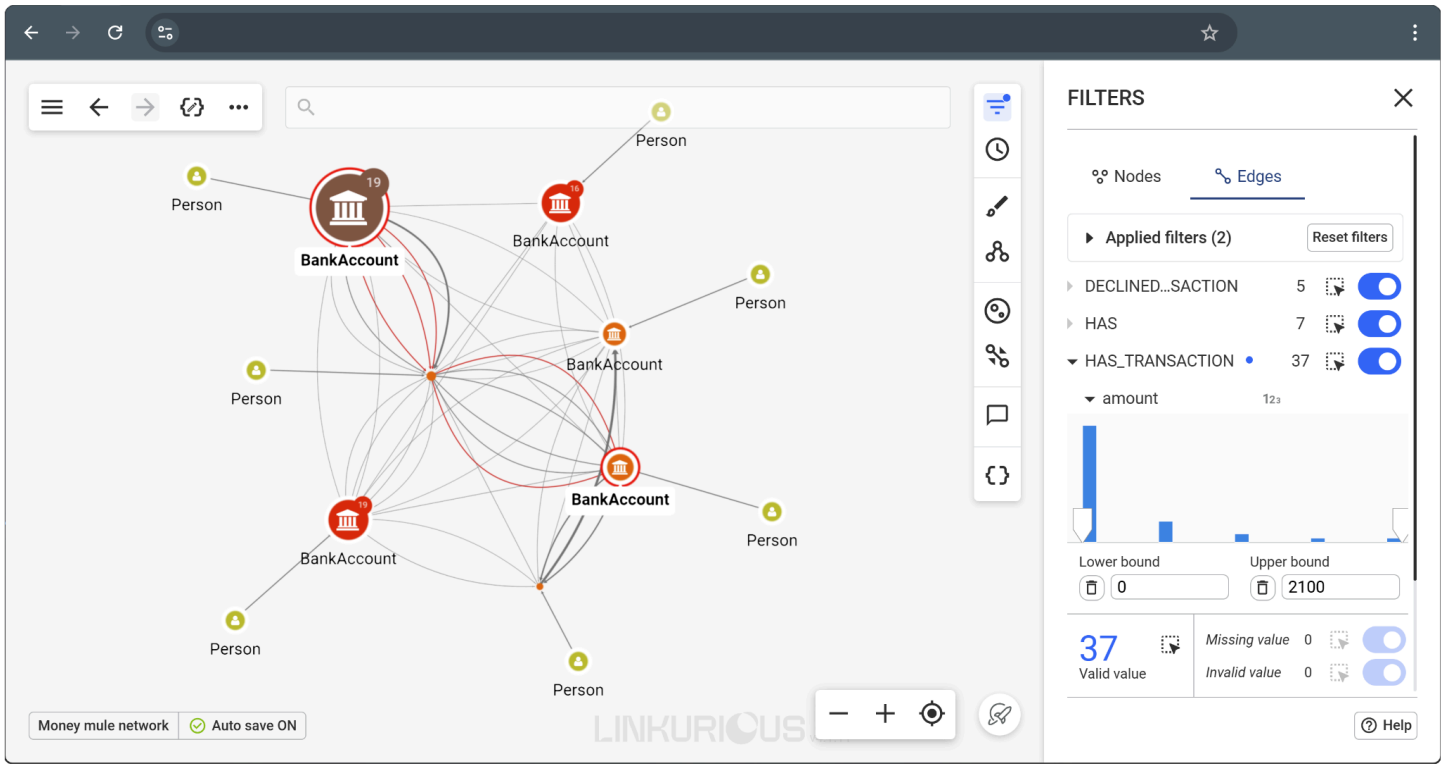
- Integration: Beginnen Sie schnell mit umfangreicher Dokumentation, Tutorials und einem Code-Playground. Laden und transformieren Sie Daten in wenigen Minuten, integrieren Sie Ogma in Ihr Framework und greifen Sie auf erweiterten Entwicklersupport zu.

Ausführlichere Informationen und Tutorials zur Integration von Ogma mit Neptune finden Sie in der Ogma-Dokumentation.

Interessiert an einer Probefahrt mit Ogma? Fordern Sie eine kostenlose 30-Tage-Testversion an.

Preise und Abonnementoptionen finden Sie auf AWS Marketplace.

Hier ist eine Visualisierung von Linkurious Enterprise:



Graph.Build

In jedem Bereich ist die Zusammenarbeit mit Fachexperten der Schlüssel zur Entwicklung von Graphmodellen, die spezifische Anwendungsfälle effektiv adressieren. Ganz gleich, ob Sie traditionelle Graph-Workloads wie Graphenanalysen in Angriff nehmen oder sich mit fortschrittlichen KI-Anwendungen wie GraphRag, Linkvorhersage oder neurosymbolischem Denken befassen, die Fähigkeit, Graphmodelle zu erstellen und zu iterieren, ist unerlässlich. Damit Graph.Build wird das Entwerfen und Verfeinern von Graphmodellen leichter zugänglich, sodass Fachexperten

Hand in Hand mit Ingenieuren und Datenanalysten zusammenarbeiten können, um wirkungsvolle, maßgeschneiderte Lösungen zu entwickeln.

Bei der Graph.BuildPlattform handelt es sich um ein eigenes Studio und eine Build-Factory für Graphmodelle ohne Programmierkenntnisse.

Graph.Build ist auf dem AWS Marketplace verfügbar.

Graph.Build Studio:

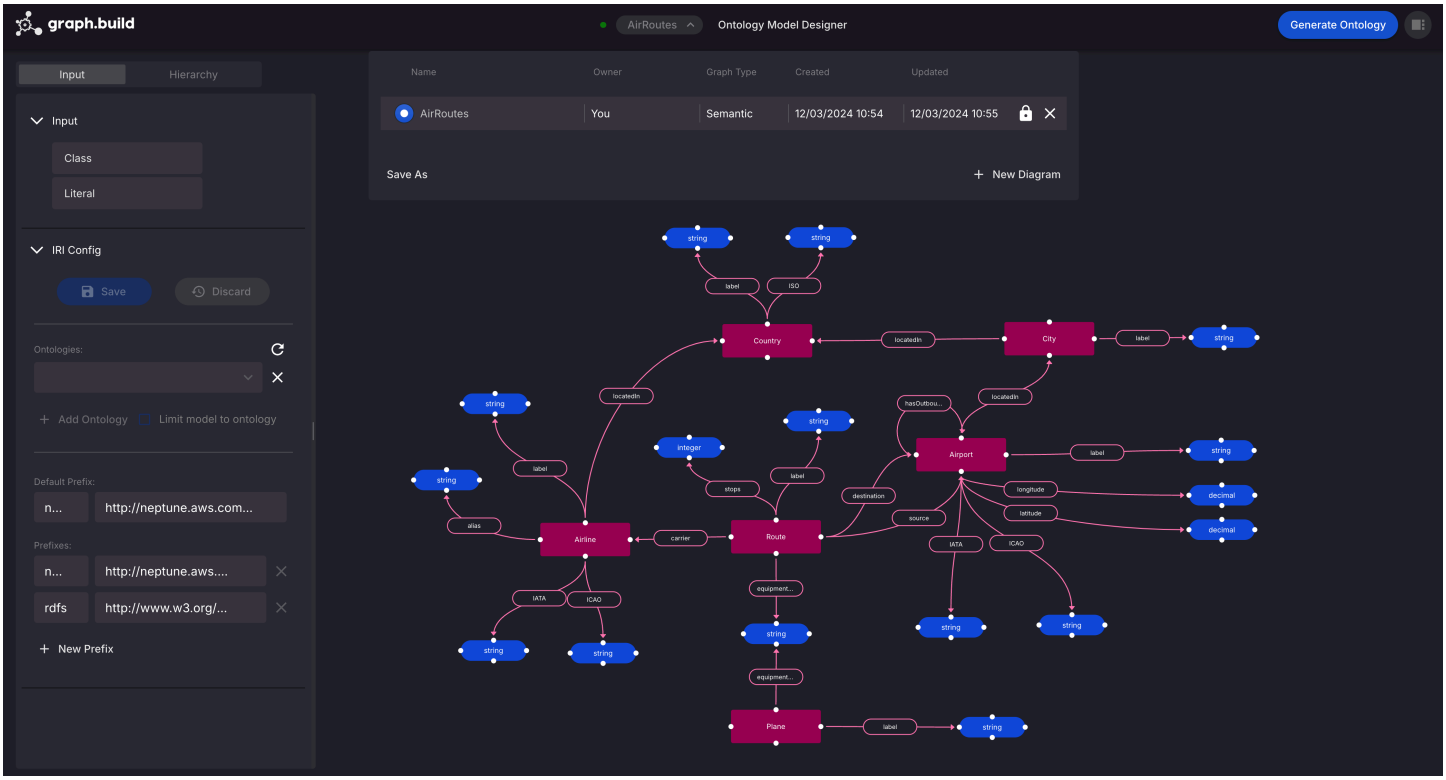
Studio ist eine kollaborative Graphdesign-IDE ohne Code. Sie können das Studio verwenden, um Schemas für LPG-Graphen, Ontologien für RDF-Diagramme, ETL-Modelle, Change Data Capture-Konfigurationen und sogar virtuelle SPARQL-Graphendpunkte zu entwerfen.

Graph.Build Transformatoren:

Transformatoren erstellen Graphmodelle aus realen Daten. Transformer-Graphmodelle können im Studio entworfen werden, was bedeutet, dass für das Entwerfen, Erstellen und Testen von Graphmodellen kein Code erforderlich ist. Sie können Transformatoren mit einer Vielzahl von Datenquellen verbinden, darunter Amazon Aurora, Amazon Athena, Amazon S3 und Amazon MSK. Transformatoren verfügen über eine Vielzahl von Funktionen zur Verwaltung von Graphmodellen, die in anderen Datensätzen zu finden sind, darunter Change Data Capture (CDC), Virtualisierung und Materialisierung im Aktualisierungs- oder Einfügemodus.

Graph.Build Autoren:

Writer sind so konfiguriert, dass sie sowohl Labelled Property Graph (LPG) als auch Resource Description Framework (RDF) -Grafikmodelle in Zielgraphdatenbanken wie Amazon Neptune und Neptune Analytics schreiben oder aktualisieren.



[Fordern Sie noch heute eine Demo an](#).

Exportieren von Daten aus einem Neptune-DB-Cluster

Es gibt mehrere gute Möglichkeiten, Daten aus einem Neptune-DB-Cluster zu exportieren:

- Verwenden Sie für kleine Datenmengen einfach die Ergebnisse einer oder mehrerer Abfragen.
- Für RDF-Daten kann das Graph Store Protocol (GSP) den Export vereinfachen. Beispiel:

```
curl --request GET \
  'https://your-neptune-endpoint:port/sparql/gsp/?graph=http://www.example.com/named/graph'
```

- Es gibt auch ein leistungsstarkes und flexibles Open-Source-Tool für den Export von Neptune-Daten, nämlich `neptune-export`. In den folgenden Abschnitten wird beschrieben, wie Sie dieses Tool nutzen können.

Themen

- Verwenden von neptune-export
- Nutzung des Neptune-Export Dienstes zum Exportieren von Neptun-Daten
- Exportieren von Daten aus Neptune mithilfe des neptune-export-Befehlszeilen-Tools
- Von Neptune-Export exportierte Dateien und neptune-export
- Parameter, die zur Steuerung des Neptun-Exportprozesses verwendet werden
- Problembehandlung beim Neptune-Exportprozess
- Exportieren von Gremlin-Abfrageergebnissen nach Amazon S3

Verwenden von **`neptune-export`**

Sie können mit dem Open-Source-Tool `neptune-export` auf zwei verschiedene Arten arbeiten:

- Als Neptune-Export-Service. Wenn Sie mit dem Neptune-Export-Service Daten aus Neptune exportieren, lösen Sie Exportaufträge über eine REST-API aus und überwachen sie.
- Als **`neptune-export`**-Java-Befehlszeilendienstprogramm. Um dieses Befehlszeilen-Tool zum Exportieren von Neptune-Daten zu verwenden, müssen Sie es in einer Umgebung ausführen, in der Ihr Neptune-DB-Cluster zugänglich ist.

Sowohl der Neptune-Export Service als auch das `neptune-export`-Befehlszeilen-Tool veröffentlichen Daten in Amazon Simple Storage Service (Amazon S3), verschlüsselt mit serverseitiger Amazon-S3-Verschlüsselung (`SSE-S3`).

> **Note**
>
> Es hat sich bewährt, die Zugriffsprotokollierung für alle Amazon-S3-Buckets zu aktivieren, damit Sie den gesamten Zugriff auf diese Buckets überprüfen können.

Wenn Sie versuchen, Daten aus einem Neptune-DB-Cluster zu exportieren, dessen Daten sich während des Exports ändern, kann die Konsistenz der exportierten Daten nicht garantiert werden. Dies bedeutet: Wenn Ihr Cluster Schreibdatenverkehr verarbeitet, während ein Exportauftrag ausgeführt wird, kann es zu Inkonsistenzen in den exportierten Daten kommen. Dies gilt unabhängig davon, ob Sie von der primären Instance im Cluster oder von einem oder mehreren Lesereplikaten exportieren.

Um sicherzustellen, dass die exportierten Daten konsistent sind, empfiehlt es sich, aus einem Klon Ihres DB-Clusters zu exportieren. Dadurch erhält das Export-Tool einerseits eine statische Version Ihrer Daten und andererseits wird sichergestellt, dass der Exportauftrag die Abfragen in Ihrem ursprünglichen DB-Cluster nicht verlangsamt.

Um dies zu vereinfachen, können Sie angeben, dass Sie den Quell-DB-Cluster klonen möchten, wenn Sie einen Exportauftrag auslösen. Wenn Sie dies tun, erstellt der Exportvorgang automatisch den Klon, verwendet ihn für den Export und löscht ihn dann, wenn der Export abgeschlossen ist.

Nutzung des Neptune-Export Dienstes zum Exportieren von Neptun-Daten

Mit den folgenden Schritten können Sie mithilfe des Service Daten aus Ihrem Neptune-DB-Cluster nach Amazon S3 exportieren: Neptune-Export

Den Service installieren Neptune-Export

Verwenden Sie eine AWS CloudFormation Vorlage, um den Stack zu erstellen:

Um den Neptune-Export Dienst zu installieren

1. Starten Sie den CloudFormation Stack auf der CloudFormation Konsole, indem Sie in der folgenden Tabelle eine der Schaltflächen Stack starten wählen:

Region	Anzeigen	In Designer anzeigen	Starten
USA Ost (Nord-Virginia)	Anzeigen	In Designer anzeigen	
USA Ost (Ohio)	Anzeigen	In Designer anzeigen	
USA West (Nordkalifornien)	Anzeigen	In Designer anzeigen	
USA West (Oregon)	Anzeigen	In Designer anzeigen	
Kanada (Zentral)	Anzeigen	In Designer anzeigen	
Südamerika (São Paulo)	Anzeigen	In Designer anzeigen	
Europa (Stockholm)	Anzeigen	In Designer anzeigen	
Europa (Irland)	Anzeigen	In Designer anzeigen	
Europa (London)	Anzeigen	In Designer anzeigen	
Europa (Paris)	Anzeigen	In Designer anzeigen	
Europa (Frankfurt)	Anzeigen	In Designer anzeigen	
Middle East (Bahrain)	Anzeigen	In Designer anzeigen	

Region	Anzeigen	In Designer anzeigen	Starten
Naher Osten (VAE)	Anzeigen	In Designer anzeigen	Launch Stack
Israel (Tel Aviv)	Anzeigen	In Designer anzeigen	Launch Stack
Afrika (Kapstadt)	Anzeigen	In Designer anzeigen	Launch Stack
Asien-Pazifik (Hongkong)	Anzeigen	In Designer anzeigen	Launch Stack
Asien-Pazifik (Tokio)	Anzeigen	In Designer anzeigen	Launch Stack
Asien-Pazifik (Seoul)	Anzeigen	In Designer anzeigen	Launch Stack
Asien-Pazifik (Singapur)	Anzeigen	In Designer anzeigen	Launch Stack
Asien-Pazifik (Sydney)	Anzeigen	In Designer anzeigen	Launch Stack
Asien-Pazifik (Mumbai)	Anzeigen	In Designer anzeigen	Launch Stack
China (Peking)	Anzeigen	In Designer anzeigen	Launch Stack
China (Ningxia)	Anzeigen	In Designer anzeigen	Launch Stack
AWS GovCloud (US-West)	Anzeigen	In Designer anzeigen	Launch Stack
AWS GovCloud (US-East)	Anzeigen	In Designer anzeigen	Launch Stack

2. Wählen Sie auf der Seite Vorlage auswählen die Option Weiter aus.
3. Stellen Sie auf der Seite Details angeben die folgenden Parameter für die Vorlage ein:

- **`VPC`**— Der einfachste Weg, den Neptune-Export Service einzurichten, besteht darin, ihn in derselben Amazon VPC wie Ihre Neptune-Datenbank zu installieren. Wenn Sie es in einer separaten VPC installieren möchten, können Sie VPC-Peering verwenden, um die Konnektivität zwischen der VPC des Neptune-DB-Clusters und der Service-VPC herzustellen. Neptune-Export
- **`Subnet1`**— Der Neptune-Export Dienst muss in einem Subnetz in Ihrer VPC installiert sein, das ausgehenden IPv4-HTTPS-Verkehr vom Subnetz zum Internet ermöglicht. Auf diese Weise kann der Neptune-Export Dienst die AWS Batch-API aufrufen, um einen Exportauftrag zu erstellen und auszuführen.

  Wenn Sie Ihren Neptun-Cluster mithilfe der CloudFormation Vorlage auf der Neptun-Cluster erstellen Seite in der Neptune-Dokumentation erstellt haben, können Sie die `PrivateSubnet1` und `PrivateSubnet2` -Ausgaben dieses Stacks verwenden, um diesen und den nächsten Parameter zu füllen.
- **`Subnet2`** – Ein zweites Subnetz in der VPC, das ausgehenden IPv4-HTTPS-Verkehr vom Subnetz ins Internet ermöglicht.
- **`EnableIAM`**— Stellen Sie dies auf ein, `true` um die Neptune-Endpoint API mithilfe von (IAM) zu sichern. AWS Identity and Access Management Wir empfehlen Ihnen, dies zu tun.

  Wenn Sie die IAM-Authentifizierung aktivieren, muss `Sigv4` alle HTTPS-Anfragen an den Endpunkt signieren. Sie können ein Tool wie awscurl verwenden, um Anfragen in Ihrem Namen zu signieren.
- **`VPCOnly`**— Wenn Sie dies auf einstellen, `true` wird der Export-Endpunkt VPC-only, sodass Sie nur von der VPC aus darauf zugreifen können, in der der Neptune-Export Service installiert ist. Dadurch wird die Neptune-Export API darauf beschränkt, nur innerhalb dieser VPC verwendet zu werden.

  Wir empfehlen Ihnen, `VPCOnly` auf `true` einzustellen.
- **`NumOfFilesULimit`** – Geben Sie in der `ulimits`-Container-Eigenschaft einen `nofile`-Wert zwischen 10 000 und 1 000 000 an. Die Standardeinstellung ist 10 000. Wir empfehlen, die Standardeinstellung beizubehalten, es sei denn, Ihr Graph enthält eine große Anzahl von eindeutigen Etiketten.
- **`PrivateDnsEnabled`** (Boolesch) – Gibt an, ob der angegebenen VPC eine privat gehostete Zone zugeordnet werden soll. Der Standardwert ist `true`.

Wenn ein VPC-Endpunkt mit diesem aktiviertem Flag erstellt wird, wird der gesamte API-Gateway-Verkehr über den VPC-Endpunkt geleitet und öffentliche API-Gateway-Endpunktaufrufe werden deaktiviert. Wenn Sie `PrivateDnsEnabled` auf `false` einstellen, ist der öffentliche API-Gateway-Endpunkt aktiviert, aber der Neptune-Export-Service kann nicht über den privaten DNS-Endpunkt verbunden werden. Sie können dann einen öffentlichen DNS-Endpunkt für den VPC-Endpunkt verwenden, um den Export-Service aufzurufen, wie hier beschrieben.

- **`NeptuneExportVersion`**— Geben Sie die Version des Neptune Export Utility an, die verwendet werden soll. Alle Versionen, die größer oder gleich `v1.1.11` sind, werden unterstützt. Eine Version von `v2.latest` kann verwendet werden, um kleinere Updates automatisch zu erhalten. Die vollständige Liste der verfügbaren Versionen sowie Patchnotizen finden Sie in den GitHub Open-Source-Versionen.

4. Wählen Sie Weiter aus.
5. Wählen Sie auf der Seite Optionen Weiter aus.
6. Aktivieren Sie auf der Seite Überprüfen das erste Kontrollkästchen, um zu bestätigen, dass CloudFormation IAM-Ressourcen erstellt. Aktivieren Sie das zweite Kontrollkästchen, um `CAPABILITY_AUTO_EXPAND` für den neuen Stack zu bestätigen.

> **Note**
>
> `CAPABILITY_AUTO_EXPAND` bestätigt explizit, dass Makros ohne vorherige Überprüfung beim Erstellen des Stacks erweitert werden. Benutzer erstellen häufig einen Änderungssatz aus einer verarbeiteten Vorlage, sodass die von Makros durchgeführten Änderungen überprüft werden können, ehe der Stack tatsächlich erstellt wird. Weitere Informationen finden Sie in der CloudFormation CreateStack-API.

Wählen Sie die Option Erstellen aus.

Ermöglichen Sie den Zugriff auf Neptune von Neptune-Export

Aktualisieren Sie nach Abschluss der Neptune-Export Installation Ihre Neptune VPC-Sicherheitsgruppe, um den Zugriff von zu ermöglichen. Neptune-Export Wenn der Neptune-Export CloudFormation Stack erstellt wurde, enthält die Registerkarte Ausgaben eine ID.

`NeptuneExportSecurityGroup` Aktualisieren Sie Ihre Neptune VPC-Sicherheitsgruppe, um den Zugriff von dieser Neptune-Export Sicherheitsgruppe aus zu ermöglichen.

Aktivieren Sie den Zugriff auf den Neptune-Export Endpunkt von einer EC2-Instance aus VPC-based

Wenn Sie Ihren Neptune-Export Endpunkt erstellen VPC-only, können Sie nur von der VPC aus darauf zugreifen, in der der Neptune-Export Service installiert ist. Um Konnektivität von einer Amazon EC2 EC2-Instance in der VPC aus zu ermöglichen, von der aus Sie Neptune-Export API-Aufrufe tätigen können, hängen Sie die vom CloudFormation Stack `NeptuneExportSecurityGroup` erstellte Instance an diese Amazon EC2 EC2-Instance an.

Führen Sie einen Neptune-Export Job mithilfe der API aus Neptune-Export

Die Registerkarte Ausgaben des CloudFormation Stacks enthält auch die`NeptuneExportApiUri`. Verwenden Sie diesen URI immer dann, wenn Sie eine Anfrage an den Neptune-Export Endpunkt senden.

Ausführen eines Exportauftrags

- Stellen Sie sicher, dass dem Benutzer oder der Rolle, unter dem der Export ausgeführt wird, die entsprechende `execute-api:Invoke`-Berechtigung erteilt wurde.
- Wenn Sie den `EnableIAM` Parameter bei der Installation `true` im CloudFormation Stack auf gesetzt haben Neptune-Export, müssen Sie alle Anfragen an die Neptune-Export API `Sigv4` signieren. Wir empfehlen, awscurl zu verwenden, um Anfragen an die API zu richten. In allen Beispielen wird davon ausgegangen, dass die IAM-Authentifizierung aktiviert ist.
- Wenn Sie den `VPCOnly` Parameter bei der Installation `true` im CloudFormation Stack auf gesetzt haben Neptune-Export, müssen Sie die Neptune-Export API innerhalb der VPC aufrufen, normalerweise von einer Amazon EC2 EC2-Instance aus, die sich in der VPC befindet.

Um mit dem Exportieren von Daten zu beginnen, senden Sie eine Anfrage mit den Anforderungsparametern `command` und `outputS3Path` und einem `endpoint`-Exportparameter an den `NeptuneExportApiUri`-Endpunkt.

Das Folgende ist ein Beispiel für eine Anfrage, die Eigenschaftsgraphdaten aus Neptune exportiert und in Amazon S3 veröffentlicht:

```
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)" }
      }'
```

Ähnlich ist hier ein Beispiel für eine Anfrage, die RDF-Daten von Neptune zu Amazon S3 exportiert:

```
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)" }
      }'
```

Wenn Sie den `command` Anforderungsparameter weglassen, wird standardmäßig Neptune-Export versucht, Eigenschaftsdiagrammdaten aus Neptune zu exportieren.

Wenn der vorherige Befehl erfolgreich ausgeführt wurde, würde die Ausgabe wie folgt aussehen:

```
{
  "jobName": "neptune-export-abc12345-1589808577790",
  "jobId": "c86258f7-a9c9-4f8c-8f4c-bbfe76d51c8f"
}
```

Überwachen Sie des soeben gestarteten Exportauftrags

Um einen laufenden Auftrag zu überwachen, hängen Sie dessen JobID an Ihren `NeptuneExportApiUri` an, etwa so:

```
curl \
  (your NeptuneExportApiUri)/(the job ID)
```

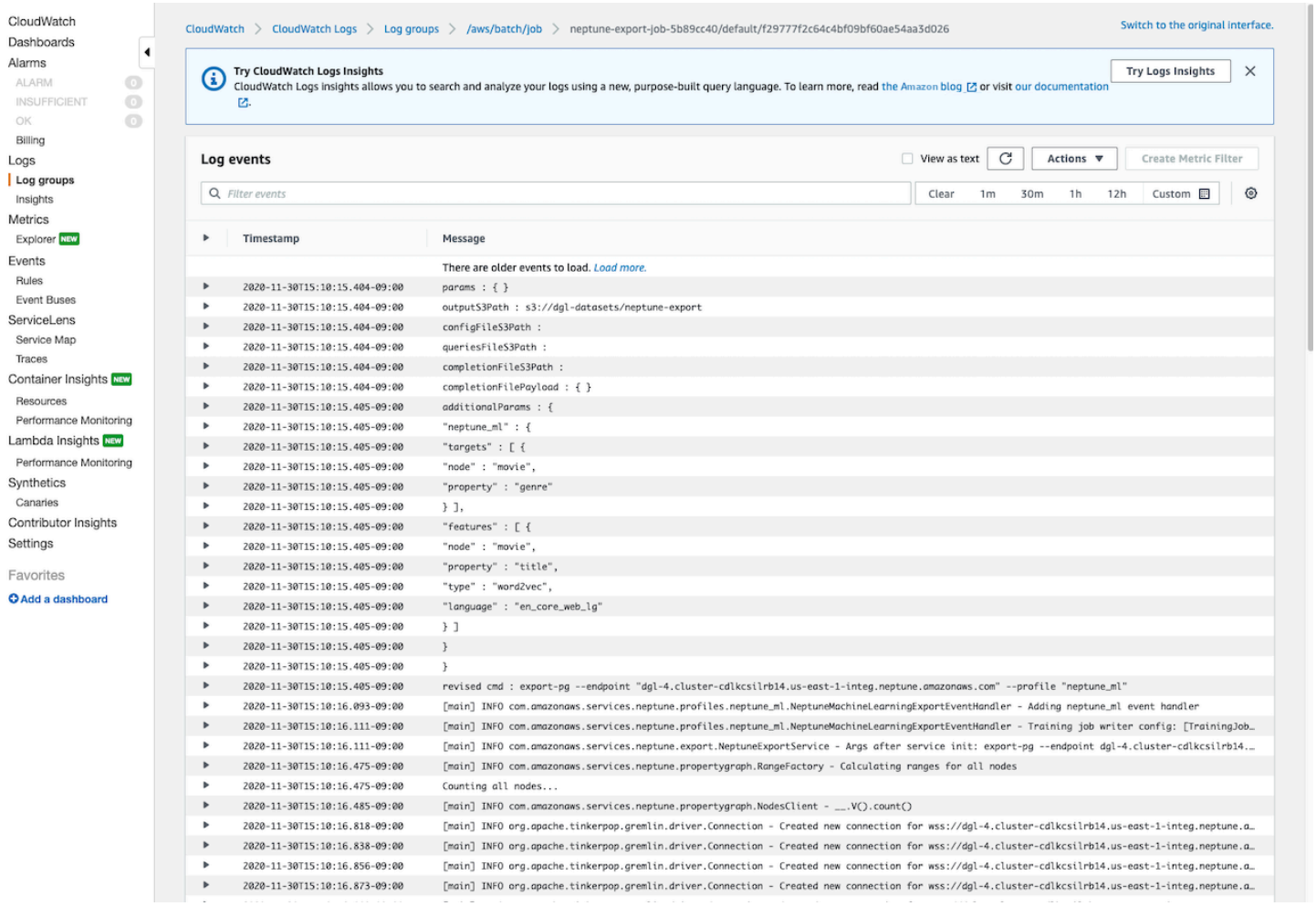
Wenn der Service den Exportauftrag noch nicht gestartet hat, sieht die Antwort wie folgt aus:

```
{
  "jobId": "c86258f7-a9c9-4f8c-8f4c-bbfe76d51c8f",
  "status": "pending"
}
```

Wenn Sie den Befehl wiederholen, nachdem der Exportauftrag gestartet wurde, sieht die Antwort etwa so aus:

```
{
  "jobId": "c86258f7-a9c9-4f8c-8f4c-bbfe76d51c8f",
  "status": "running",
  "logs": "https://us-east-1.console.aws.amazon.com/cloudwatch/home?..."
}
```

Wenn Sie die CloudWatch Protokolle in Logs mit der URI öffnen, die im Statusaufruf bereitgestellt wurde, können Sie den Fortschritt des Exports im Detail überwachen:



Stornieren eines laufenden Exportauftrags

Um einen laufenden Exportauftrag abzubrechen, verwenden Sie den AWS-Managementkonsole

1. Öffnen Sie die AWS Batch Konsole unter https://console.aws.amazon.com/batch/.
2. Wählen Sie Jobs (Aufträge) aus.
3. Suchen Sie nach dem laufenden Auftrag, den Sie stornieren möchten, anhand seiner `jobID`.
4. Wählen Sie Auftrag stornieren aus.

So stornieren Sie einen laufenden Exportauftrag mit der Neptune-Export-API:

Senden Sie eine `HTTP DELETE`-Anfrage mit der angehängten `jobID` an den `NeptuneExportApiUri`, etwa so:

```
curl -X DELETE \
  (your NeptuneExportApiUri)/(the job ID)
```

Exportieren von Daten aus Neptune mithilfe des **`neptune-export`**-Befehlszeilen-Tools

Anhand der folgenden Schritte können Sie mit dem `neptune-export`-Befehlszeilen-Dienstprogramm Daten aus Ihrem Neptune-DB-Cluster zu Amazon S3 exportieren:

Voraussetzungen für die Verwendung des **`neptune-export`**-Befehlszeilen-Dienstprogramms

Bevor Sie beginnen

- Version 8 des JDK – Sie müssen Version 8 des Java SE Development Kit (JDK) installiert haben.
- Download des Neptune-Export-Hilfsprogramm – Laden und installieren Sie die Datei neptune-export.jar.
- Sicherstellen, dass **`neptune-export`** Zugriff auf Ihre Neptune-VPC hat – Führen Sie neptune-export von einem Ort aus, an dem es auf die VPC zugreifen kann, in der sich Ihr Neptune-DB-Cluster befindet.

Sie können es beispielsweise auf einer Amazon-EC2-Instance innerhalb der Neptune-VPC oder in einer separaten VPC, die mit der Neptune-VPC gepaart ist, oder auf einem separaten Bastion-Host ausführen.

- Sicherstellen, dass die VPC-Sicherheitsgruppen Zugriff auf **`neptune-export`** gewähren – Vergewissern Sie sich, dass die mit der Neptune-VPC verbundenen VPC-Sicherheitsgruppen den Zugriff auf Ihren DB-Cluster von der IP-Adresse oder Sicherheitsgruppe aus ermöglichen, die der `neptune-export`-Umgebung zugeordnet ist.
- Richten Sie die erforderlichen IAM-Berechtigungen ein — Wenn in Ihrer Datenbank die AWS Identity and Access Management (IAM-) Datenbankauthentifizierung aktiviert ist, stellen Sie sicher, dass die Rolle, unter der `neptune-export` ausgeführt wird, mit einer IAM-Richtlinie verknüpft ist, die Verbindungen zu Neptune zulässt. Informationen zu Neptune-Richtlinien finden Sie unter Verwenden von IAM-Richtlinien.

  Wenn Sie den `clusterId`-Exportparameter in Ihren Abfrageanfragen verwenden möchten, erfordert die Rolle, unter der `neptune-export` ausgeführt wird, die folgenden IAM-Berechtigungen:

  - `rds:DescribeDBClusters`
  - `rds:DescribeDBInstances`
  - `rds:ListTagsForResource`

  Wenn Sie aus einem geklonten Cluster exportieren möchten, erfordert die Rolle, unter der `neptune-export` ausgeführt wird, die folgenden IAM-Berechtigungen:

  - `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`

Um die exportierten Daten in Amazon S3 zu veröffentlichen, erfordert die Rolle, unter der `neptune-export` ausgeführt wird, die folgenden IAM-Berechtigungen für den/die Amazon-S3-Standort(e):

- `s3:PutObject`
- `s3:PutObjectTagging`
- `s3:GetObject`

- Einrichten der Umgebungsvariable **`SERVICE_REGION`** – Legen Sie die Umgebungsvariable `SERVICE_REGION` fest, um die Region zu identifizieren, in der sich Ihr DB-Cluster befindet (eine Liste der Regionskennungen finden Sie unter Herstellen einer Verbindung zu Neptune).

Ausführen des **`neptune-export`**-Hilfsprogramms, um einen Exportvorgang zu starten

Verwenden Sie den folgenden Befehl, um neptune-export von der Befehlszeile aus auszuführen und einen Exportvorgang zu starten:

```
java -jar neptune-export.jar nesvc \
  --root-path (path to a local directory) \
  --json (the JSON file that defines the export)
```

Der Befehl hat zwei Parameter:

Parameter für neptune-export beim Starten eines Exports

- **`--root-path`** – Pfad zu einem lokalen Verzeichnis, in das Exportdateien geschrieben werden, bevor sie in Amazon S3 veröffentlicht werden.
- **`--json`** – Ein JSON-Objekt, das den Export definiert.

Beispielbefehle mit dem **neptune-export**-Befehlszeilen-Dienstprogramm

So exportieren Sie Eigenschaftsgraph-Daten direkt aus Ihrem Quell-DB-Cluster:

```
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)"
            }
          }'
```

So exportieren Sie RDF-Daten direkt aus Ihrem Quell-DB-Cluster:

```
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)"
            }
          }'
```

Wenn Sie den `command`-Anforderungsparameter weglassen, exportiert das `neptune-export`-Hilfsprogramm standardmäßig Eigenschaftsgraph-Daten aus Neptune.

So exportieren Sie aus einem Klon Ihres DB-Clusters:

```
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
            }
          }'
```

So exportieren Sie mithilfe der IAM-Authentifizierung aus Ihrem DB-Cluster:

```
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
            }
          }'
```

Von Neptune-Export exportierte Dateien und **`neptune-export`**

Wenn ein Export abgeschlossen ist, werden die Exportdateien an dem von Ihnen angegebenen Amazon-S3-Speicherort veröffentlicht. Alle in Amazon S3 veröffentlichten Dateien werden mit der serverseitigen Verschlüsselung von Amazon S3 (`SSE-S3`) verschlüsselt. Die in Amazon S3 veröffentlichten Ordner und Dateien variieren je nachdem, ob Sie Eigenschaften-/Graph- oder RDF-Daten exportieren. Wenn Sie den Amazon-S3-Speicherort öffnen, an dem die Dateien veröffentlicht werden, sehen Sie den folgenden Inhalt:

Speicherorte der exportierten Dateien in Amazon S3

- **`nodes/`** – Dieser Ordner enthält Knotendatendateien im CSV- oder JSON-Format.

  In Neptune können Knoten ein oder mehrere Etiketten haben. Knoten mit unterschiedlichen individuellen Etiketten (oder unterschiedlichen Kombinationen mehrerer Etiketten) werden in verschiedene Dateien geschrieben, was bedeutet, dass keine einzelne Datei Daten für Knoten mit unterschiedlichen Kombinationen von Etiketten enthält. Wenn ein Knoten mehrere Etiketten hat, werden diese Etiketten alphabetisch sortiert, bevor sie einer Datei zugewiesen werden.
- **`edges/`** – Dieser Ordner enthält Edge-Datendateien im CSV- oder JSON-Format.

  Wie bei den Knotendateien werden Edge-Daten auf der Grundlage einer Kombination ihrer Etiketten in verschiedene Dateien geschrieben. Für das Modelltraining werden Edge-Daten verschiedenen Dateien zugewiesen, die auf einer Kombination aus dem Edge-Etikett und den Etiketten der Start- und Endknoten des Edges basieren.
- **`statements/`** – Dieser Ordner enthält RDF-Datendateien im Turtle-, N-Quads-, N-Triples- oder JSON-Format.

- **`config.json`** – Diese Datei enthält das Schema des Graphen, wie es durch den Exportvorgang abgeleitet wurde.
- **`lastEventId.json`** – Diese Datei enthält `commitNum` und `opNum` des letzten Ereignisses in den Neptun-Streams der Datenbank. Der Exportvorgang enthält diese Datei nur, wenn Sie den `includeLastEventId`-Exportparameter auf `true` setzen und in der Datenbank, aus der Sie Daten exportieren, Neptune-Streams aktiviert sind.

Parameter, die zur Steuerung des Neptun-Exportprozesses verwendet werden

Unabhängig davon, ob Sie den Neptune-Export-Service oder das `neptune-export`-Befehlszeilen-Dienstprogramm verwenden, sind die Parameter, die Sie zur Steuerung des Exports verwenden, größtenteils dieselben. Sie enthalten ein JSON-Objekt, das an den Neptune-Export-Endpunkt oder an `neptune-export` auf der Befehlszeile übergeben wird.

Das an den Exportprozess übergebene Objekt hat die folgenden Felder der obersten Ebene:

```
-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) }
    }'
```

Inhalt

- Der Parameter command
- Der Parameter outputS3Path
- Der Parameter expectedBucketOwner
- Der Parameter jobSize
- Das params-Objekt
- Das additionalParams-Objekt
- Exportieren Sie Parameterfelder im params-JSON-Objekt der obersten Ebene

- Liste der möglichen Felder im params-Exportparameterobjekt
  - Liste der Felder, die allen Exporttypen gemeinsam sind
  - Liste der Felder für den Export von Eigenschaftsgraphen
  - Liste der Felder für RDF-Exporte
- Felder, die allen Exporttypen gemeinsam sind
  - cloneCluster-Feld in params
  - cloneClusterInstanceType-Feld in params
  - cloneClusterReplicaCount-Feld in params
  - cloneClusterEnableAuditLogs-Feld in params
  - clusterId-Feld in params
  - endpoint-Feld in params
  - endpoints-Feld in params
  - profile-Feld in params
  - useIamAuth-Feld in params
  - includeLastEventId-Feld in params
- Felder für den Export von Eigenschaftsgraphen
  - concurrency-Feld in params
  - edgeLabels-Feld in params
  - filter-Feld in params
  - filterConfigFile-Feld in params
  - format-Feld, das für Eigenschaftsgraphdaten in params verwendet wird
  - gremlinFilter-Feld in params
  - gremlinNodeFilter-Feld in params
  - gremlinEdgeFilter-Feld in params
  - nodeLabels-Feld in params
  - scope-Feld in params
- Felder für den RDF-Export
  - format-Feld, das für RDF-Daten in params verwendet wird
  - rdfExportScope-Feld in params
  - sparql-Feld in params

        - namedGraph-Feld in params
- Beispiele für das Filtern der exportierten Inhalte
  - Filtern des Exports von Eigenschaftsgraphdaten
    - Beispiel für die Verwendung von scope, um nur Edges zu exportieren
    - Beispiel für die Verwendung von nodeLabels und edgeLabels, um nur Knoten und Edges mit bestimmten Etiketten zu exportieren
    - Beispiel für die Verwendung von filter, um nur bestimmte Knoten, Edges und Eigenschaften zu exportieren
    - Beispiel mit gremlinFilter
    - Beispiel mit gremlinNodeFilter
    - Beispiel mit gremlinEdgeFilter
    - Kombination von filter, gremlinNodeFilter, nodeLabels, edgeLabels und scope
  - Filtern des Exports von RDF-Daten
    - Verwendung rdfExportScope und sparql zum Exportieren bestimmter Edges
    - Wird verwendetnamedGraph, um ein einzelnes benanntes Diagramm zu exportieren

Der Parameter **`command`**

Der Parameter `command` auf der obersten Ebene bestimmt, ob Eigenschaftsgraphdaten oder RDF-Daten exportiert werden. Wenn Sie den `command`-Parameter weglassen, werden beim Export standardmäßig Eigenschaftsgraphdaten exportiert.

- **`export-pg`** – Export von Daten aus Eigenschaftsgraphen
- **`export-rdf`** – Export von RDF-Daten.

Der Parameter **`outputS3Path`**

Der Parameter `outputS3Path` der obersten Ebene ist erforderlich und muss die URI eines Amazon-S3-Speicherorts enthalten, an dem die exportierten Dateien veröffentlicht werden können:

```
  "outputS3Path" : "s3://(your Amazon S3 bucket)/(path to output folder)"
```

Der Wert muss mit `s3://` beginnen, gefolgt von einem gültigen Bucket-Namen und optional einem Ordnerpfad innerhalb des Buckets.

Der Parameter `expectedBucketOwner`

Der Parameter der `expectedBucketOwner` obersten Ebene ist optional. Wenn es auf eine AWS Konto-ID gesetzt ist, bestätigt es, dass der Amazon S3 S3-Bucket, der in angegeben `outputS3Path` ist, diesem Konto gehört. Wenn es weggelassen wird, wird standardmäßig die Konto-ID verwendet, die aus den für Amazon S3 S3-Operationen verwendeten Anmeldeinformationen aufgelöst wurde.

```
  "expectedBucketOwner" : "(your AWS account ID for Amazon S3 bucket)"
```

Der Parameter `jobSize`

Der Parameter `jobSize` der obersten Ebene wird nur mit dem Neptune-Export-Service verwendet, nicht mit dem `neptune-export`-Befehlszeilen-Dienstprogramm, und ist optional. Damit können Sie die Größe des Exportauftrags, den Sie starten, angeben. Auf diese Weise können Sie die Menge der Computing-Ressourcen, die für den Auftrag bereitgestellt werden, und den maximalen Parallelitätsgrad bestimmen.

```
  "jobSize" : "(one of four size descriptors)"
```

Die vier gültigen Größendeskriptoren sind:

- `small` – Maximale Parallelität: 8. Geeignet für Speichervolumina von bis zu 10 GB.
- `medium` – Maximale Parallelität: 32. Geeignet für Speichervolumina von bis zu 100 GB.
- `large` – Maximale Parallelität: 64. Geeignet für Speichervolumina über 100 GB, aber unter 1 TB.
- `xlarge` – Maximale Parallelität: 96. Geeignet für Speichervolumina über 1 TB.

Standardmäßig wird ein auf dem Neptune-Export-Service initiierter Export als `small` Auftrag ausgeführt.

Die Leistung eines Exports hängt nicht nur von der `jobSize`-Einstellung ab, sondern auch von der Anzahl der Datenbank-Instances, aus denen Sie exportieren, der Größe jeder Instance und dem effektiven Parallelitätsgrad des Auftrags.

Für Exporte von Eigenschaftsgraphen können Sie die Anzahl der Datenbank-Instances mithilfe des cloneClusterReplicaZählen-Parameters konfigurieren und mithilfe des concurrency-Parameters den effektiven Parallelitätsgrad des Auftrags konfigurieren.

Das **`params`**-Objekt

Der Parameter `params` der obersten Ebene ist ein JSON-Objekt, das Parameter enthält, mit denen Sie den Exportvorgang selbst steuern, wie unter Exportieren Sie Parameterfelder im `params`-JSON-Objekt der obersten Ebene erläutert. Einige Felder im `params`-Objekt sind spezifisch für Eigenschaftsgraph-Exporte, andere für RDF.

Das **`additionalParams`**-Objekt

Der Parameter `additionalParams` der obersten Ebene ist ein JSON-Objekt, das Parameter enthält, mit denen Sie Aktionen steuern können, die nach dem Export auf die Daten angewendet werden. `additionalParams` wird derzeit nur für den Export von Trainingsdaten für Neptune ML verwendet.

Exportieren Sie Parameterfelder im **`params`**-JSON-Objekt der obersten Ebene

Mit dem `params`-Neptune-Export-JSON-Objekt können Sie den Export steuern, einschließlich des Typs und des Formats der exportierten Daten.

Liste der möglichen Felder im **`params`**-Exportparameterobjekt

Im Folgenden sind alle möglichen Felder der obersten Ebene aufgeführt, die in einem `params`-Objekt vorkommen können. Nur eine Teilmenge dieser Felder kommt in einem Objekt vor.

Liste der Felder, die allen Exporttypen gemeinsam sind

- `cloneCluster`
- `cloneClusterInstanceType`
- `cloneClusterReplicaCount`
- `cloneClusterEnableAuditLogs`
- `clusterId`
- `endpoint`
- `endpoints`
- `profile`
- `useIamAuth`
- `includeLastEventId`

Liste der Felder für den Export von Eigenschaftsgraphen

- `concurrency`
- `edgeLabels`
- `filter`
- `filterConfigFile`
- `gremlinFilter`
- `gremlinNodeFilter`
- `gremlinEdgeFilter`
- `format`

- nodeLabels
- scope

Liste der Felder für RDF-Exporte

- format
- rdfExportScope
- sparql
- namedGraph

Felder, die allen Exporttypen gemeinsam sind

`cloneCluster`-Feld in **`params`**

(Optional). Standard: `false`.

Wenn der `cloneCluster`-Parameter auf `true` gesetzt ist, verwendet der Exportvorgang einen schnellen Klon Ihres DB-Clusters:

```
  "cloneCluster" : true
```

Standardmäßig exportiert der Exportvorgang Daten aus dem DB-Cluster, den Sie mit den Parametern `endpoint`, `endpoints` oder `clusterId` angeben. Wenn Ihr DB-Cluster jedoch während des Exports verwendet wird und sich Daten ändern, kann der Exportvorgang die Konsistenz der exportierten Daten nicht garantieren.

Um sicherzustellen, dass die exportierten Daten konsistent sind, verwenden Sie stattdessen den `cloneCluster`-Parameter, um aus einem statischen Klon Ihres DB-Clusters zu exportieren.

Der geklonte DB-Cluster wird in derselben VPC wie der Quell-DB-Cluster erstellt und übernimmt die Sicherheitsgruppe, die Subnetzgruppe und die IAM-Datenbankauthentifizierungse instellungen der Quelle. Wenn der Export abgeschlossen ist, löscht Neptune den geklonten DB-Cluster.

Standardmäßig besteht ein geklonter DB-Cluster aus einer einzigen Instance desselben Instance-Typs wie die primäre Instance im Quell-DB-Cluster. Sie können den Instance-Typ ändern, der für den geklonten DB-Cluster verwendet wird, indem Sie mit `cloneClusterInstanceType` einen anderen Instance-Typ angeben.

> **Note**
>
> Wenn Sie die `cloneCluster`-Option nicht verwenden und direkt aus Ihrem Haupt-DB-Cluster exportieren, müssen Sie möglicherweise das Timeout für die Instances erhöhen, aus denen Daten exportiert werden. Bei großen Datensätzen sollte das Timeout auf mehrere Stunden festgelegt werden.

`cloneClusterInstanceType`-Feld in **`params`**

(Optional).

Wenn der `cloneCluster`-Parameter vorhanden und auf `true` gesetzt ist, können Sie den `cloneClusterInstanceType`-Parameter verwenden, um den Instance-Typ anzugeben, der für den geklonten DB-Cluster verwendet wird:

Standardmäßig besteht ein geklonter DB-Cluster aus einer einzigen Instance desselben Instance-Typs wie die primäre Instance im Quell-DB-Cluster.

```
  "cloneClusterInstanceType" : "(for example, r5.12xlarge)"
```

`cloneClusterReplicaCount`-Feld in **`params`**

(Optional).

Wenn der `cloneCluster`-Parameter vorhanden und auf `true` gesetzt ist, können Sie den `cloneClusterReplicaCount`-Parameter verwenden, um die Anzahl der Lesereplikat-Instances anzugeben, die im geklonten DB-Cluster erstellt wurden:

```
  "cloneClusterReplicaCount" : (for example, 3)
```

Standardmäßig besteht ein geklonter DB-Cluster aus einer einzigen primären Instance. Mit dem `cloneClusterReplicaCount`-Parameter können Sie angeben, wie viele zusätzliche Lesereplikat-Instances erstellt werden sollen.

`cloneClusterEnableAuditLogs`-Feld in **`params`**

(Optional). Standard: falsch.

Wenn der `cloneCluster` Parameter vorhanden und auf true gesetzt ist, können Sie den `cloneClusterEnableAuditLogs` Parameter verwenden, um Audit-Logs im geklonten Cluster zu aktivieren oder zu deaktivieren.

Standardmäßig ist die Auditprotokollierung deaktiviert.

```
"cloneClusterEnableAuditLogs" : true
```

`clusterId`-Feld in **`params`**

(Optional).

Der `clusterId`-Parameter gibt die ID eines zu verwendenden DB-Clusters an:

```
  "clusterId" : "(the ID of your DB cluster)"
```

Wenn Sie den `clusterId`-Parameter verwenden, verwendet der Exportvorgang alle verfügbaren Instances in diesem DB-Cluster, um Daten zu extrahieren.

> **Note**
>
> Die Parameter `endpoint`, `endpoints` und `clusterId` schließen sich gegenseitig aus. Verwenden Sie stets nur einen davon.

`endpoint`-Feld in **`params`**

(Optional).

Verwenden Sie `endpoint`, um einen Endpunkt einer Neptune-Instance in Ihrem DB-Cluster anzugeben, den der Exportprozess abfragen kann, um Daten zu extrahieren (siehe Endpunktverbindungen). Dies ist nur der DNS-Name und beinhaltet nicht das Protokoll oder den Port:

```
  "endpoint" : "(a DNS endpoint of your DB cluster)"
```

Verwenden Sie einen Cluster oder Instance-Endpunkt, aber nicht den Haupt-Leser-Endpunkt.

> **Note**
>
> Die Parameter `endpoint`, `endpoints` und `clusterId` schließen sich gegenseitig aus. Verwenden Sie stets nur einen davon.

`endpoints`-Feld in **`params`**

(Optional).

Verwenden Sie `endpoints`, um ein JSON-Array von Endpunkten in Ihrem DB-Cluster anzugeben, das der Exportprozess abfragen kann, um Daten zu extrahieren (siehe Endpunktverbindungen). Dies sind nur DNS-Namen ohne Protokoll oder Port:

```
"endpoints": [
  "(one endpoint in your DB cluster)",
  "(another endpoint in your DB cluster)",
  "(a third endpoint in your DB cluster)"
  ]
```

Wenn Sie mehrere Instances in Ihrem Cluster haben (eine primäre und ein oder mehrere Lesereplikate), können Sie die Exportleistung verbessern, indem Sie den `endpoints`-Parameter verwenden, um Abfragen auf eine Liste dieser Endpunkte zu verteilen.

> **Note**
>
> Die Parameter `endpoint`, `endpoints` und `clusterId` schließen sich gegenseitig aus. Verwenden Sie stets nur einen davon.

`profile`-Feld in **`params`**

(Erforderlich, um Trainingsdaten für Neptune ML zu exportieren, sofern das *`neptune_ml`*-Feld nicht im *`additionalParams`*-Feld vorhanden ist).

Der `profile`-Parameter stellt Sätze vorkonfigurierter Parameter für bestimmte Workloads bereit. Derzeit unterstützt der Exportvorgang nur das Profil `neptune_ml`

Wenn Sie Trainingsdaten für Neptune ML exportieren, fügen Sie dem `params`-Objekt den folgenden Parameter hinzu:

```
  "profile" : "neptune_ml"
```

`useIamAuth`-Feld in **`params`**

(Optional). Standard: `false`.

Wenn für die Datenbank, aus der Sie Daten exportieren, die IAM-Authentifizierung aktiviert ist, müssen Sie den Parameter `useIamAuth` mit der Einstellung auf `true` einschließen:

```
  "useIamAuth" : true
```

`includeLastEventId`-Feld in **`params`**

Wenn Sie den Wert `includeLastEventId` auf „true“ setzen und in der Datenbank, aus der Sie Daten exportieren, Neptune Streams aktiviert ist, schreibt der Exportvorgang eine `lastEventId.json`-Datei an den angegebenen Exportspeicherort. Diese Datei enthält das `commitNum` und das `opNum` des letzten Ereignisses im Stream.

```
  "includeLastEventId" : true
```

Eine durch den Exportvorgang erstellte geklonte Datenbank übernimmt die Streams-Einstellung der übergeordneten Datenbank. Wenn für das übergeordnete Objekt Streams aktiviert sind, sind auch Streams für den Klon aktiviert. Der Inhalt des Streams auf dem Clone entspricht dem Inhalt des übergeordneten Streams (einschließlich desselben Ereignisses IDs) zu dem Zeitpunkt, zu dem der Clone erstellt wurde.

Felder für den Export von Eigenschaftsgraphen

`concurrency`-Feld in **`params`**

(Optional). Standard: `4`.

Der `concurrency`-Parameter gibt die Anzahl der parallelen Abfragen an, die der Exportvorgang verwenden soll:

```
  "concurrency" : (for example, 24)
```

Eine gute Richtlinie besteht darin, die Parallelitätsstufe für alle Instances, aus denen Sie Daten exportieren, CPUs auf die doppelte Anzahl von v festzulegen. Eine r5.xlarge-Instance hat

beispielsweise 4 v. CPUs Wenn Sie aus einem Cluster von 3 r5.xlarge-Instances exportieren, können Sie die Parallelitätsebene auf 24 (= 3 x 2 x 4) festlegen.

Wenn Sie den Neptune-Export-Service verwenden, ist die Parallelitätsebene durch die JobSize-Einstellung begrenzt. Ein kleiner Auftrag unterstützt beispielsweise eine Parallelitätsebene von 8. Wenn Sie versuchen, mithilfe des `concurrency`-Parameters eine Parallelitätsebene von 24 für einen kleinen Auftrag anzugeben, bleibt die effektive Ebene bei 8.

Wenn Sie aus einem geklonten Cluster exportieren, berechnet der Exportprozess auf der Grundlage der Größe der geklonten Instances und der Auftragsgröße eine angemessene Parallelitätsebene.

`edgeLabels`-Feld in `params`

(Optional).

Verwenden Sie `edgeLabels`, um nur die Edges zu exportieren, deren Etiketten Sie angeben:

```
  "edgeLabels" : ["(a label)", "(another label"]
```

Jedes Etikett im JSON-Array muss ein einzelnes, einfaches Etikett sein.

Der `scope`-Parameter hat Vorrang vor dem `edgeLabels`-Parameter. Wenn der `scope`-Wert also keine Edges enthält, hat der `edgeLabels`-Parameter keine Wirkung.

`filter`-Feld in `params`

(Optional).

Verwenden Sie diese Option, `filter` um anzugeben, dass nur Knoten, and/or Kanten mit bestimmten Beschriftungen exportiert werden sollen, und um die Eigenschaften zu filtern, die für jeden Knoten oder jede Kante exportiert werden.

Die allgemeine Struktur eines `filter`-Objekts, entweder direkt oder in einer Filterkonfigurationsdatei, sieht wie folgt aus:

```
  "filter" : {
    "nodes": [ (array of node label and properties objects) ],
    "edges": [ (array of edge definition an properties objects) ]
  }
```

- **`nodes`** – Enthält ein JSON-Array von Knoten und Knoteneigenschaften in der folgenden Form:

```
"nodes : [
  {
    "label": "(node label)",
    "properties": [ "(a property name)", "(another property name)", ( ... ) ]
  }
]
```

  - `label` – Etikett(en) des Eigenschaftsgraphen des Knotens.

    Akzeptiert einen einzelnen Wert oder, wenn der Knoten mehrere Etiketten hat, ein Array von Werten.
  - `properties` – Enthält ein Array mit den Namen der Eigenschaften des Knotens, die Sie exportieren möchten.
- **`edges`** – Enthält ein JSON-Array mit Edge-Definitionen in der folgenden Form:

```
"edges" : [
  {
    "label": "(edge label)",
    "properties": [ "(a property name)", "(another property name)", ( ... ) ]
  }
]
```

  - `label` – Etikett des Edge-Eigenschaftsgraphen. Nimmt einen einzelnen Wert an.
  - `properties` – Enthält ein Array mit den Namen der Eigenschaften des Edges, die Sie exportieren möchten.

`filterConfigFile`-Feld in **`params`**

(Optional).

Verwenden Sie `filterConfigFile`, um eine JSON-Datei anzugeben, die eine Filterkonfiguration in derselben Form enthält, die der `filter`-Parameter annimmt:

```
 "filterConfigFile" : "s3://(your Amazon S3 bucket)/neptune-export/(the name of the
JSON file)"
```

Informationen zum Format der `filterConfigFile`-Datei finden Sie unter Filter.

`format`-Feld, das für Eigenschaftsgraphdaten in **`params`** verwendet wird

(Optional). Standart: `csv` (CSV)

Der `format`-Parameter gibt das Ausgabeformat der exportierten Eigenschaftsgraphdaten an:

```
"format" : (one of: csv, csvNoHeaders, json, neptuneStreamsJson)
```

- **`csv`** – Ausgabe im CSV-Format mit Spaltenüberschriften, die gemäß dem Gremlin-Ladedatenformat formatiert sind.
- **`csvNoHeaders`** – CSV-formatierte Daten ohne Spaltenüberschriften.
- **`json`** – JSON-formatierte Daten.
- **`neptuneStreamsJson`** – JSON-formatierte Daten, die das GREMLIN_JSON-Änderungsserialisierungsformat verwenden.

`gremlinFilter`-Feld in **`params`**

(Optional).

Mit dem `gremlinFilter`-Parameter können Sie ein Gremlin-Snippet angeben, z. B. einen `has()`-Schritt, mit dem sowohl Knoten als auch Edges gefiltert werden:

```
"gremlinFilter" : (a Gremlin snippet)
```

Feldnamen und Zeichenfolgenwerte sollten von doppelten Escape-Anführungszeichen umgeben sein. Für Datums- und Uhrzeitangaben können Sie die Datetime-Methode verwenden.

Im folgenden Beispiel werden nur die Knoten und Edges exportiert, deren Erstellungsdatum nach dem 10.10.2021 liegt:

```
"gremlinFilter" : "has(\"created\", gt(datetime(\"2021-10-10\")))"
```

`gremlinNodeFilter`-Feld in **`params`**

(Optional).

Mit dem `gremlinNodeFilter`-Parameter können Sie ein Gremlin-Snippet angeben, z. B. einen `has()`-Schritt, mit dem Knoten gefiltert werden:

```
"gremlinNodeFilter" : (a Gremlin snippet)
```

Feldnamen und Zeichenfolgenwerte sollten von doppelten Escape-Anführungszeichen umgeben sein. Für Datums- und Uhrzeitangaben können Sie die Datetime-Methode verwenden.

Das folgende Beispiel exportiert nur die Knoten mit einer booleschen `deleted` Eigenschaft, deren Wert `true` ist:

```
"gremlinNodeFilter" : "has(\"deleted\", true)"
```

`gremlinEdgeFilter`-Feld in **`params`**

(Optional).

Mit dem `gremlinEdgeFilter`-Parameter können Sie ein Gremlin-Snippet angeben, z. B. einen `has()`-Schritt, mit dem Edges gefiltert werden:

```
"gremlinEdgeFilter" : (a Gremlin snippet)
```

Feldnamen und Zeichenfolgenwerte sollten von doppelten Escape-Anführungszeichen umgeben sein. Für Datums- und Uhrzeitangaben können Sie die Datetime-Methode verwenden.

Das folgende Beispiel exportiert nur die Edges mit einer numerischen `strength`-Eigenschaft, deren Wert 5 ist:

```
"gremlinEdgeFilter" : "has(\"strength\", 5)"
```

`nodeLabels`-Feld in **`params`**

(Optional).

Verwenden Sie `nodeLabels`, um nur die Knoten zu exportieren, deren Etiketten Sie angeben:

```
"nodeLabels" : ["(a label)", "(another label"]
```

Jedes Etikett im JSON-Array muss ein einzelnes, einfaches Etikett sein.

Der `scope`-Parameter hat Vorrang vor dem `nodeLabels`-Parameter. Wenn der `scope`-Wert keine Knoten enthält, hat der `nodeLabels`-Parameter daher keine Wirkung.

`scope`-Feld in **`params`**

(Optional). Standard: `all`.

Der `scope`-Parameter gibt an, ob nur Knoten, nur Edges oder Knoten und Edges exportiert werden sollen:

```
  "scope" : (one of: nodes, edges, or all)
```

- `nodes` – Exportiert nur Knoten und ihre Eigenschaften.
- `edges` – Exportiert nur Edges und ihre Eigenschaften.
- `all` – Exportiert sowohl Knoten als auch Edges und ihre Eigenschaften (Standard).

Felder für den RDF-Export

`format`-Feld, das für RDF-Daten in **`params`** verwendet wird

(Optional). Standardwert: `turtle`

Der `format`-Parameter gibt das Ausgabeformat der exportierten RDF-Daten an:

```
  "format" : (one of: turtle, nquads, ntriples, neptuneStreamsJson)
```

- **`turtle`** – Ausgabe im Turtle-Format.
- **`nquads`** – N-Quads-formatierte Daten ohne Spaltenüberschriften.
- **`ntriples`** – N-Triples-formatierte Daten.
- **`neptuneStreamsJson`** – JSON-formatierte Daten, die das SPARQL NQUADS-Änderungsserialisierungsformat verwenden.

`rdfExportScope`-Feld in **`params`**

(Optional). Standard: `graph`.

Der `rdfExportScope`-Parameter gibt den Umfang des RDF-Exports an:

```
  "rdfExportScope" : (one of: graph, edges, or query)
```

- `graph` – Exportiert alle RDF-Daten.

- `edges` – Exportiert nur die Triples, die Edges repräsentieren.
- `query` – Exportiert Daten, die durch eine SPARQL-Abfrage abgerufen wurden, die mithilfe des Felds `sparql` bereitgestellt wird.

`sparql`-Feld in **`params`**

(Optional).

Mit dem `sparql`-Parameter können Sie eine SPARQL-Abfrage angeben, um die zu exportierenden Daten abzurufen:

```
  "sparql" : (a SPARQL query)
```

Wenn Sie eine Abfrage mithilfe des `sparql`-Felds angeben, müssen Sie das `rdfExportScope`-Feld auch auf `query` setzen.

`namedGraph`-Feld in **`params`**

(Optional).

Mit dem `namedGraph` Parameter können Sie einen IRI angeben, um den Export auf ein einzelnes benanntes Diagramm zu beschränken:

```
  "namedGraph" : (Named graph IRI)
```

Der `namedGraph` Parameter kann nur verwendet werden, wenn das `rdfExportScope` Feld auf gesetzt ist`graph`.

Beispiele für das Filtern der exportierten Inhalte

Im Folgenden finden Sie Beispiele, die veranschaulichen, wie die exportierten Daten gefiltert werden können.

Filtern des Exports von Eigenschaftsgraphdaten

Beispiel für die Verwendung von **`scope`**, um nur Edges zu exportieren

```
{
  "command": "export-pg",
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
    "scope": "edges"
  },
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export"
}
```

Beispiel für die Verwendung von **`nodeLabels`** und **`edgeLabels`**, um nur Knoten und Edges mit bestimmten Etiketten zu exportieren

Der `nodeLabels`-Parameter im folgenden Beispiel gibt an, dass nur Knoten mit dem Etikett `Person` oder `Post` exportiert werden sollen. Der `edgeLabels`-Parameter gibt an, dass nur Edges mit dem Etikett `likes` exportiert werden sollen:

```
{
  "command": "export-pg",
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
    "nodeLabels": ["Person", "Post"],
    "edgeLabels": ["likes"]
  },
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export"
}
```

Beispiel für die Verwendung von **`filter`**, um nur bestimmte Knoten, Edges und Eigenschaften zu exportieren

Das `filter`-Objekt in diesem Beispiel exportiert `country`-Knoten mit den Eigenschaften `type`, `code` und `desc` sowie `route`-Edges mit der Eigenschaft `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"
}
```

Beispiel mit **`gremlinFilter`**

In diesem Beispiel werden mit `gremlinFilter` nur die Knoten und Edges exportiert, die nach dem 10.10.2021 erstellt wurden (d. h. mit einer `created`-Eigenschaft, deren Wert größer als 2021-10-10 ist):

```
{
  "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"
}
```

Beispiel mit **`gremlinNodeFilter`**

In diesem Beispiel werden mit `gremlinNodeFilter` nur gelöschte Knoten exportiert (Knoten mit der booleschen Eigenschaft `deleted`, deren Wert `true` ist):

```
{
  "command": "export-pg",
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
    "gremlinNodeFilter" : "has(\"deleted\", true)"
  },
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export"
}
```

Beispiel mit **`gremlinEdgeFilter`**

In diesem Beispiel werden mit `gremlinEdgeFilter` nur Edges mit einer numerischen `strength`-Eigenschaft exportiert, deren Wert 5 ist:

```
{
  "command": "export-pg",
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
    "gremlinEdgeFilter" : "has(\"strength\", 5)"
  },
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export"
}
```

Kombination von **`filter`**, **`gremlinNodeFilter`**, **`nodeLabels`**, **`edgeLabels`** und **`scope`**

Das `filter`-Objekt in diesem Beispiel exportiert:

- `country`-Knoten mit ihren Eigenschaften `type`, `code` und `desc`
- `airport`-Knoten mit ihren Eigenschaften `code`, `icao` und `runways`
- `route`-Edges mit ihrer Eigenschaft `dist`

Der Parameter `gremlinNodeFilter` filtert die Knoten so, dass nur Knoten mit einer `code`-Eigenschaft exportiert werden, deren Wert mit A beginnt.

Die Parameter `nodeLabels` und `edgeLabels` schränken die Ausgabe weiter ein, so dass nur `airport`-Knoten und `route`-Edges exportiert werden.

Schließlich entfernt der Parameter `scope` Edges aus dem Export, so dass nur die angegebenen `airport`-Knoten in der Ausgabe übrig bleiben.

```
{
  "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"
}
```

Filtern des Exports von RDF-Daten

Verwendung **`rdfExportScope`** und **`sparql`** zum Exportieren bestimmter Edges

In diesem Beispiel werden Tripel exportiert, deren Prädikat < http://kelvinlawrence ist. net/air-routes/objectProperty/route> und dessen Objekt kein Literal ist:

```
{
  "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"
}
```

Wird verwendet**`namedGraph`**, um ein einzelnes benanntes Diagramm zu exportieren

In diesem Beispiel werden Tripel exportiert, die zum benannten Graphen gehören < 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"
}
```

Problembehandlung beim Neptune-Exportprozess

Der Amazon-Neptune-Exportprozess verwendet AWS Batch für die Bereitstellung der Computing- und Speicherressourcen, die für den Export Ihrer Neptune-Daten erforderlich sind. Wenn ein Export

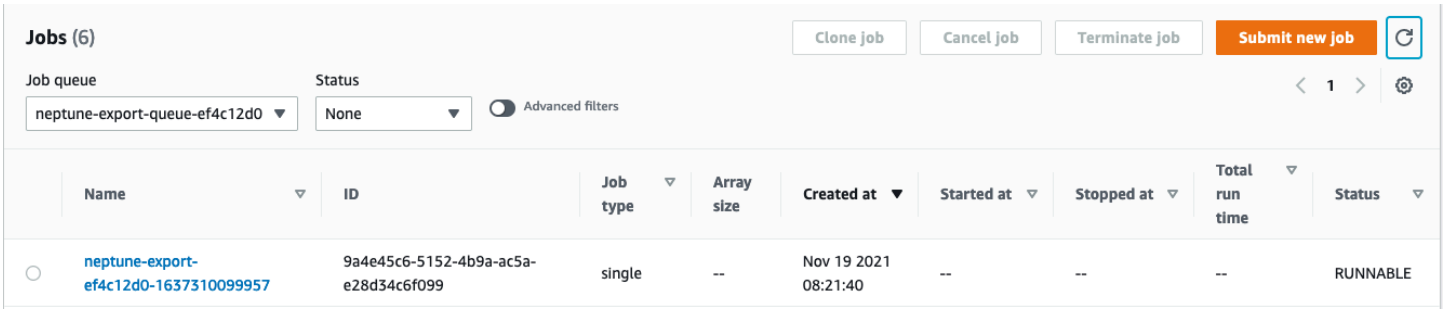
ausgeführt wird, können Sie den Link im `logs` Feld verwenden, um auf die CloudWatch Protokolle für den Exportjob zuzugreifen.

Die CloudWatch Protokolle für den AWS Batch Job, der den Export ausführt, sind jedoch nur verfügbar, wenn der AWS Batch Job ausgeführt wird. Wenn Neptune Export meldet, dass sich ein Export im Status „Ausstehend“ befindet, wird es keinen Log-Link geben, über den Sie auf Protokolle zugreifen CloudWatch können. Wenn ein Exportauftrag länger als ein paar Minuten im `pending` Status verbleibt, liegt möglicherweise ein Problem bei der Bereitstellung der zugrunde liegenden Ressourcen vor. AWS Batch

Wenn der Exportauftrag den Status „Ausstehend“ verlässt, können Sie seinen Status wie folgt überprüfen:

Um den Status eines Auftrags zu AWS Batch überprüfen

1. Öffnen Sie die AWS Batch Konsole unter https://console.aws.amazon.com/batch/.
2. Wählen Sie die Warteschlange für Neptune-Exportauftrags aus.
3. Suchen Sie nach dem Auftrag, dessen Name mit dem vom Neptune-Export zurückgegebenen `jobName` übereinstimmt, als Sie den Export gestartet haben.

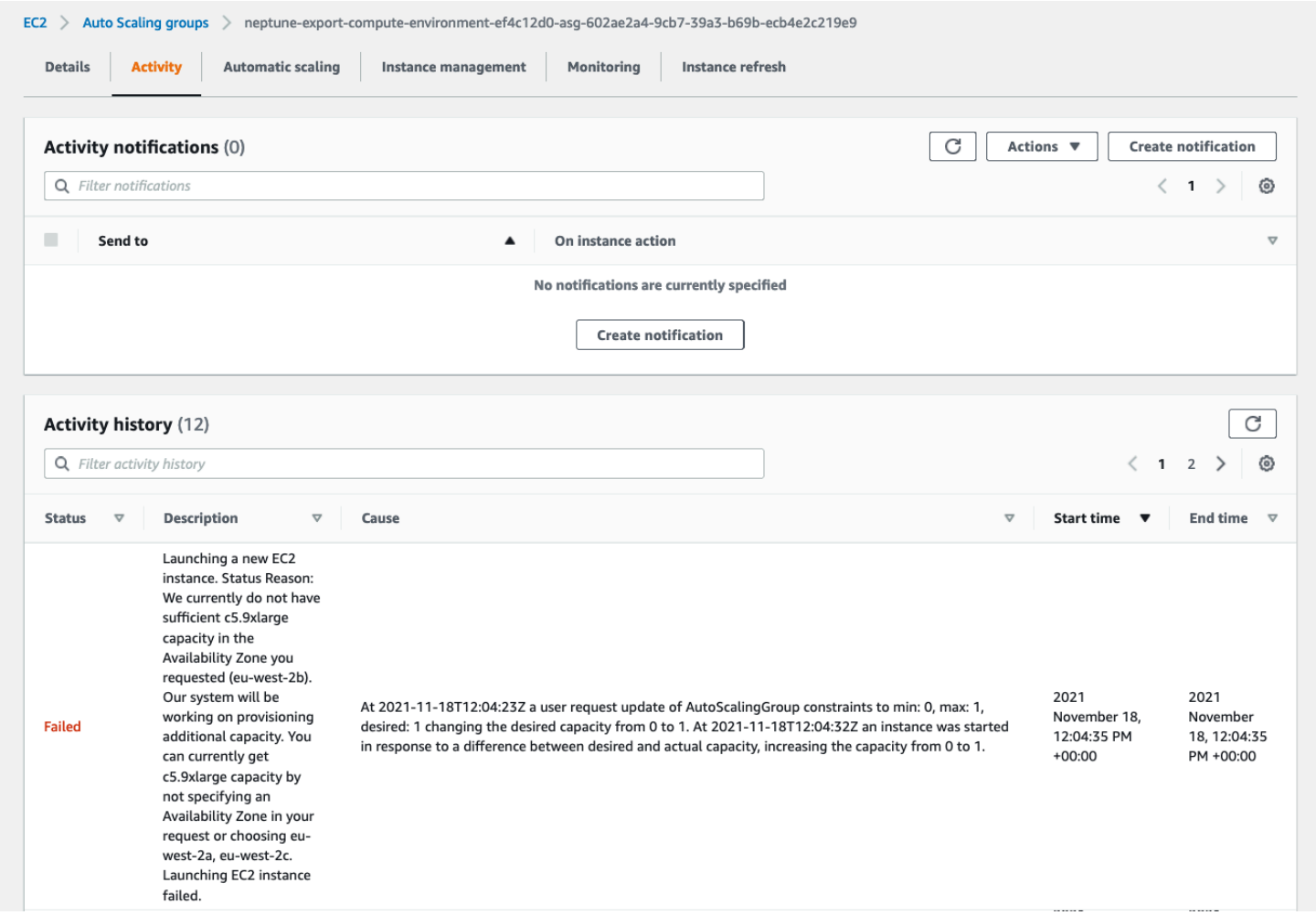


Bleibt der Auftrag im Status `RUNNABLE`, kann dies daran liegen, dass Netzwerk- oder Sicherheitsprobleme verhindern, dass die Container-Instance dem zugrunde liegenden Amazon Elastic Container Service (Amazon ECS)-Cluster beitritt. Weitere Informationen zur Überprüfung der Netzwerk- und Sicherheitseinstellungen der Datenverarbeitungsumgebung finden Sie in diesem Support-Artikel.

Sie können auch mit Auto Scaling nach Problemen suchen:

Um die Amazon EC2 auto-scaling Scaling-Gruppe für die AWS Batch Rechenumgebung zu überprüfen

1. Öffnen Sie die EC2 Amazon-Konsole unter https://console.aws.amazon.com/ec2/.
2. Wählen Sie die Auto Scaling-Gruppe für die Neptune-Export-Datenverarbeitungsumgebung aus.
3. Öffnen Sie die Registerkarte Aktivität und überprüfen Sie den Aktivitätsverlauf auf erfolglose Ereignisse.



Häufige Fehler bei Neptune-Exporten

`org.eclipse.rdf4j.query.QueryEvaluationException: Tag mismatch!`

Wenn ein `export-rdf`-Auftrag regelmäßig mit `Tag mismatch! QueryEvaluationException` fehlschlägt, ist die Neptune-Instance für die großen, lang andauernden Abfragen, die von Neptune Export verwendet werden, zu klein.

Sie können diesen Fehler vermeiden, indem Sie auf eine größere Neptune-Instance skalieren oder den Auftrag wie folgt so konfigurieren, dass er aus einem großen geklonten Cluster exportiert:

```
'{
  "command": "export-rdf",
  "outputS3Path": "s3://(your Amazon S3 bucket)/neptune-export",
  "params": {
    "endpoint": "(your Neptune endpoint DNS name)",
    "cloneCluster": True,
    "cloneClusterInstanceType" : "r5.24xlarge"
  }
}'
```

Exportieren von Gremlin-Abfrageergebnissen nach Amazon S3

Ab Engine-Version 1.4.3.0 unterstützt Amazon Neptune den direkten Export von Gremlin-Abfrageergebnissen nach Amazon S3. Mit dieser Funktion können Sie große Abfrageergebnisse effizient verarbeiten, indem Sie sie in einen Amazon S3 S3-Bucket exportieren, anstatt sie als Abfrageantwort zurückzugeben.

Um Abfrageergebnisse nach Amazon S3 zu exportieren, verwenden Sie den `call()` Schritt mit dem `neptune.query.exportToS3` Servicenamen als letzten Schritt in Ihrer Gremlin-Abfrage. Ein Terminalschritt in Tinkerpop-Treibern, die Bytecode verwenden, kann nach dem Schritt hinzugefügt werden. `call()` Die Exportparameter müssen als Zeichenkettenwerte bereitgestellt werden.

> **Note**
>
> Die Abfrage mit dem `call()` Schritt hat schlägt `neptune.query.exportToS3` fehl, wenn sie nicht als letzter Schritt verwendet wird. Die Gremlin-Clients, die Bytecode verwenden, können Terminalschritte verwenden. Weitere Informationen finden Sie unter Best Practices für Gremlin in der Amazon Neptune Neptune-Dokumentation.

```
g.V()
  ...
  .call('neptune.query.exportToS3', [
    'destination': 's3://your-bucket/path/result.json',
    'format': 'GraphSONv3',
```

```
    'kmskeyArn': 'optional-kms-key-arn'
  ])
```

Parameters

- `destination`: erforderlich — Die Amazon S3 S3-URI, in die die Ergebnisse geschrieben werden.
- `format`: erforderlich — Das Ausgabeformat unterstützt derzeit nur 'Graph SONv3'.
- `keyArn`: optional — Der ARN eines AWS KMS Schlüssels für die serverseitige Amazon S3 S3-Verschlüsselung.

Beispiele

Beispielabfrage

```
g.V().
    hasLabel('Comment').
    valueMap().
    call('neptune.query.exportToS3', [
    'destination': 's3://your-bucket/path/result.json',
    'format': 'GraphSONv3',
    'keyArn': 'optional-kms-key-arn'
  ])
```

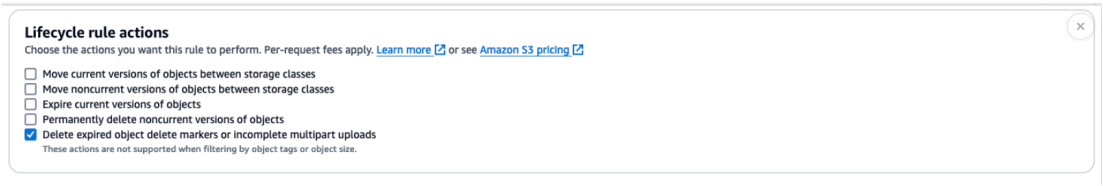
Beispiel für eine Antwort auf eine Anfrage

```
{
    "destination":"s3://your-bucket/path/result.json,
    "exportedResults": 100,
    "exportedBytes": 102400
}
```

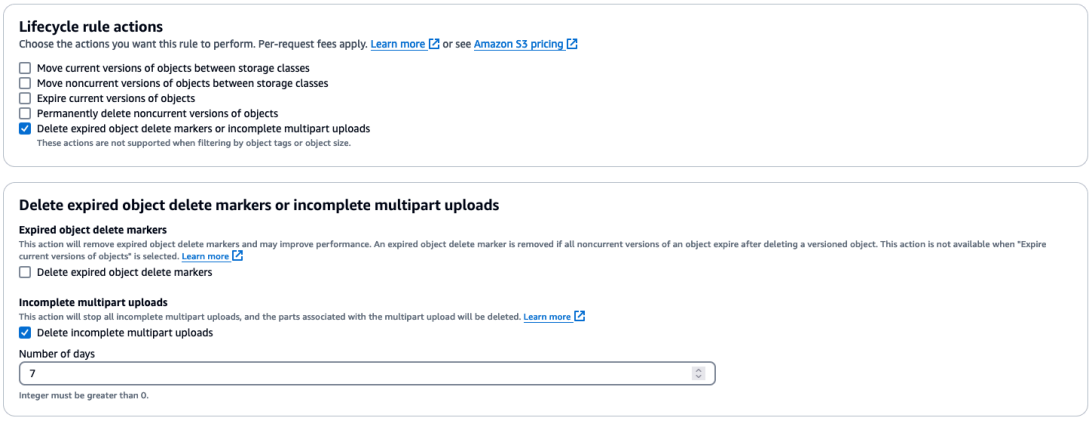
Voraussetzungen

- Ihre Neptune-DB-Instance muss über einen VPC-Endpunkt vom Typ Gateway Zugriff auf Amazon S3 haben.
- Um eine benutzerdefinierte AWS KMS Verschlüsselung in der Abfrage zu verwenden, ist ein VPC-Endpunkt vom Typ Interface für AWS KMS erforderlich, mit dem Neptune kommunizieren kann. AWS KMS

- Sie müssen die IAM-Authentifizierung auf Neptune aktivieren und über die entsprechenden IAM-Berechtigungen verfügen, um in den Amazon S3 S3-Ziel-Bucket zu schreiben. Wenn Sie dies nicht tun, wird der Fehler 400 Bad Request ausgelöst: „Für den S3-Export muss im Cluster die IAM-Authentifizierung aktiviert sein“.
- Der Amazon S3-Ziel-Bucket:
  - Der Amazon S3 S3-Ziel-Bucket darf nicht öffentlich sein. `Block public access`muss aktiviert sein.
  - Das Amazon S3 S3-Zielziel muss leer sein.
  - Für den Amazon S3 S3-Ziel-Bucket muss eine Lebenszyklusregel `Delete expired object delete markers or incomplete multipart uploads` mit aktiviert sein`Delete incomplete multipart uploads`. Weitere Informationen finden Sie unter Amazon S3 Lifecycle Management-Update — Unterstützung für mehrteilige Uploads und Löschmarkierungen.



  - Im Amazon S3 S3-Ziel-Bucket muss eine Lebenszyklusregel aktiviert `Delete expired object delete markers or incomplete multipart uploads` sein, deren Wert höher ist, als die Auswertung der Abfrage dauert (z. B. 7 Tage). `Delete incomplete multipart uploads` Dies ist erforderlich, um unvollständige Uploads (die nicht direkt sichtbar sind, aber Kosten verursachen würden) zu löschen, falls sie von Neptune nicht abgeschlossen oder abgebrochen werden können (z. B. aufgrund von Fehlern). instance/engine Weitere Informationen finden Sie unter Amazon S3 Lifecycle Management-Update — Unterstützung für mehrteilige Uploads und Löschmarkierungen.



Wichtige Überlegungen

- Der Exportschritt muss der letzte Schritt in Ihrer Gremlin-Abfrage sein.
- Wenn am angegebenen Amazon S3 S3-Standort bereits ein Objekt vorhanden ist, schlägt die Abfrage fehl.
- Die maximale Ausführungszeit für Exportabfragen ist auf 11 Stunden und 50 Minuten begrenzt. Diese Funktion verwendet Forward-Access-Sitzungen. Sie ist derzeit auf 11 Stunden und 50 Minuten begrenzt, um Probleme mit dem Ablauf von Tokens zu vermeiden.

> **Note**
>
> Die Exportabfrage berücksichtigt weiterhin das Abfrage-Timeout. Für umfangreiche Exporte sollten Sie ein geeignetes Abfrage-Timeout verwenden.

- Alle Uploads neuer Objekte auf Amazon S3 werden automatisch verschlüsselt.
- Um Speicherkosten durch unvollständige mehrteilige Uploads im Falle von Fehlern oder Abstürzen zu vermeiden, empfehlen wir, eine Lebenszyklusregel für Ihren `Delete incomplete multipart uploads` Amazon S3 S3-Bucket einzurichten.

Reaktionsformat

Anstatt die Abfrageergebnisse direkt zurückzugeben, gibt die Abfrage Metadaten über den Exportvorgang zurück, einschließlich Status und Exportdetails. Die Abfrageergebnisse in Amazon S3 werden im SONv3Graph-Format vorliegen.

```
{
  "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"
          ]
        }
      ]
    },


    ...


  ]
```

```
  }
}
```

Sicherheit

- Alle an Amazon S3 übertragenen Daten werden bei der Übertragung mit SSL verschlüsselt.
- Sie können einen AWS KMS Schlüssel für die serverseitige Verschlüsselung der exportierten Daten angeben. Amazon S3 verschlüsselt neue Daten standardmäßig. Wenn der Bucket für die Verwendung eines bestimmten AWS KMS Schlüssels konfiguriert ist, wird dieser Schlüssel verwendet.
- Neptune überprüft, ob der Ziel-Bucket nicht öffentlich ist, bevor der Export gestartet wird.
- Konten- und regionsübergreifende Exporte werden nicht unterstützt.

Fehlerbehandlung

- Der Amazon S3 S3-Ziel-Bucket ist öffentlich.
- Das angegebene Objekt ist bereits vorhanden.
- Sie verfügen nicht über ausreichende Berechtigungen, um in den Amazon S3 S3-Bucket zu schreiben.
- Die Ausführung der Abfrage überschreitet das maximale Zeitlimit.

Best Practices

- Verwenden Sie die Amazon S3 S3-Bucket-Lebenszyklusregeln, um unvollständige mehrteilige Uploads zu bereinigen.
- Überwachen Sie Ihre Exportvorgänge mithilfe von Neptune-Protokollen und -Metriken. Sie können den Gremlin-Status-Endpunkt überprüfen, um festzustellen, ob gerade eine Abfrage ausgeführt wird. Solange der Client keine Antwort erhalten hat, wird davon ausgegangen, dass die Abfrage ausgeführt wird.

Zugriff auf die Amazon S3 S3-Exportfunktion von Gremlin gewähren

Erforderliche IAM-Richtlinien

1. Neptune Query Lesezugriff

```
{
  "Sid": "NeptuneQueryRead",
  "Effect": "Allow",
  "Action": ["neptune-db:Read*"],
  "Resource": "arn:aws:neptune-db:us-east-1:123456789012:cluster-ABCD12/*"
}
```

Warum sie benötigt wird: Diese Berechtigung ermöglicht das Lesen von Daten aus Neptune-Datenbanken, was für die Ausführung der zu exportierenden Gremlin-Abfragen erforderlich ist. Das vorherige Beispiel erlaubt Leseabfragen. Für eine read/write Abfrage sind write/delete Berechtigungen erforderlich.

2. Amazon S3 S3-Exportberechtigungen

```
{
  "Sid": "NeptuneS3Export",
  "Effect": "Allow",
  "Action": [
    "s3:ListBucket",
    "s3:PutObject",
    "s3:AbortMultipartUpload",
    "s3:GetBucketPublicAccessBlock"
  ],
  "Resource": "arn:aws:s3:::neptune-export-bucket/*"
}
```

Warum jede Genehmigung benötigt wird:

- `s3:ListBucket`: Erforderlich, um das Vorhandensein eines Buckets und den Inhalt der Liste zu überprüfen.
- `s3:PutObject`: Erforderlich, um die exportierten Daten in Amazon S3 zu schreiben.
- `s3:AbortMultipartUpload`: Erforderlich, um unvollständige mehrteilige Uploads zu bereinigen, falls der Export fehlschlägt.
- `s3:GetBucketPublicAccessBlock`: Aus Sicherheitsgründen erforderlich, um vor dem Exportieren von Daten zu überprüfen, ob der Bucket nicht öffentlich ist.

3. AWS KMSBerechtigungen — optional. Nur erforderlich, wenn eine benutzerdefinierte AWS KMS Verschlüsselung verwendet wird.

```
{
```

```
  "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"
      ]
    }
  }
}
```

Warum jede Genehmigung benötigt wird:

- `kms:Decrypt`: Erforderlich, um den AWS KMS Schlüssel für die Datenverschlüsselung zu entschlüsseln.
- `kms:GenerateDataKey`: Erforderlich, um Datenschlüssel für die Verschlüsselung der exportierten Daten zu generieren.
- `kms:DescribeKey`: Erforderlich, um Informationen über den AWS KMS Schlüssel zu überprüfen und abzurufen.
- `kms:ViaService`: Erhöht die Sicherheit, indem erzwungen wird, dass der Schlüssel von dieser Rolle für keinen anderen AWS Dienst verwendet werden kann.

Wichtige Voraussetzungen

- IAM-Authentifizierung: Muss auf dem Neptune-Cluster aktiviert sein, um diese Berechtigungen durchzusetzen.
- VPC-Endpunkt:
  - Ein VPC-Endpunkt vom Typ Gateway für Amazon S3 ist erforderlich, damit Neptune mit Amazon S3 kommunizieren kann.
  - Um eine benutzerdefinierte AWS KMS Verschlüsselung in der Abfrage zu verwenden, ist ein VPC-Endpunkt vom Typ Interface für AWS KMS erforderlich, mit dem Neptune kommunizieren kann. AWS KMS

- Amazon S3 S3-Bucket-Konfiguration:
  - Darf nicht öffentlich sein.
  - Sollte eine Lebenszyklusregel haben, um unvollständige mehrteilige Uploads zu bereinigen.
  - Verschlüsselt automatisch neue Objekte.

Diese Berechtigungen und Voraussetzungen gewährleisten einen sicheren und zuverlässigen Export von Gremlin-Abfrageergebnissen und gewährleisten gleichzeitig die Einhaltung angemessener Zugriffskontrollen und Datenschutzmaßnahmen.

Verwalten Ihrer Amazon-Neptune-Datenbank

In diesem Abschnitt wird gezeigt, wie Sie Ihren Neptune-DB-Cluster mit dem AWS-Managementkonsole und dem verwalten und warten. AWS CLI

Neptune wird auf Clustern von Datenbankservern ausgeführt, die mittels Replikationstopologie verbunden sind. Daher beinhaltet die Verwaltung von Neptune häufig das Bereitstellen von Änderungen an mehreren Servern und das Sicherstellen, dass alle Neptune-Replicas beim primären Server aufrechterhalten werden.

Da Neptune den zugrunde liegenden Speicher bei wachsender Datenmenge transparent skaliert, erfordert die Verwaltung von Neptune relativ wenig Verwaltung von Festplattenspeicher. Da Neptune automatisch fortlaufend Sicherungen durchführt, erfordert ein Neptune-Cluster ebenfalls keine umfassende Planung und keine Ausfallzeiten für die Durchführung von Sicherungen.

Themen

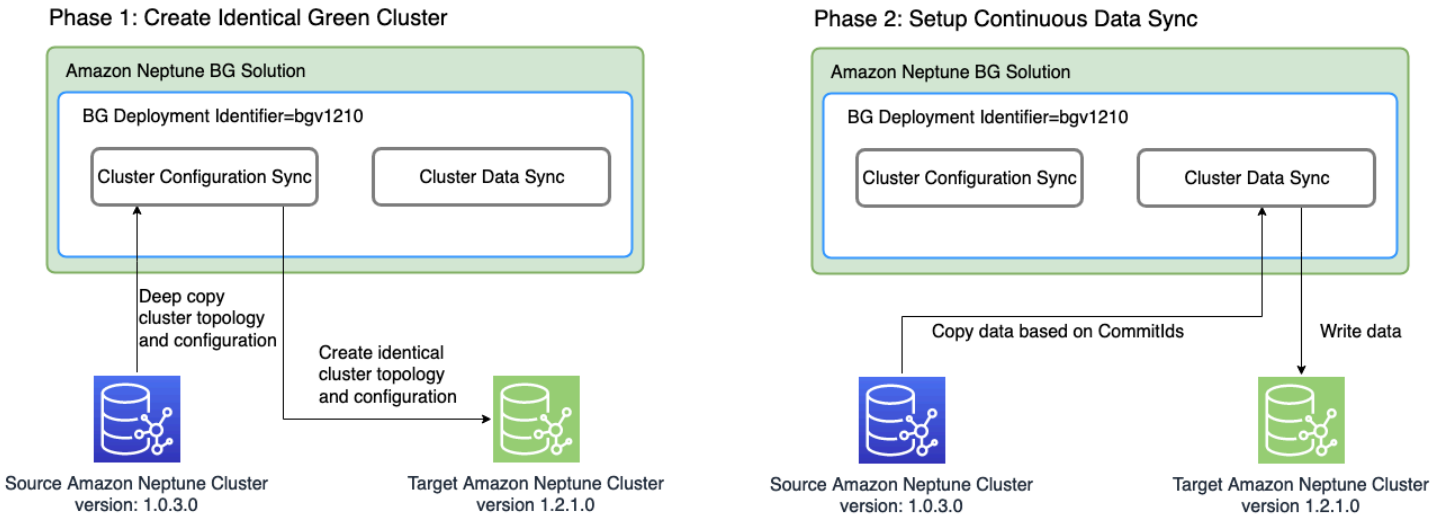
- Verwendung der Blue/Green Neptune-Lösung zur Durchführung von blaugrünen Updates
- Erstellen eines IAM-Benutzers mit Berechtigungen für Neptune
- Amazon-Neptune-Parametergruppen
- Amazon-Neptune-Parameter
- Starten eines Neptune-DB-Clusters mit dem AWS-Managementkonsole
- Stoppen und Starten eines DB-Clusters in Amazon Neptune
- Leeren eines Amazon-Neptune-DB-Clusters über die Fast-Reset-API
- Hinzufügen von Neptune-Reader-Instances zu einem DB-Cluster
- Erstellen einer Neptune-Reader-Instance über die Konsole
- Ändern eines Neptune-DB-Clusters über die Konsole
- Leistung und Skalierung in Amazon Neptune
- Auto Scaling der Anzahl der Replikate in einem Amazon Neptune-DB-Cluster
- Warten eines Amazon-Neptune-DB-Clusters
- Verwenden Sie eine CloudFormation Vorlage, um die Engine-Version Ihres Neptune DB-Clusters zu aktualisieren
- Klonen von Datenbanken in Neptune
- Verwalten von Amazon-Neptune-Instances

Verwendung der Blue/Green Neptune-Lösung zur Durchführung von blaugrünen Updates

Amazon-Neptune-Engine-Upgrades können Anwendungsausfallzeiten erfordern, da die Datenbank nicht verfügbar ist, während Updates installiert und verifiziert werden. Dies gilt unabhängig davon, ob die Updates manuell oder automatisch initiiert werden.

Neptune bietet eine Blue/Green Bereitstellungslösung, die Sie mithilfe eines CloudFormation Stacks ausführen können und die solche Ausfallzeiten erheblich reduziert. Die Lösung erstellt eine grüne Staging-Umgebung, die mit Ihrer blauen Produktionsumgebung synchronisiert ist. Anschließend können Sie diese Staging-Umgebung mit einem kleineren oder größeren Upgrade der Engine-Version, einer Änderung des Diagrammdatenmodells oder einer Aktualisierung des Betriebssystems zu aktualisieren und das Ergebnis zu testen. Anschließend können Sie sie schnell und mit sehr geringen Ausfallzeiten zu Ihrer Produktionsumgebung machen.

Die Blue/Green Neptun-Lösung durchläuft zwei Phasen, wie in diesem Diagramm dargestellt:



Phase 1 erstellt einen grünen, mit Ihrem Produktionscluster identischen DB-Cluster

Die Lösung erstellt einen DB-Cluster mit einer eindeutigen blue/green Bereitstellungs-ID und derselben Clustertopologie wie Ihr Produktionscluster. Das bedeutet, dass dieser Cluster dieselbe Zahl von DB-Instances mit denselben Größen, dieselben Parametergruppen und dieselben Konfigurationen wie der (blaue) Produktions-DB-Cluster besitzt. Der Unterschied besteht darin, dass er auf eine Engine-Zielversion aktualisiert wurde, die höher als die aktuelle (blaue) Engine-Version sein muss. Sie können eine Neben- und eine Haupt-Engine-Version für das Ziel angeben.

Wenn notwendig, führt die Lösung alle nötigen Zwischenaktualisierungen durch, um die angegebene Engine-Zielversion zu erreichen. Dieser neue Cluster bildet die grüne Staging-Umgebung.

Phase 2 richtet eine kontinuierliche Datensynchronisierung ein

Nach der vollständigen Vorbereitung der grünen Umgebung richtet die Lösung mithilfe von Neptune-Streams eine kontinuierliche Replikation zwischen dem Quell-Cluster (blau) und dem Ziel-Cluster (grün) ein. Wenn der Replikationsunterschied zwischen ihnen null erreicht, ist die Staging-Umgebung bereit für Tests. An diesem Punkt müssen Sie Schreibvorgänge zum blauen Cluster anhalten, um weitere Verzögerungen bei der Replikation zu vermeiden.

Ihre Engine-Zielversion kann neue Features oder Abhängigkeiten besitzen, die sich auf Ihre Anwendungen auswirken. Auf den Seiten für die Engine-Zielversion und die dazwischenliegenden Engine-Versionen unter Engine-Versionen finden Sie Informationen zu den Änderungen seit Ihrer aktuellen Engine-Version. Sie sollten Integrationstests durchführen oder Ihre Anwendungen manuell im grünen Cluster verifizieren, bevor Sie ihn zur Produktionsumgebung heraufstufen.

Nach dem Testen und Qualifizieren der Änderungen im grünen Cluster wechseln Sie einfach den Datenbank-Endpunkt in Ihren Anwendungen vom blauen zum grünen Cluster.

Nach der Umstellung löscht die Blue/Green Neptune-Lösung die alte blaue Produktionsumgebung nicht. Sie haben weiterhin Zugriff auf sie, um weitere Validierungen und Tests durchführen zu können, wenn notwendig. Für ihre Instances werden Ihnen bis zur Löschung die Standardgebühren berechnet. Die Blue/Green Lösung nutzt auch andere AWS Dienste, deren Kosten zu normalen Preisen abgerechnet werden. Einzelheiten zum Löschen der Lösung, wenn Sie diese nicht mehr benötigen, finden Sie im Abschnitt Bereinigen.

Voraussetzungen für den Betrieb des Neptune-Stacks Blue/Green

Bevor Sie den Blue/Green Neptune-Stack starten:

- Sie müssen in Ihrem (blauen) Produktions-Cluster Neptune-Streams aktivieren.
- Alle Instances in Ihrem blauen Cluster müssen den Status Verfügbar haben. Sie können den Status der Instances in der Neptune-Konsole oder mittels der API describe-db-instances überprüfen.
- Alle Instances müssen außerdem mit der DB-Cluster-Parametergruppe synchronisiert sein.
- Die Blue/Green Neptune-Lösung erfordert einen DynamoDB-VPC-Endpunkt in der VPC, in der sich Ihr Blue-Cluster befindet. Siehe Verwenden von Amazon-VPC-Endpunkten für den Zugriff auf DynamoDB.

- Wählen Sie einen Zeitraum für die Ausführung der Lösung, in dem der Schreib-Workload für den blauen Produktions-DB-Cluster so gering wie möglich ist. Sie sollten die Lösung möglichst nicht ausführen, während ein Massenladevorgang stattfindet oder wenn es aus einem anderen Grund wahrscheinlich eine große Zahl von Schreiboperationen geben wird.

Verwenden eines CloudFormation Vorlage zum Ausführen der Blue/Green Neptune-Lösung

Sie können verwenden AWS CloudFormation , um die Blue/Green Neptune-Lösung bereitzustellen. Die CloudFormation Vorlage erstellt eine Amazon EC2 EC2-Instance in derselben VPC wie Ihre Blue-Source-Neptune-Datenbank, installiert die Lösung dort und führt sie aus. Sie können den Fortschritt in CloudWatch Protokollen überwachen, wie unter Fortschritt überwachen beschrieben.

Sie können diese Links verwenden, um die Lösungsvorlage zu überprüfen, oder auf die Schaltfläche Stack starten klicken, um sie in der CloudFormation Konsole zu starten:

Anzeigen	In Designer anzeigen	

Wählen Sie in der Konsole aus der Dropdownliste oben rechts im Fenster die AWS Region aus, in der Sie die Lösung ausführen möchten.

Legen Sie die Stack-Parameter wie folgt fest:

- **`DeploymentID`**— Eine Kennung, die für jeden Neptun-Einsatz Blue/Green einzigartig ist.

  Sie wird als ID des grünen DB-Clusters und als Präfix bei der Benennung neuer Ressourcen verwendet, die während der Bereitstellung erstellt wurden.
- **`NeptuneSourceClusterId`** – Die ID des blauen DB-Clusters, den Sie aktualisieren möchten.
- **`NeptuneTargetClusterVersion:`** – Die Neptune-Engine-Version, zu der Sie den blauen DB-Cluster aktualisieren möchten.

  Diese Version muss höher als die aktuelle Engine-Version des blauen DB-Clusters sein.
- **`DeploymentMode`** – Gibt an, ob es sich um eine neue Bereitstellung handelt oder um den Versuch, eine frühere Bereitstellung fortzusetzen. Wenn Sie eine `DeploymentID` verwenden, die mit der ID einer vorherigen Bereitstellung identisch ist, legen Sie `DeploymentMode` auf `resume` fest.

Gültige Werte sind: `new` (Standardwert) und `resume`.

- **`GraphQueryType`** – Der Diagrammdatentyp für Ihre Datenbank.

  Gültige Werte sind: `propertygraph` (Standardwert) und `rdf`.

- **`SubnetId`** – Eine Subnetz-ID in derselben VPC, in der sich auch Ihr blaues DB-Cluster befindet. (Siehe Herstellen einer Verbindung zu einem Neptune-DB-Cluster von einer Amazon-EC2-Instance in derselben VPC).

  Stellen Sie die ID eines öffentlichen Subnetzes bereit, wenn Sie eine SSH-Verbindung zur Instance über EC2 Connect herstellen möchten.

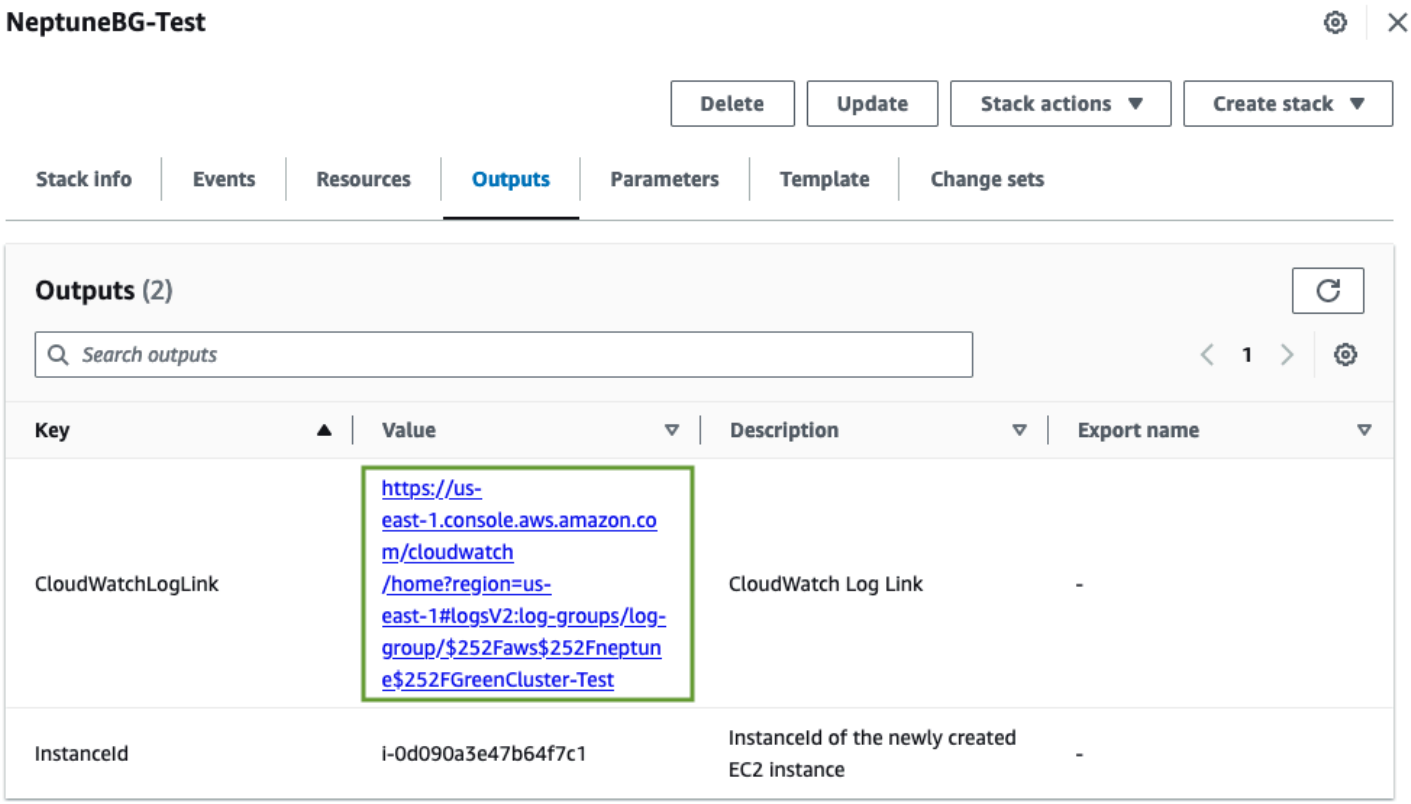
- **`InstanceSecurityGroup`** – Eine Sicherheitsgruppe für Ihre Amazon-EC2-Instance.

  Die Sicherheitsgruppe muss Zugriff auf den blauen DB-Cluster haben und Sie müssen eine SSH-Verbindung zur Instance herstellen können. Siehe Eine Sicherheitsgruppe über die Konsole erstellen.

Warten Sie, bis der Stack vollständig ist. Anschließend wird die Lösung sofort gestartet. Anschließend können Sie den Bereitstellungsprozess mithilfe von CloudWatch Protokollen überwachen, wie im nächsten Abschnitt beschrieben.

Überwachung des Fortschritts eines Neptun-Einsatzes Blue/Green

Sie können den Fortschritt der Blue/Green Neptune-Lösung überwachen, indem Sie zur CloudWatch Konsole gehen und sich die Protokolle in der `/aws/neptune/`*`(Neptune Blue/Green deployment ID)`* CloudWatch Protokollgruppe ansehen. In den Ausgaben des Lösungsstapels finden Sie einen Link zu den CloudWatch Protokollen: CloudFormation



Wenn Sie ein öffentliches Subnetz als Stack-Parameter angegeben haben, können Sie auch eine SSH-Verbindung zu der Amazon-EC2-Instance herstellen, die als Teil des Stacks erstellt wurde, und auf das Protokoll in `/var/log/cloud-init-output.log` verweisen.

Das Protokoll zeigt die Aktionen der Blue/Green Neptune-Lösung, wie in diesem Screenshot gezeigt:

```
============================================================

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':
```

Die Protokollmeldungen zeigen den Synchronisierungsstatus zwischen den blauen und grünen Clustern an:

```
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'}}
```

Der Synchronisierungsprozess überprüft die Replikationsverzögerung, indem er die Differenz zwischen dem neuesten Stream `eventID` auf dem blauen Cluster und dem Replikationsprüfpunkt berechnet, der in der vom Replikationsstapel erstellten DynamoDB-Checkpoint-Tabelle vorhanden ist. Neptune-to-Neptune Anhand dieser Meldungen können Sie die aktuelle Replikationsdifferenz überwachen.

Wechsel vom blauen Produktions-Cluster zum aktualisierten grünen Cluster

Bevor Sie den grünen Cluster zur Produktion heraufstufen, müssen Sie sicherstellen, dass die Commit-Differenz zwischen dem blauen und dem grünen Cluster null ist, und den gesamten Schreibverkehr zum blauen Cluster deaktivieren. Wenn Sie weiter zum blauen Cluster schreiben, während Sie den Datenbankendpunkt zum grünen Cluster wechseln, kann dies zu beschädigten Daten führen, da in beiden Clustern unvollständige Daten geschrieben werden. Möglicherweise müssen Sie den Leseverkehr noch nicht deaktivieren.

Wenn Sie die IAM-Authentifizierung für den (blauen) Quell-Cluster aktiviert haben, müssen Sie alle in Ihren Anwendungen verwendeten IAM-Richtlinien so aktualisieren, dass sie auf den grünen Cluster verweisen. (Ein Beispiel für eine solche Richtlinie finden Sie in dieser Richtlinie für uneingeschränkten Zugriff).

Warten Sie nach dem Deaktivieren des Schreibverkehrs, bis die Replikation abgeschlossen ist, und aktivieren Sie dann den Schreibverkehr für den grünen Cluster (nicht für den blauen Cluster). Wechseln Sie auch den Leseverkehr vom blauen zum grünen Cluster.

Aufräumen, nachdem die Blue/Green Neptun-Lösung abgeschlossen ist

Nachdem Sie den Staging-Cluster (grün) zur Produktion hochgestuft haben, bereinigen Sie die von der Blue/Green Neptune-Lösung erstellten Ressourcen:

- Löschen Sie die Amazon-EC2-Instance, die zur Ausführung der Lösung erstellt wurde.
- Löschen Sie die CloudFormation Vorlagen für die auf Neptune Streams basierende Replikation, die den grünen Cluster mit dem blauen Cluster synchron hielt. Eine Vorlage hat den zuvor angegebenen Stack-Namen und die andere Vorlage besteht aus der Bereitstellungs-ID gefolgt von „-replication“: *(DeploymentID)*`-replication`.

Durch das Löschen von CloudFormation Vorlagen werden die Cluster selbst nicht gelöscht. Sobald Sie sich vergewissert haben, dass der grüne Cluster wie erwartet funktioniert, können Sie optional einen Snapshot erstellen, bevor Sie den blauen Cluster manuell löschen.

Bewährte Methoden für Blue/Green Neptune-Lösungen

- Bevor Sie den grünen Cluster zur Produktion wechseln, sollten Sie sorgfältig überprüfen, ob er ordnungsgemäß funktioniert. Prüfen Sie die Konsistenz der Daten und die Konfiguration der Datenbank. Es ist möglich, dass einige neue Engine-Versionen auch Client-Upgrades erfordern. Prüfen Sie vor dem Upgrade die Engine-Versionshinweise. Es lohnt sich, all dies in Entwicklungs-, Test- und Vorproduktionsumgebungen zu testen, bevor Sie mit einem blue/green Upgrade in der Produktion beginnen.
- Der Wechsel vom blauen zum grünen Server sollte bevorzugt während eines Wartungsfensters erfolgen.
- Um sicherzustellen, dass nach Upgrade und Synchronisation alles ordnungsgemäß funktioniert, sollten Sie den ursprünglichen Cluster für einige Zeit beibehalten, bevor Sie ihn löschen. Dies könnte sich als nützlich erweisen, wenn ein unvorhergesehenes Problem auftritt.
- Vermeiden Sie umfangreiche Schreibvorgänge wie Massenladevorgänge, wenn Sie die Blue/Green Neptune-Lösung ausführen, da diese zu Verzögerungen bei der Replikation führen können, was zu erheblichen Ausfallzeiten führt. Im Idealfall liegt zwischen der Deaktivierung von Schreiboperationen im blauen Cluster und ihrer Aktivierung im grünen Cluster nur ein kurzer Moment.

Fehlerbehebung bei der Neptune-Lösung Blue/Green

In den folgenden Informationen werden Probleme hervorgehoben, die während des Blue/Green Lösungsbereitstellungsprozesses auftreten können, z. B. Konflikte mit vorhandenen Clustern, die Notwendigkeit, Neptune-Streams zu aktivieren, laufende Massenladevorgänge und Anforderungen an die Versionskompatibilität. Durch die Behebung dieser potenziellen Probleme können Sie eine reibungslose und erfolgreiche Implementierung der Blue/Green Neptune-Lösung sicherstellen.

Durch die Neptun-Lösung Blue/Green hervorgerufene Fehler

- **`Cluster with id = (blue_green_deployment_id) already exists`**— Es gibt einen vorhandenen Cluster mit Kennung. *`(blue_green_deployment_id)`*

  Geben Sie eine neue Bereitstellungs-ID ein oder setzen Sie den Bereitstellungsmodus auf, `resume` wenn der Cluster in einem früheren Blue/Green Neptune-Lauf erstellt wurde.
- **`Streams should be enabled on the source Cluster for Blue Green Deployment`**
  – Aktivieren Sie Neptune-Streams für den blauen (Quell-)Cluster.
- **`No Bulkload should be in progress on source cluster: (cluster_id)`**— Die Blue/Green Neptune-Lösung wird beendet, wenn sie eine laufende Massenladung feststellt.

  Dies soll sicherstellen, dass der Synchronisierungsprozess mit den Schreiboperationen Schritt halten kann. Vermeiden oder brechen Sie alle laufenden Massenladeaufträge ab, bevor Sie die Neptune-Lösung Blue/Green starten.
- **`Blue Green deployment requires instances to be in sync with db cluster parameter group`** – Alle Änderungen an der Cluster-Parametergruppe sollten im gesamten DB-Cluster synchronisiert werden. Siehe Amazon-Neptune-Parametergruppen.
- **`Invalid target engine version for Blue Green Deployment`** – Die Engine-Zielversion muss in Engine-Versionen für Amazon Neptune als aktiv aufgelistet sein und höher als die aktuelle Engine-Version des (blauen) Quell-Clusters sein.

Erstellen eines IAM-Benutzers mit Berechtigungen für Neptune

Um zur Erstellung und Verwaltung eines Neptune-DB-Clusters auf die Neptune-Konsole zuzugreifen, müssen Sie einen IAM-Benutzer mit allen notwendigen Berechtigungen erstellen.

Im ersten Schritt erstellen Sie eine serviceverknüpfte Rollenrichtlinie für Neptune:

Erstellen einer serviceverknüpften Rollenrichtlinie für Amazon Neptune

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie im Navigationsbereich auf der linken Seite Policies (Richtlinien).
3. Wählen Sie auf der Seite Richtlinien die Option Richtlinie erstellen aus.
4. Wählen Sie auf der Seite Richtlinie erstellen die Registerkarte JSON aus und kopieren Sie die folgende serviceverknüpfte Rollenrichtlinie:

   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. Wählen Sie Weiter: Tags und auf der Seite Tags hinzufügen die Option Weiter: Überprüfen aus.
6. Geben Sie auf der Seite „Richtlinie überprüfen“ der neuen Richtlinie den Namen "NeptuneServiceLinked“.

Weitere Informationen zu serviceverknüpften Rollen finden Sie unter Verwenden von serviceverknüpften Rollen für Amazon Neptune.

Erstellen eines neuen IAM-Benutzers mit allen notwendigen Berechtigungen

Als Nächstes erstellen Sie den neuen IAM-Benutzer und fügen diesem die verwalteten Richtlinien an, die die benötigten Berechtigungen gewähren, sowie die von Ihnen erstellte serviceverknüpfte Rollenrichtlinie (`NeptuneServiceLinked`).

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie im Navigationsbereich links Benutzer und dann Benutzer hinzufügen aus.
3. Geben Sie auf der Seite „Benutzer hinzufügen“ einen Namen für den neuen IAM-Benutzer ein, wählen Sie als AWS Anmeldeinformationstyp die Option Access key — Programatic access aus und wählen Sie Next: Permissions aus.
4. Geben Sie auf der Seite Berechtigungen festlegen in das Feld Filterrichtlinien „Neptune“ ein. Wählen Sie nun aus den aufgelisteten Richtlinien Folgendes aus:

   - NeptuneFullAccess
   - NeptuneConsoleFullAccess
   - NeptuneServiceLinked(vorausgesetzt, Sie haben die zuvor erstellte servicebezogene Rollenrichtlinie so genannt).
5. Geben Sie als Nächstes „VPC“ anstelle von „Neptune“ in das Feld Filterrichtlinien ein. Wählen Sie Amazon VPCFull Access aus den aufgelisteten Richtlinien aus.
6. Wählen Sie Weiter: Tags und auf der Seite Tags hinzufügen die Option Weiter: Überprüfen aus.
7. Prüfen Sie auf der Seite Überprüfen, ob die folgenden Richtlinien jetzt mit Ihrem neuen Benutzer verknüpft sind:

   - NeptuneFullAccess
   - NeptuneConsoleFullAccess
   - NeptuneServiceLinked
   - VPCFullZugriff auf Amazon

   Wählen Sie anschließend Benutzer erstellen aus.

8. Laden Sie zum Schluss die Zugriffsschlüssel-ID und den geheimen Zugriffsschlüssel für den neuen Benutzer herunter und speichern Sie diese.

Für die Zusammenarbeit mit anderen -Diensten wie Amazon Simple Storage Service (Amazon S3) müssen Sie weitere Berechtigungen und Vertrauensbeziehungen hinzufügen.

Amazon-Neptune-Parametergruppen

Sie können Ihre Datenbankkonfiguration in Amazon Neptune mittels Parametern in einer Parametergruppe verwalten. Parametergruppen dienen als Container für Engine-Konfigurationswerte, die auf eine oder mehrere DB-Instances angewendet werden.

Es gibt zwei Arten von Parametergruppen: DB-Cluster-Parametergruppen und DB-Parametergruppen.

- DB-Parametergruppen werden auf Instance-Ebene angewendet und sind im Allgemeinen mit Einstellungen für die Neptune-Diagramm-Engine verknüpft, z. B. dem Parameter `neptune_query_timeout`.
- DB-Cluster-Parametergruppen gelten für alle Instances im Cluster und weisen weitergefasste Einstellungen auf. Jeder Neptune-Cluster ist einer DB-Cluster-Parametergruppe zugeordnet. Jede DB-Instance in diesem Cluster übernimmt die Engine-Konfigurationswerte aus der DB-Cluster-Parametergruppe.

Alle Konfigurationswerte, die Sie in der DB-Cluster-Parametergruppe ändern, überschreiben die Standardwerte in der DB-Parametergruppe. Wenn Sie die entsprechenden Werte in der DB-Parametergruppe bearbeiten, überschreiben diese Werte die Einstellungen in der DB-Cluster-Parametergruppe.

Wenn Sie eine DB-Instance ohne Angabe einer benutzerdefinierten DB-Parametergruppe erstellen, wird eine Standard-DB-Parametergruppe verwendet. Sie können die Parametereinstellungen der Standard-DB-Parametergruppe nicht ändern. Um die Standard-Parametereinstellungen zu ändern, müssen Sie stattdessen eine neue DB-Parametergruppe erstellen. In einer von Ihnen erstellten DB-Parametergruppe können nicht alle DB-Engine-Parameter geändert werden.

Parametergruppen werden in Familien erstellt, die mit bestimmten Neptune-Engine-Versionen kompatibel sind. Wenn Sie auf eine neue Haupt- oder Nebenversion der Engine aktualisieren, müssen Sie möglicherweise Ihre benutzerdefinierten Parametergruppen mit der entsprechenden Parametergruppenfamilie für diese Version neu erstellen.

Die Benennung der Parametergruppenfamilie folgt dem Muster`neptune`X.Y, das der Engine-Version X.Y entspricht. Beispiel:

- `neptune1`— für Engine-Versionen vor 1.2.0.0
- `neptune1.2`— für Engine-Versionen 1.2.x

- `neptune1.3`— für Engine-Versionen 1.3.x
- `neptune1.4`— für Engine-Versionen 1.4.x

Wenn Sie Ihren Neptune-Cluster aktualisieren, überprüfen Sie die Versionshinweise für Ihre Ziel-Engine-Version, um festzustellen, ob eine neue Parametergruppenfamilie erforderlich ist. In diesem Fall müssen Sie vor dem Upgrade alle benutzerdefinierten Parametergruppen in der neuen Familie neu erstellen.

Einige Neptune-Parameter sind statisch, andere sind dynamisch. Die Unterschiede sind wie folgt:

Statische Parameter

- Ein statischer Parameter ist ein Parameter, der erst wirksam wird, wenn eine DB-Instance neu gestartet wird. Wenn Sie einen statischen Parameter ändern und die Instance-DB-Parametergruppe speichern, müssen Sie daher die DB-Instance manuell neu starten, damit die Parameteränderung wirksam wird. Zurzeit sind alle Neptune-Parameter auf Instance-Ebene statisch (in DB-Parametergruppen, nicht in DB-Cluster-Parametergruppen).
- Wenn Sie einen statischen Parameter auf Cluster-Ebene ändern und die DB-Cluster-Parametergruppe speichern, wird die Änderung des Parameters nach dem manuellen Neustart aller DB-Instances im Cluster wirksam.

Dynamische Parameter

- Ein dynamischer Parameter ist ein Parameter, der beinahe direkt nach der Aktualisierung des Parameters in der Parametergruppe wirksam wird. Sie müssen daher eine DB-Instance nach der Aktualisierung eines dynamischen Parameters nicht neu starten, damit die Parameteränderung wirksam wird.
- Die Änderung eines dynamischen Cluster-Parameters wird mit einer geringfügigen Verzögerung auf alle DB-Instances angewendet.
- Ein aktualisierter dynamischer Parameterwert wird nicht auf die aktuell ausgeführten Anforderungen angewendet, sondern nur auf Anforderungen, die nach der Änderung eingereicht wurden.
- Wenn Sie einen dynamischen Parameter auf Cluster-Ebene ändern, wird die Parameteränderung standardmäßig sofort auf den DB-Cluster angewendet, ohne dass ein Neustart erforderlich ist. Um die Parameteränderung zu verschieben, bis die DB-Instances im Cluster neu gestartet wurden,

können Sie mit dem den Wert AWS CLI auf `pending-reboot` für die Parameteränderung setzen. `ApplyMethod`

Zurzeit sind alle Parameter statisch, abgesehen von den folgenden neuen Cluster-Parametern:

- `neptune_enable_slow_query_log` (Cluster-Ebene)
- `neptune_slow_query_log_threshold` (Cluster-Ebene)

Hier sind einige wichtige Punkte, die Sie über die Arbeit mit Parametern in einer DB-Parametergruppe kennen sollten:

- Werden die Parameter in einer DB-Parametergruppe unpassend eingestellt, kann dies unbeabsichtigte unerwünschte Auswirkungen haben, einschließlich verminderter Leistung und Systeminstabilität. Gehen Sie immer mit Bedacht vor, wenn Sie Datenbankparameter ändern, und sichern Sie Ihre Daten, bevor Sie eine DB-Parametergruppe ändern. Testen Sie Änderungen an Ihren Parametergruppeneinstellungen zunächst auf einer Test-DB-Instance, ehe Sie die Änderungen für eine Produktions-DB-Instance übernehmen.
- Wenn Sie die DB-Parametergruppe ändern, die einer DB-Instance zugeteilt sind, müssen Sie eine Instance manuell neustarten, bevor Sie die neue DB-Parametergruppe von der DB-Instance verwendet werden kann.

> **Note**
>
> Vor Release: 1.2.0.0 (21.07.2022) wurden bei einem Neustart der primären Instance (Writer-Instance) alle Read-Replicas in einem DB-Cluster automatisch neu gestartet. Ab Release: 1.2.0.0 (21.07.2022) führt ein Neustart der primären Instance nicht mehr zum Neustart der Read–Replica-Instances. Wenn Sie einen Parameter auf Cluster-Ebene ändern, müssen Sie daher jede Instance getrennt neu starten, um die Parameteränderung zu übernehmen.

Bearbeiten einer DB-Cluster-Parametergruppe oder DB-Parametergruppe

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die Amazon Neptune Neptune-Konsole zu Hausehttps://console.aws.amazon.com/neptune/.
2. Wählen Sie im Navigationsbereich Parameter groups (Parametergruppen) aus.

3. Wählen Sie den Name-Link für die DB-Parametergruppe aus, die Sie bearbeiten möchten.

   (Optional) Wählen Sie Create Parameter Group (Parametergruppe erstellen) aus, um eine neue Cluster-Parametergruppe zu erstellen und legen Sie die neue Gruppe an. Wählen Sie dann den Name für die Parametergruppe aus.

   > **Important**
   >
   > Dieser Schritt ist erforderlich, wenn Sie nur die Standard-DB-Cluster-Parametergruppe haben, da diese nicht geändert werden kann.

4. Suchen Sie nach dem Parameter und klicken Sie auf das Feld Wert neben der Spalte Name.
5. Geben Sie den zulässigen Wert ein und wählen Sie das Häkchen neben dem Wertfeld aus.
6. Wählen Sie Änderungen speichern aus.
7. Starten Sie jede DB-Instance im Neptune-Cluster neu, wenn Sie einen DB-Cluster-Parameter ändern, oder eine oder mehrere spezifische Instances, wenn Sie einen DB-Instance-Parameter ändern.

Erstellen einer DB-Parametergruppe oder DB-Cluster-Parametergruppe

Sie können die Neptune-Konsole verwenden, um auf einfache Weise eine neue Parametergruppe zu erstellen:

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die Amazon Neptune Neptune-Konsole zu Hausehttps://console.aws.amazon.com/neptune/.
2. Klicken Sie im linken Navigationsbereich auf Parameter groups (Parametergruppen).
3. Wählen Sie Create DB parameter Group (DB-Parametergruppe erstellen) aus.

   Die Seite Create DB parameter group (DB-Parametergruppe erstellen) wird angezeigt.
4. Wählen Sie in der Liste der Parametergruppen die Familie aus, die Ihrer Zielversion der Neptune-Engine entspricht (z. B. Neptun1.2, Neptun1.3 oder Neptune1.4).
5. Wählen Sie in der Liste Typ entweder DB-Parametergruppe oder DB-Cluster-Parametergruppe) aus.
6. Geben Sie im Feld Gruppenname den Namen der neuen DB-Parametergruppe ein.
7. Geben Sie im Feld Beschreibung eine Beschreibung für die neue DB-Parametergruppe ein.
8. Wählen Sie Erstellen aus.

Sie können eine neue Parametergruppe auch über die AWS CLI erstellen:

```
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)
```

Amazon-Neptune-Parameter

Sie können Ihre Datenbankkonfiguration in Amazon Neptune mittels Parametern in Parametergruppen verwalten. Die folgenden Parameter sind für die Konfiguration Ihrer Neptune-Datenbank verfügbar:

Parameter auf Cluster-Ebene

- 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

Parameter auf Instance-Ebene

- neptune_dfe_query_engine
- neptune_query_timeout
- neptune_result_cache
- UndoLogPurgeConfig

Veraltete Parameter

- neptune_enforce_ssl

neptune_enable_audit_log (Parameter auf Cluster-Ebene)

Dieser Parameter aktiviert/deaktiviert die Audit-Protokollierung für Neptune.

Zulässige Werte sind `0` (deaktiviert) und `1` (aktiviert). Der Standardwert ist `0`.

Da dieser Parameter statisch ist, werden Änderungen für Instances erst wirksam, wenn sie neu gestartet werden.

Sie können Audit-Logs auf Amazon veröffentlichen CloudWatch, wie unter beschriebenVerwenden der CLI zum Veröffentlichen von Neptune-Auditprotokollen in Logs CloudWatch.

neptune_enable_slow_query_log (Parameter auf Cluster-Ebene)

Mit diesem Parameter können Sie das Neptune-Feature für die Protokollierung langsamer Abfragen aktivieren oder deaktivieren.

Da dieser Parameter dynamisch ist, erfordert oder verursacht die Änderung des Werts keinen Neustart des DB-Clusters.

Die zulässigen Werte sind:

- **`info`** – Aktiviert die Protokollierung langsamer Abfragen und protokolliert ausgewählte Attribute, die zur Langsamkeit beitragen könnten.
- **`debug`** – Aktiviert die Protokollierung langsamer Abfragen und protokolliert alle verfügbaren Attribute der Abfrageausführung.
- **`disabled`** – Deaktiviert die Protokollierung langsamer Abfragen.

Der Standardwert ist `disabled`.

Sie können Logs für langsame Abfragen auf Amazon veröffentlichen CloudWatch, wie unter beschrieben. Verwenden der CLI zum Veröffentlichen von Neptune-Protokollen für langsame Abfragen in Logs CloudWatch

neptune_slow_query_log_threshold (Parameter auf Cluster-Ebene)

Dieser Parameter gibt den Schwellenwert für die Ausführungszeit in Millisekunden an, nach dessen Ablauf eine Abfrage als langsame Abfrage gilt. Wenn die Protokollierung langsamer Abfragen aktiviert ist, werden Abfragen, deren Ausführung diesen Schwellenwert überschreitet, zusammen mit einigen ihrer Attribute protokolliert.

Der Standardwert ist 5.000 Millisekunden (5 Sekunden).

Da dieser Parameter dynamisch ist, erfordert oder verursacht die Änderung des Werts keinen Neustart des DB-Clusters.

`neptune_lab_mode` (Parameter auf Cluster-Ebene)

Bei Festlegung aktiviert dieser Parameter bestimmte experimentelle Features von Neptune. Informationen zu den aktuell verfügbaren experimentellen Features finden Sie unter Neptune-Labor-Modus.

Da dieser Parameter statisch ist, werden Änderungen für Instances erst wirksam, wenn sie neu gestartet werden.

Labor-Modus-Features sind in der Regel standardmäßig deaktiviert. Um eine experimentelle Funktion zu aktivieren oder explizit zu deaktivieren, fügen Sie *`(feature name)`* `=enabled` oder *`(feature name)`* `=disabled` in diesen Parameter ein. Sie können mehrere Features aktivieren oder deaktivieren, indem Sie sie wie folgt durch Komma trennen:

`(feature #1 name)``=enabled,` *`(feature #2 name)`*`=enabled`

`neptune_query_timeout` (Parameter auf Cluster-Ebene)

Gibt eine bestimmte Timeout-Dauer für Diagrammabfragen in Millisekunden an.

Zulässige Werte liegen zwischen `10` und `2,147,483,647` (2^{31} - 1). Der Standardwert ist `120,000` (2 Minuten).

Da dieser Parameter statisch ist, werden Änderungen für Instances erst wirksam, wenn sie neu gestartet werden.

Wenn mehrere Timeout-Einstellungen konfiguriert sind (Clusterebene, Instanzebene und pro Abfrage), zeigt die folgende Tabelle, welcher Timeout-Wert Vorrang hat:

Cluster PG	Instanz PG	Abfragehinweis	Ergebnis
Standard	Standard	Keine	Cluster
Benutzerdefiniert	Standard	Keine	Cluster

Cluster PG	Instanz PG	Abfragehinweis	Ergebnis
Benutzerdefiniert	Benutzerdefiniert	Keine	Instance
Standard	Benutzerdefiniert	Keine	Instance
Beliebig	Beliebig	Am niedrigsten	Query
Standard	Benutzerdefiniert	nicht am niedrigsten	Instance
Benutzerdefiniert	Standard	nicht am niedrigsten	Cluster
Benutzerdefiniert	Benutzerdefiniert	nicht am niedrigsten	Instance

> **Note**
>
> Es können unerwartete Kosten entstehen, wenn Sie den Wert für den Abfrage-Timeout zu hoch festlegen, besonders bei Serverless-Instances. Ohne eine angemessene Timeout-Einstellung könnten Sie versehentlich eine Abfrage ausgeben, die sehr viel länger als erwartet ausgeführt wird. Dies kann zu Kosten führen, die Sie nie erwartet haben. Dies gilt insbesondere für eine Serverless-Instance, die während der Ausführung der Abfrage auf einen großen, teuren Instance-Typ hochskaliert werden könnte.
> Sie können mit einem Abfrage-Timeout, der für die meisten Ihrer Abfragen geeignet ist und nur bei einer ungewöhnlich langen Ausführung zu einem Timeout führt, unerwartete Ausgaben dieser Art vermeiden.

`neptune_streams` (Parameter auf Cluster-Ebene)

Aktiviert oder deaktiviert Neptune-Streams.

Da dieser Parameter statisch ist, werden Änderungen für Instances erst wirksam, wenn sie neu gestartet werden.

Zulässige Werte sind `0` (deaktiviert, Standardwert) und `1` (aktiviert).

`neptune_streams_expiry_days` (Parameter auf Cluster-Ebene)

Gibt die Anzahl der Tage an, bis der Server Stream-Datensätze löscht.

Gültige Werte sind 1 bis einschließlich 90. Der Standardwert ist 7.

Dieser Parameter wurde in Engine-Version 1.2.0.0 eingeführt.

Da dieser Parameter statisch ist, werden Änderungen für Instances erst wirksam, wenn sie neu gestartet werden.

`neptune_lookup_cache` (Parameter auf Cluster-Ebene)

Deaktiviert oder aktiviert den Neptune-Lookup-Cache auf R5d-Instances.

Da dieser Parameter statisch ist, werden Änderungen für Instances erst wirksam, wenn sie neu gestartet werden.

Zulässige Werte sind 1 (aktiviert) und 0 (deaktiviert). Der Standardwert ist 0. Wenn jedoch eine R5d-Instance im DB-Cluster erstellt wird, wird der Parameter `neptune_lookup_cache` automatisch auf 1 festgelegt und es wird ein Lookup-Cache für diese Instance erstellt.

`neptune_autoscaling_config` (Parameter auf Cluster-Ebene)

Legt die Konfigurationsparameter für die Read-Replica-Instances fest, die Neptune Auto Scaling erstellt und verwaltet.

Da dieser Parameter statisch ist, werden Änderungen für Instances erst wirksam, wenn sie neu gestartet werden.

Sie können mit einer JSON-Zeichenfolge, die Sie als Wert für den Parameter `neptune_autoscaling_config` festlegen, Folgendes angeben:

- Den Instance-Typ, den Neptune Auto Scaling für alle neuen Read-Replica-Instances verwendet, die es erstellt.
- Die Wartungsfenster, die diesen Read-Replicas zugewiesen sind.
- Die Tags, die allen neuen Read-Replicas zugeordnet werden sollen.

Die JSON-Zeichenfolge hat eine Struktur wie diese:

```
"{
  \"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\"
}"
```

Beachten Sie, dass alle Anführungszeichen in der Zeichenfolge mit einem Backslash-Zeichen (\) maskiert werden müssen.

Wenn eine dieser drei Konfigurationseinstellungen nicht im Parameter `neptune_autoscaling_config` angegeben ist, wird sie aus der Konfiguration der primären Writer-Instance des DB-Clusters kopiert.

`neptune_ml_iam_role` (Parameter auf Cluster-Ebene)

Gibt den in Neptune ML verwendeten IAM-Rollen-ARN an. Der Wert kann jeder gültige IAM-Rollen-ARN sein.

Da dieser Parameter statisch ist, werden Änderungen für Instances erst wirksam, wenn sie neu gestartet werden.

Sie können den standardmäßigen IAM-Rollen-ARN für Machine Learning in Diagrammen angeben.

`neptune_ml_endpoint` (Parameter auf Cluster-Ebene)

Gibt den für Neptune ML verwendeten Endpunkt an. Der Wert kann ein beliebiger gültiger SageMaker AI-Endpunktname sein.

Da dieser Parameter statisch ist, werden Änderungen für Instances erst wirksam, wenn sie neu gestartet werden.

Sie können den SageMaker Standard-KI-Endpunkt für maschinelles Lernen in Diagrammen angeben.

`neptune_enable_inline_server_generated_edge_id` (Parameter auf Cluster-Ebene)

Aktivieren oder deaktivieren Sie die vom Neptune-Inline-Server generierte Edge-ID-Funktion.

Da dieser Parameter statisch ist, werden Änderungen für Instances erst wirksam, wenn sie neu gestartet werden.

Zulässige Werte sind 1 (aktiviert) und `0` (deaktiviert). Der Standardwert ist `0`.

neptune_dfe_query_engine (Parameter auf Instance-Ebene)

Dieser DB-Instance-Parameter steuert, wie die DFE-Abfrage-Engine verwendet wird. Die zulässigen Werte sind:

Da dieser Parameter statisch ist, werden Änderungen für Instances erst wirksam, wenn sie neu gestartet werden.

- **`enabled`** – Bewirkt die Verwendung der DFE-Engine, wann immer möglich, es sei denn, der Abfragehinweis `useDFE` ist vorhanden und auf `false` festgelegt.
- **`viaQueryHint`** (Standardwert) – Bewirkt die Verwendung der DFE-Engine ausschließlich für Abfragen, die explizit den auf `true` festgelegten Abfragehinweis `useDFE` enthalten.

Wenn dieser Parameter nicht explizit festgelegt ist, wird der Standardwert (`viaQueryHint`) verwendet, wenn die Instance gestartet wird.

> **Note**
>
> Alle openCypher-Abfragen werden von der DFE-Engine ausgeführt, unabhängig von der Festlegung dieses Parameter.

neptune_query_timeout (Parameter auf Instance-Ebene)

Dieser DB-Instance-Parameter gibt eine Timeout-Dauer für Diagrammabfragen in Millisekunden für eine einzelne Instance an.

Da dieser Parameter statisch ist, werden Änderungen für Instances erst wirksam, wenn sie neu gestartet werden.

Zulässige Werte liegen zwischen `10` und `2,147,483,647` (2^{31} - 1). Der Standardwert ist `120,000` (2 Minuten).

> **Note**
>
> Es können unerwartete Kosten entstehen, wenn Sie den Wert für den Abfrage-Timeout zu hoch festlegen, besonders bei Serverless-Instances. Ohne eine angemessene Timeout-Einstellung könnten Sie versehentlich eine Abfrage ausgeben, die sehr viel länger als

erwartet ausgeführt wird. Dies kann zu Kosten führen, die Sie nie erwartet haben. Dies gilt insbesondere für eine Serverless-Instance, die während der Ausführung der Abfrage auf einen großen, teuren Instance-Typ hochskaliert werden könnte.
Sie können mit einem Abfrage-Timeout, der für die meisten Ihrer Abfragen geeignet ist und nur bei einer ungewöhnlich langen Ausführung zu einem Timeout führt, unerwartete Ausgaben dieser Art vermeiden.

neptune_result_cache (Parameter auf Instance-Ebene)

neptune_result_cache – Dieser DB-Instance-Parameter aktiviert oder deaktiviert Zwischenspeichern von Abfrageergebnissen.

Da dieser Parameter statisch ist, werden Änderungen für Instances erst wirksam, wenn sie neu gestartet werden.

Zulässige Werte sind `0` (deaktiviert, Standardwert) und `1` (aktiviert).

UndoLogPurgeConfig (Parameter auf Instance-Ebene)

UndoLogPurgeConfig— Verwenden Sie diesen Parameter, um aggressives UndoLog Purging in Neptune zu aktivieren oder zu deaktivieren.

Zulässige Werte sind`default`, was die Standardanzahl von Threads für das Löschen von Undo-Logs verwendet, und, das eine erhöhte Anzahl von Threads verwendet`aggressive`, um die Bereinigung von Undo-Logs zu beschleunigen. Wenn die `agressive` Option ausgewählt ist, können Sie davon ausgehen, dass Sie einen höheren Wert für die Metrik beobachten werden. `NumUndoPagesPurged`

neptune_enforce_ssl (VERALTETER Parameter auf ClusterEbene)

(Veraltet) Früher gab es Regionen, die HTTP-Verbindungen mit Neptune zuließen. Dieser Parameter wurde verwendet, um bei Festlegung auf 1 HTTPS für alle Verbindungen durchzusetzen. Dieser Parameter ist jedoch nicht mehr relevant, da Neptune jetzt in allen Regionen nur noch HTTPS-Verbindungen akzeptiert.

Starten eines Neptune-DB-Clusters mit dem AWS-Managementkonsole

Der einfachste Weg, einen neuen Neptune-DB-Cluster zu starten, besteht darin, eine CloudFormation Vorlage zu verwenden, die alle erforderlichen Ressourcen für Sie erstellt, wie unter erklärt. Neptun-Cluster erstellen

Sie können jedoch auch die Neptune-Konsole verwenden, um einen neuen DB-Cluster manuell zu starten, wie hier beschrieben.

> **Note**
>
> Bevor Sie auf die Neptune-Konsole zugreifen können, um einen Neptune-Cluster zu erstellen, benötigen Sie einen Benutzer mit ausreichenden Berechtigungen. Wenn Ihr aktueller Benutzer nicht über diese Berechtigungen verfügt, können Sie einen IAM-Benutzer mit den dafür erforderlichen Berechtigungen erstellen, wie unter erklärt. Erstellen eines IAM-Benutzers mit Berechtigungen für Neptune

Sobald Sie überprüft haben, ob Ihr Benutzer über die richtigen Berechtigungen verfügt, oder Sie einen Benutzer mit den richtigen Berechtigungen erstellt haben, melden Sie sich AWS-Managementkonsole als dieser IAM-Benutzer an und führen Sie die folgenden Schritte aus, um einen neuen DB-Cluster zu erstellen:

So starten Sie einen Neptune-DB-Cluster über die Konsole

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die Amazon Neptune Neptune-Konsole zu Hausehttps://console.aws.amazon.com/neptune/.
2. Navigieren Sie unter Datenbanken zur Seite Cluster und wählen Sie Datenbank erstellen. Daraufhin wird die Seite Datenbank erstellen geöffnet.
3. Geben Sie unter Einstellungen einen Namen für Ihren neuen DB-Cluster ein oder akzeptieren Sie den angegebenen Standardnamen. Dieser Name wird in der Endpunktadresse der Instance verwendet und muss die folgenden Bedingungen erfüllen:
   - Er darf 1 bis 63 alphanumerische Zeichen oder Bindestriche enthalten.
   - Das erste Zeichen muss ein Buchstabe sein.

- Er darf nicht mit einem Bindestrich enden oder zwei aufeinanderfolgende Bindestriche enthalten.
- Sie muss für alle DB-Instances in Ihrem AWS Konto in einer bestimmten AWS Region eindeutig sein.

4. Wählen Sie unter Vorlagen entweder Produktion oder Entwicklung und Testen aus.
5. Wählen Sie unter DB-Instance-Größe eine Instance-Größe aus. Dies legt die Verarbeitungs- und Arbeitsspeicherkapazität der primären Schreib-Instance des neuen DB-Clusters fest.

   Bei Auswahl der Vorlage Produktion können Sie nur aus den aufgelisteten verfügbaren speicheroptimierten Klassen wählen. Bei Auswahl der Option Entwicklung und Testen können Sie auch aus den wirtschaftlicheren burstfähigen Klassen wählen (siehe Burstable-T3-Instances für eine Beschreibung der burstfähigen Klassen.)

   > **Note**
   >
   > Neptune unterstützt keine R4 Instance-Typen mehr.

6. Unter Verfügbarkeit und Haltbarkeit können Sie wählen, ob Sie die multi-availability-zone (Multi-AZ-) Bereitstellung aktivieren möchten oder nicht. Die Produktionsvorlage unterstützt standardmäßig die Multi-AZ-Bereitstellung. Dies ist bei der Vorlage für Entwicklung und Testen nicht der Fall. Wenn die Multi-AZ-Bereitstellung aktiviert ist, lokalisiert Neptune Read-Replica-Instances, die Sie in verschiedenen Availability Zones () erstellen, um die Verfügbarkeit zu verbessern. AZs
7. Wählen Sie unter Konnektivität aus den verfügbaren Optionen die Virtual Private Cloud (VPC) aus, die Ihren neuen DB-Cluster hosten soll. Hier können Sie Neue VPC erstellen auswählen, wenn Neptune die VPC für Sie erstellen soll. Sie müssen eine Amazon-EC2-Instance in dieser VPC erstellen, um auf die Neptune-Instance zugreifen zu können (weitere Informationen finden Sie unter Schützen Sie Ihre Amazon Neptune Neptune-Datenbank mit Amazon VPC). Beachten Sie, dass Sie nach Erstellung des DB-Clusters die VPC nicht mehr ändern können.

   Sie können die Konnektivität für Ihren Cluster unter Zusätzliche Konnektivitätskonfiguration weiter konfigurieren, wenn notwendig:

   a. Unter Subnetzgruppe können Sie die Neptune-DB-Subnetzgruppe auswählen, die für den neuen DB-Cluster verwendet werden soll. Wenn Ihre VPC noch nicht über Subnetzgruppen verfügt, erstellt Neptune eine DB-Subnetzgruppe für Sie (siehe Schützen Sie Ihre Amazon Neptune Neptune-Datenbank mit Amazon VPC).

b. Wählen Sie unter VPC-Sicherheitsgruppen eine oder mehrere vorhandene VPC-Sicherheitsgruppen aus, um den Netzwerkzugriff auf den neuen DB-Cluster zu sichern, oder wählen Sie Neu erstellen aus, wenn Neptune eine Sicherheitsgruppe für Sie erstellen soll. Geben Sie dann einen Namen für die neue VPC-Sicherheitsgruppe ein (siehe Eine Sicherheitsgruppe über die Konsole erstellen).

c. Geben Sie unter Datenbankport den Port ein, den die Datenbank für TCP/IP Anwendungsverbindungen verwenden soll. Neptune verwendet standardmäßig die Portnummer 8182.

8. Wählen Sie unter Notebook-Konfiguration die Option Notebook erstellen aus, wenn Neptune in der Neptune-Workbench Jupyter-Notebooks für Sie erstellen soll (siehe Amazon Neptune mit Grafiknotizbüchern verwenden und Verwenden der Neptune-Workbench zum Hosten von Neptune-Notebooks). Anschließend können Sie auswählen, wie die neuen Notebooks konfiguriert werden sollen:

   a. Wählen Sie unter Notebook-Instance-Typ eine der verfügbaren Instance-Klassen für Ihr Notebook aus.

   b. Geben Sie unter Notebook-Name einen Namen für das Notebook ein.

   c. Sie können unter Beschreibung – optional auch eine Beschreibung des Notebooks eingeben.

   d. Geben Sie unter IAM-Rollenname an, dass Neptune eine IAM-Rolle für das Notebook erstellen soll, und geben Sie einen Namen für die neue Rolle ein oder wählen Sie eine vorhandene IAM-Rolle aus den verfügbaren Rollen aus.

   e. Wählen Sie abschließend aus, ob Ihr Notebook direkt oder über Amazon SageMaker AI oder über eine VPC mit einem NAT-Gateway eine Verbindung zum Internet herstellt. Weitere Informationen finden Sie unter Verbinden einer Notebook-Instance mit Ressourcen in einer VPC.

9. Unter Tags können Sie bis zu 50 Tags mit Ihrem neuen DB-Cluster verknüpfen.

10. Unter Zusätzliche Konfiguration gibt es weitere Einstellungen, die Sie für Ihren neuen DB-Cluster festlegen können (häufig können Sie diese überspringen und vorerst die Standardwerte akzeptieren):

Option	Was Sie tun können
DB-Instance-Kennung	Sie können einen Namen für die Writer-In stance des Clusters angeben. Andernfalls

Option	Was Sie tun können
	wird eine Standard-ID basierend auf dem Cluster-Namen verwendet. Wenn Sie dies tun, geben Sie einen Namen an, der für alle DB-Instances, die Ihrem AWS Konto gehören, in der aktuellen Region eindeutig ist. Die DB-Instance-Kennung unterscheidet zwischen Groß- und Kleinschreibung, wird jedoch in Kleinbuchstaben gespeichert.
DB-Cluster-Parametergruppe	Wählen Sie eine DB-Cluster-Paramet ergruppe aus, um die Standardkonfiguration für alle DB-Instances im Cluster zu definiere n. Neptune verwendet eine Standard-DB-Cluster-Parametergruppe, wenn Sie nichts anderes angeben. Weitere Informationen zu Parametergruppen finden Sie unter Amazon-Neptune-Parametergruppen.
DB-Parametergruppe	Wählen Sie eine DB-Parametergruppe aus, um die Konfiguration der primären DB-Instance im Cluster zu definieren. Neptune verwendet eine Standard-Parametergruppe, wenn Sie nichts anderes angeben. Weitere Informationen zu Parametergruppen finden Sie unter Parametergruppen.

Option	Was Sie tun können
IAM-DB-Authentifizierung	Wenn Sie die Option IAM-DB-Authentifizierung aktivieren aktivieren aktivieren, wird der gesamte Zugriff auf Ihre Datenbank mit AWS Identity and Access Management (IAM) authentifiziert. **Important** Dazu müssen Sie alle Anfragen mit AWS Signature Version 4 signieren. Weitere Informationen finden Sie unter Authentifizieren Sie Ihre Amazon Neptune Neptune-Datenbank mit AWS Identity and Access Management.
Failover-Priorität	Wählen Sie `No preference` oder eine Prioritätsstufe für Failover aus. Wenn Sie eine Stufe auswählen und es gibt einen Konflikt, wird das Replikat ausgewählt, das dieselbe Größe wie die primäre Instance hat.
Aufbewahrungszeitraum für Backups	Wählen Sie einen Zeitraum zwischen 1 und 35 Tagen aus, über den Neptune automatische Backups dieser DB-Instance aufbewahren soll. Sie können eine point-in-time Wiederherstellung (PITR) nur für einen Zeitpunkt innerhalb des Aufbewahrungszeitraums für Backups durchführen.
Tags zu Snapshots kopieren	(Standardmäßig aktiviert) Alle mit Ihrem DB-Cluster verknüpften Tags werden zu allen Snapshots des Clusters kopiert.

Option	Was Sie tun können
Verschlüsselung aktivieren	(Standardmäßig aktiviert) Die Daten in Ihrem DB-Cluster werden im Ruhezustand verschlüsselt. Wählen Sie den Masterschlüssel aus, der zum Schutz des Schlüssels für die Verschlüsselung dieses Datenbank-Volumes verwendet wird. Sie können den `aws/rds`-Standardschlüssel auswählen, einen der Masterschlüssel in Ihrem Konto auswählen oder den ARN eines Schlüssels aus einem anderen Konto eingeben. Sie können auf der Registerkarte Verschlüsselungsschlüssel der IAM-Konsole einen neuen Master-Verschlüsselungsschlüssel erstellen. Weitere Informationen finden Sie unter Neptun-Ressourcen im Ruhezustand verschlüsseln.
Prüfungsprotokoll	Aktivieren Sie diese Option, wenn Sie möchten, dass Audit-Logs aus Ihrem DB-Cluster in CloudWatch Logs veröffentlicht werden.
Nebenversions-Upgrades automatisch ausführen	(Standardmäßig aktiviert) Ihr DB-Cluster wird automatisch auf neue Engine-Nebenversionen aktualisiert, wenn sie veröffentlicht werden. Die automatischen Upgrades erfolgen während des Wartungsfensters für die Datenbank. Siehe Verwenden AutoMinorVersionUpgrade.

Option	Was Sie tun können
Wartungsfenster	Sie können einen bestimmten Zeitraum auswählen, in dem ausstehende Änderungen für Ihren DB-Cluster ausgeführt werden sollen, z. B. die Änderung einer DB-Instance-Klasse oder ein automatischer Engine-Patch. Diese Wartungsvorgänge werden innerhalb des ausgewählten Zeitraums gestartet und abgeschlossen. Wenn Sie keinen Zeitraum auswählen, weist Neptune einen Wartungszeitraum nach dem Zufallsprinzip zu.
Löschschutz aktivieren	(Standardmäßig aktiviert) Der Löschschutz verhindert, dass Ihr DB-Cluster gelöscht wird. Sie müssen ihn explizit deaktivieren, um den DB-Cluster zu löschen.

11. Wählen Sie Datenbank erstellen aus, um Ihren neuen Neptune-DB-Cluster und dessen primäre Instance zu starten.

    Der neue DB-Cluster wird in der Amazon-Neptune-Konsole in der Liste der Datenbanken angezeigt. Der DB-Cluster hat den Status Wird erstellt, bis er erstellt und einsatzbereit ist. Wenn der Status in Verfügbar geändert wird, können Sie die Verbindung mit der primären Instance des DB-Clusters herstellen. Je nach Klasse und Speicherort der DB-Instance kann es einige Minuten dauern, bis neue Instances verfügbar sind.

    Wählen Sie in der Neptune-Konsole die Ansicht Datenbanken aus, um den neu erstellten Cluster anzuzeigen.

    > **Note**
    >
    > Wenn Sie alle Neptune-DB-Instances in einem DB-Cluster mithilfe von löschen AWS-Managementkonsole, löscht die Konsole automatisch den DB-Cluster selbst. Wenn Sie das AWS CLI oder SDK verwenden, müssen Sie den DB-Cluster manuell löschen, nachdem Sie die letzte Instanz gelöscht haben.

Notieren Sie den Wert für Cluster-Endpunkt. Sie benötigen diesen Wert, um eine Verbindung mit Ihrem Neptune-DB-Cluster herzustellen.

Stoppen und Starten eines DB-Clusters in Amazon Neptune

Mithilfe des Stoppens und Startens von Amazon-Neptune-Clustern können Sie die Kosten für Entwicklungs- und Testumgebungen verwalten. Sie können vorübergehend alle DB-Instances in Ihrem Cluster stoppen, anstatt alle DB-Instances jedes Mal, wenn Sie den Cluster verwenden, einzurichten und zu entfernen.

Themen

- Übersicht über Stoppen und Starten eines Neptune-DB-Clusters
- Stoppen eines Neptune-DB-Clusters
- Starten eines gestoppten Neptune-DB-Clusters

Übersicht über Stoppen und Starten eines Neptune-DB-Clusters

In Zeiträumen, in denen Sie keinen Neptune-Cluster benötigen, können Sie alle Instances im betreffenden Cluster gleichzeitig stoppen. Sie können den Cluster jederzeit erneut starten, wenn notwendig. Durch das Starten und Stoppen werden die Einrichtungs- und Entfernungsvorgänge für Cluster erleichtert, die in der Entwicklung, für Tests oder ähnliche Aktivitäten verwendet werden, die keine kontinuierliche Verfügbarkeit erfordern. Sie können dies AWS-Managementkonsole mit einer einzigen Aktion erreichen, unabhängig davon, wie viele Instanzen sich im Cluster befinden.

Solange Ihr DB-Cluster gestoppt ist, werden Ihnen nur Gebühren für Cluster-Speicherspeicherplatz, manuelle Snapshots und automatisierten Sicherungsspeicher innerhalb des von Ihnen festgelegten Aufbewahrungsfensters berechnet. Für DB-Instance-Stunden werden Ihnen keine Gebühren berechnet.

Nach sieben Tagen startet Neptune Ihren DB-Cluster automatisch erneut, um keine erforderlichen Wartungs-Updates zu verpassen.

Zur Minimierung der Kosten für einen wenig ausgelasteten Neptune-Cluster können Sie den Cluster stoppen, statt alle Read Replicas zu löschen. Bei Clustern mit mehr als einer oder zwei Instances ist das häufige Löschen und Neuerstellen der DB-Instances nur mit der AWS CLI oder Neptune-API praktikabel. Außerdem kann es schwierig sein, Löschungen in der richtigen Reihenfolge durchzuführen. Beispielsweise müssen Sie alle Read Replicas löschen, bevor Sie die primäre Instance löschen, um die Aktivierung des Failover-Mechanismus zu vermeiden.

Verwenden Sie die Funktion zum Starten und Stoppen nicht, wenn Ihr DB-Cluster aktiv bleiben muss, Sie jedoch die Kapazität verringern möchten. Wenn Ihr Cluster zu kostspielig oder nicht sehr

ausgelastet ist, können Sie eine oder mehrere DB-Instances löschen oder Ihre DB-Instances so ändern, dass eine kleinere Instance-Klasse verwendet wird, Sie können jedoch keine einzelne DB-Instance stoppen.

Stoppen eines Neptune-DB-Clusters

Wenn Sie einen Neptune-DB-Cluster einige Zeit nicht verwenden werden, können Sie den ausgeführten DB-Cluster stoppen und erneut starten, wenn Sie ihn benötigen. Während der Cluster gestoppt ist, werden Ihnen Cluster-Speicherplatz, manuelle Snapshots und automatisierter Sicherungsspeicher innerhalb des von Ihnen festgelegten Aufbewahrungsfensters in Rechnung gestellt, nicht jedoch DB-Instance-Stunden.

Bei dem Stoppvorgang werden zunächst alle Read Replica-Instanzen des Clusters gestoppt, bevor die primäre Instance beendet wird, um die Aktivierung des Failover-Mechanismus zu vermeiden.

Stoppen eines DB-Clusters mithilfe der AWS-Managementkonsole

Um den zu verwenden AWS-Managementkonsole , um einen Neptun-Cluster zu stoppen

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die Amazon Neptune Neptune-Konsole zu Hausehttps://console.aws.amazon.com/neptune/.
2. Wählen Sie im Navigationsbereich die Option Databases (Datenbanken) und anschließend einen Cluster aus. Sie können den Stoppvorgang von dieser Seite ausführen oder zur Detailseite für den zu stoppenden DB-Cluster navigieren.
3. Wählen Sie unter Actions (Aktionen) die Option Stop (Stopp) aus.

Stoppen eines DB-Clusters mit dem AWS CLI

Um eine DB-Instance mithilfe von zu beenden AWS CLI, rufen Sie den stop-db-clusterBefehl auf und verwenden Sie dabei den `--db-cluster-identifier` Parameter, um den DB-Cluster zu identifizieren, den Sie beenden möchten.

Example

```
aws neptune stop-db-cluster --db-cluster-identifier mydbcluster
```

Stoppen eines DB-Clusters über die Neptune-Management-API

Um eine DB-Instance mithilfe der Neptune-Management-API zu stoppen, rufen Sie die DBClusterStop-API auf und verwenden Sie den `DBClusterIdentifier` Parameter, um den DB-Cluster zu identifizieren, den Sie stoppen möchten.

Mögliche Operationen, während ein DB-Cluster gestoppt ist

- Sie können ihn von einem Snapshot wiederherstellen (siehe Wiederherstellen aus einem DB-Cluster-Snapshot).
- Sie können die Konfiguration des DB-Clusters oder seiner DB-Instances nicht ändern.
- Sie können keine DB-Instances zu dem Cluster hinzufügen oder daraus entfernen.
- Sie können den Cluster nicht löschen, wenn ihm noch DB-Instances zugeordnet sind.
- Im Allgemeinen müssen Sie einen gestoppten DB-Cluster neu starten, um die meisten Verwaltungsaktionen ausführen zu können.
- Neptune wendet alle geplanten Wartungen auf Ihren gestoppten Cluster an, sobald dieser erneut gestartet wird. Nach sieben Tagen startet Neptune einen gestoppten Cluster automatisch neu, damit dessen Wartungsstatus nicht zu weit zurückfällt.
- Neptune führt keine automatisierten Sicherungen eines gestoppten DB-Clusters aus, da sich die zugrunde liegenden Daten nicht ändern können, während der Cluster gestoppt ist.
- Neptune verlängert den Aufbewahrungszeitraum für Backups des DB-Clusters nicht, während dieser gestoppt ist.

Starten eines gestoppten Neptune-DB-Clusters

Sie können nur einen gestoppten Neptune-DB-Cluster starten. Wenn Sie den Cluster starten, werden alle seine DB-Instances wieder verfügbar. Der Cluster behält seine Konfigurationseinstellungen, wie z. B. Endpunkte, Parametergruppen und VPC-Sicherheitsgruppen, bei.

Starten eines gestoppten DB-Clusters mit dem AWS-Managementkonsole

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die Amazon Neptune Neptune-Konsole zu Hausehttps://console.aws.amazon.com/neptune/.
2. Wählen Sie im Navigationsbereich die Option Databases (Datenbanken) und anschließend einen Cluster aus. Sie können den Startvorgang von dieser Seite ausführen oder zur Detailseite für den DB-Cluster navigieren und ihn von dort aus starten.

3. Wählen Sie unter Actions (Aktionen) die Option Start aus.

Starten eines gestoppten DB-Clusters mit dem AWS CLI

Um einen gestoppten DB-Cluster mit dem zu starten AWS CLI, rufen Sie den start-db-clusterBefehl mit dem `--db-cluster-identifier` Parameter auf, um den gestoppten DB-Cluster anzugeben, den Sie starten möchten. Geben Sie entweder den Clusternamen an, den Sie beim Erstellen des DB-Clusters ausgewählt haben, oder verwenden Sie einen von Ihnen ausgewählten DB-Instance-Namen, an den `-cluster` angefügt ist.

Example

```
aws neptune start-db-cluster --db-cluster-identifier mydbcluster
```

Starten eines gestoppten DB-Clusters über die Neptune-Management-API

Um einen Neptune-DB-Cluster mithilfe der Neptune-Management-API zu starten, rufen Sie die DBCluster Start-API auf, indem Sie den `DBCluster` Parameter verwenden, um den gestoppten DB-Cluster anzugeben, den Sie starten möchten. Geben Sie entweder den Clusternamen an, den Sie beim Erstellen des DB-Clusters ausgewählt haben oder verwenden Sie einen von Ihnen ausgewählten Namen einer DB-Instance mit dem Anhang `-cluster`.

Leeren eines Amazon-Neptune-DB-Clusters über die Fast-Reset-API

Über die Fast-Reset-REST-API können Sie ein Neptune-Diagramm schnell und einfach zurücksetzen und dessen Daten vollständig entfernen.

Sie können dies in einem Neptune-Notebook über die Zeilen-Magics %db_reset ausführen.

- In den meisten Fällen ist eine Fast-Reset-Operation innerhalb weniger Minuten abgeschlossen. Die Dauer ist in gewisser Weise von der Auslastung des Clusters bei Einleitung der Operation abhängig.
- Eine Fast-Reset-Operation führt nicht zu zusätzlichen E/As.
- Die Größe des Speicher-Volumes nimmt nach einem Fast Reset nicht ab. Stattdessen wird der Speicher wiederverwendet, wenn neue Daten eingefügt werden. Daher sind die Volume-Größen von Snapshots gleich, die vor und nach einer Fast-Reset-Operation erstellt werden. Die Volume-Größen wiederhergestellter Cluster, die Snapshots vor und nach einer Fast-Reset-Operation verwenden, sind ebenfalls gleich.
- Im Rahmen der Zurücksetzungsoperation werden alle Instances im DB-Cluster neu gestartet.

> **Note**
>
> In seltenen Fällen können diese Serverneustarts auch zu einem Failover des Clusters führen.

> **Important**
>
> Die Fast-Reset-Operation kann die Integration Ihres Neptune-DB-Clusters mit anderen Services unterbrechen. Beispiel:
>
> - Eine Fast-Reset-Operation löscht alle Stream-Daten aus Ihrer Datenbank und setzt Streams vollständig zurück. Daher müssen Sie Ihre Stream-Konsumenten möglicherweise neu konfigurieren, damit sie funktionieren.
> - Fast Reset entfernt alle Metadaten über SageMaker KI-Ressourcen, die von Neptune ML verwendet werden, einschließlich Jobs und Endpoints. Sie existieren weiterhin in SageMaker KI, und Sie können bestehende SageMaker KI-Endpunkte weiterhin für

Neptune ML-Inferenzabfragen verwenden, aber das Neptune ML-Management APIs funktioniert nicht mehr mit ihnen.

- Integrationen wie die full-text-search Integration mit ElasticSearch werden ebenfalls durch einen schnellen Reset gelöscht und müssen manuell neu eingerichtet werden, bevor sie wieder verwendet werden können.

Löschen aller Daten aus einem Neptune-DB-Cluster über die API

1. Zunächst generieren Sie ein Token, mit dem Sie dann die Datenbankzurücksetzung ausführen können. Dieser Schritt soll verhindern, dass eine Datenbank versehentlich zurückgesetzt wird.

   Hierzu senden Sie eine `HTTP POST`-Anforderung an den Endpunkt `/system` auf der Writer-Instance Ihres DB-Clusters, um die Aktion `initiateDatabaseReset` anzugeben.

   Der `curl`-Befehl, der den JSON-Inhaltstyp verwendet, wäre:

   ```
   curl -X POST \
     -H 'Content-Type: application/json' \
         https://your_writer_instance_endpoint:8182/system \
     -d '{ "action" : "initiateDatabaseReset" }'
   ```

   Oder mit dem Inhaltstyp `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 '
   ```

   Die Anforderung `initiateDatabaseReset` gibt das Zurücksetzungs-Token in der JSON-Antwort wie folgt zurück:

   ```
   {
     "status" : "200 OK",
     "payload" : {
       "token" : "new_token_guid"
     }
   }
   ```

Das Token bleibt nach der Ausgabe eine Stunde (60 Minuten) gültig.

Wenn Sie die Anforderung an eine Reader-Instance oder an den Statusendpunkt senden, gibt Neptune eine `ReadOnlyViolationException` aus.

Wenn Sie mehrere `initiateDatabaseReset`-Anforderungen senden, ist nur das zuletzt generierte Token für den zweiten Schritt gültig, in dem Sie die Zurücksetzung tatsächlich durchführen.

Wenn der Server direkt nach Ihrer `initiateDatabaseReset`-Anforderung neu gestartet wird, wird das generierte Token ungültig und Sie müssen eine neue Anforderung senden, um ein neues Token zu erhalten.

2. Als Nächstes senden Sie eine `performDatabaseReset`-Anforderung mit dem Token, das Sie von `initiateDatabaseReset` erhalten haben, an den Endpunkt `/system` auf der Writer-Instance Ihres DB-Clusters. Dies löscht alle Daten aus Ihrem DB-Cluster.

   Der Befehl `curl`, der den JSON-Inhaltstyp verwendet, wäre:

   ```
   curl -X POST \
     -H 'Content-Type: application/json' \
         https://your_writer_instance_endpoint:8182/system \
     -d '{
           "action" : "performDatabaseReset",
           "token" : "token_guid"
         }'
   ```

   Oder mit dem Inhaltstyp `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'
   ```

   Die Anforderung gibt eine JSON-Antwort zurück. Wenn die Anforderung akzeptiert wird, lautet die Antwort:

   ```
   {
     "status" : "200 OK"
   ```

```
}
```

Wenn das von Ihnen gesendete Token nicht mit dem ausgegebenen Token übereinstimmt, sieht die Antwort wie folgt aus:

```
{
  "code" : "InvalidParameterException",
  "requestId":"token_guid",
  "detailedMessage" : "System command parameter 'token' : 'token_guid' does not
 match database reset token"
}
```

Wenn die Anforderung akzeptiert wird und die Zurücksetzung beginnt, startet der Server neu und löscht die Daten. Sie können während der Zurücksetzung keine weiteren Anforderungen an den DB-Cluster senden.

Verwenden der Fast-Reset-API mit IAM-Auth

Wenn Sie IAM-Auth für Ihren DB-Cluster aktiviert haben, können Sie awscurl verwenden, um Fast-Reset-Befehle zu senden, die mit IAM-Auth authentifiziert werden:

Verwenden von awscurl zum Senden von Fast-Reset-Anforderungen mit IAM-Auth

1. Legen Sie die Umgebungsvariablen `AWS_ACCESS_KEY_ID` und `AWS_SECRET_ACCESS_KEY` korrekt fest (ebenfalls `AWS_SECURITY_TOKEN`, wenn Sie temporäre Anmeldeinformationen verwenden).
2. Eine `initiateDatabaseReset`-Anforderung sieht wie folgt aus:

```
awscurl -X POST --service neptune-db "$SYSTEM_ENDPOINT" \
  -H 'Content-Type: application/json' --region us-west-2 \
  -d '{ "action" : "initiateDatabaseReset" }'
```

3. Eine `performDatabaseReset`-Anforderung sieht wie folgt aus:

```
awscurl -X POST --service neptune-db "$SYSTEM_ENDPOINT" \
  -H 'Content-Type: application/json' --region us-west-2 \
  -d '{ "action" : "performDatabaseReset" }'
```

Verwenden der Zeilen-Magics `%db_reset` der Neptune-Workbench zum Zurücksetzen eines DB-Clusters

Die Neptune-Workbench unterstützt die Zeilen-Magics `%db_reset`, mit der Sie eine schnelle Datenbankzurücksetzung in einem Neptune-Notebook ausführen können.

Wenn Sie die Zeilen-Magics ohne Parameter aufrufen, wird ein Bildschirm angezeigt, in dem Sie gefragt werden, ob Sie alle Daten im Cluster löschen möchten. Mit der Aktivierung des Kontrollkästchens bestätigen Sie, dass die Cluster-Daten nach dem Löschen nicht mehr verfügbar sind. An diesem Punkt können Sie auswählen, ob Sie die Daten löschen oder den Vorgang abbrechen möchten.

Eine gefährlichere Option ist der Aufruf von `%db_reset` mit den Optionen `-y` oder `--yes`, wodurch der Löschvorgang ohne weitere Aufforderung ausgeführt wird.

Sie können die Zurücksetzung auch in zwei Schritten ausführen, genau wie bei der REST-API:

```
%db_reset --generate-token
```

Die Antwort ist:

```
{
  "status" : "200 OK",
  "payload" : {
    "token" : "new_token_guid"
  }
}
```

Aktion:

```
%db_reset --token new_token_guid
```

Die Antwort ist:

```
{
  "status" : "200 OK"
}
```

Häufige Fehlercodes für Fast-Reset-Operationen

Neptune-Fehlercode	HTTP-Status	Fehlermeldung	Beispiel
`InvalidParameterException`	400	Der Systembefehlsparameter '*`action`*' hat einen nicht unterstützten Wert '' *`XXX`*	Ungültiger Parameter
`InvalidParameterException`	400	Es wurden zu viele Werte angegeben für: *`action`*	Eine Anfrage zum schnellen Zurücksetzen mit mehr als einer Aktion, die mit dem Header x-www-form-urlencoded 'Content-Type:Application/' gesendet wurde
`InvalidParameterException`	400	Dupliziertes Feld 'action'	Eine Fast-Reset-Anforderung mit mehr als einer Aktion, die mit dem Header 'Content-Type: application/json' gesendet wurde
`MethodNotAllowedException`	400	Schlechte Route:/*`bad_endpoint`*	Anforderung wurde an einen falschen Endpunkt gesendet
`MissingParameterException`	400	Fehlende erforderliche Parameter: [action]	Eine Fast-Reset-Anforderung enthält den erforderlichen Parameter 'action' nicht

Neptune-Fehlercode	HTTP-Status	Fehlermeldung	Beispiel
`ReadOnlyViolationException`	400	Schreibvorgänge auf einer Read–Replica-Instance sind nicht zulässig	Eine Fast-Reset-Anforderung wurde an einen Lese- oder Statusendpunkt gesendet
`AccessDeniedException`	403	Fehlendes Authentifizierungstoken	Eine Fast-Reset-Anforderung wurde ohne korrekte Signaturen an einen DB-Endpunkt unter Aktivierung von IAM-Auth gesendet
`ServerShutdownException`	500	Die Datenbank zurücksetzung ist in Bearbeitung. Wiederholen Sie die Abfrage, wenn der Cluster verfügbar ist.	Beim Start der Fast-Reset-Operation schlagen vorhandene und eingehende Gremlin/SPARQL-Abfragen fehl.

Hinzufügen von Neptune-Reader-Instances zu einem DB-Cluster

In Neptune-DB-Clustern gibt es eine einzelne primäre DB-Instance und bis zu 15 Neptune-Reader-Instances. Die primäre DB-Instance unterstützt Lese- und Schreiboperationen und führt alle Datenänderungen im Cluster-Volume durch. Neptune-Reader-Instances stellen eine Verbindung zum selben Speicher-Volume her, in dem sich die primäre DB-Instance befindet, und unterstützen ausschließlich Leseoperationen.

Verwenden Sie Reader-Instances, um Lese-Workloads von der primären DB-Instance auszulagern.

Wir empfehlen Ihnen, die primäre Instance und die Neptune-Reader-Instances in Ihrem DB-Cluster über mehrere Availability Zones zu verteilen, um die Verfügbarkeit Ihres DB-Clusters zu verbessern.

Im folgenden Abschnitt wird beschrieben, wie Sie eine Reader-Instance in Ihrem DB-Cluster erstellen.

Erstellen einer Neptune-Reader-Instance über die Konsole

Nach der Erstellung der primären Instance für Ihren Neptune-DB-Cluster können Sie über die Neptune-Konsole zusätzliche Neptune-Reader-Instances hinzufügen.

Um eine Neptune-Reader-Instanz mit dem AWS-Managementkonsole

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die Amazon Neptune Neptune-Konsole zu Hausehttps://console.aws.amazon.com/neptune/.
2. Wählen Sie im Navigationsbereich Datenbanken aus.
3. Wählen Sie das DB-Cluster aus, in dem Sie die Reader-Instance erstellen möchten.
4. Wählen Sie Aktionen und dann Reader hinzufügen aus.
5. Geben Sie auf der Seite Replikat-DB-Instance erstellen die Optionen für Ihr Neptune-Replikat an. In der folgenden Tabelle werden Einstellungen für eine Neptune-Read-Replica angezeigt.

Für diese Option...	Vorgehensweise
DB-Instance-Klasse	Wählen Sie eine DB-Instance-Klasse aus, die die Verarbeitungs- und Arbeitsspeicheranforderungen für das Neptune-Replikat definiert. Eine aktuelle Liste der DB-Instance-Klassen, die von Neptune in verschied enen Regionen angeboten werden, finden Sie auf der Seite für Neptune-Preise.
Availability Zone	Geben Sie eine Availability Zone an. Wählen Sie eine andere Zone als die primäre DB-Instance aus. Die Liste beinhaltet nur jene Availability Zones, die von der DB-Subnetzgruppe für den DB-Cluster zugeordnet werden.
Verschlüsselung	Aktivieren oder deaktivieren Sie die Verschlüsselung.
Lese-Replikat-Quelle	Wählen Sie die Kennung der primären Instance aus, für die Sie ein Neptune-Replikat erstellen möchten.
DB-Instance-Kennung	Geben Sie einen Namen für die Instance ein, der eindeutig für Ihr Konto in der ausgewählten Region

Für diese Option...	Vorgehensweise
	ist. Sie können den Namen aussagekräftiger machen, indem Sie Angaben wie die ausgewählte Availability Zone hinzufügen, z. B. `neptune-us-east-1c` .
Datenbankport	Die Portnummer, an der die Datenbank Verbindungen akzeptiert.
DB-Parametergruppe	Dies ist die Parametergruppe für diese DB-Instance.
Protokollexporte	Wählen Sie die Protokolle aus, die Sie veröffentlichen möchten, wenn vorhanden.
Automatische Upgrades von Nebenversionen	Wählen Sie Ja aus, wenn Sie möchten, dass Ihr Neptune-Replikat automatisch Nebenversions-Updates für Ihre Neptune-DB-Engine erhält, sobald diese verfügbar sind. Die Option Auto Minor Version Upgrade (Upgrade einer Unterversion automatisch durchführen) gilt nur für kleinere Upgrades. Sie gilt nicht für Engine-Wartungs-Patches, die immer automatisch angewendet werden, um die Systemstabilität aufrechtzuerhalten.

6. Wählen Sie Read Replica erstellen aus, um die Neptune-Replikat-Instance zu erstellen.

Um eine Neptune-Reader-Instance aus einem DB-Cluster zu entfernen, folgen Sie den Anweisungen unter Löschen einer DB-Instance in Amazon Neptune.

Ändern eines Neptune-DB-Clusters über die Konsole

Wenn Sie eine DB-Instance mit dem ändern AWS-Managementkonsole, können Sie wählen, ob Sie die Änderungen sofort anwenden möchten, indem Sie Sofort anwenden auswählen. Wenn Sie sich entscheiden, die Änderungen sofort zu übernehmen, werden alle Ihre neuen Änderungen sowie alle ausstehenden Änderungen in der Warteschlange auf einmal übernommen.

Wenn Sie sich entscheiden, die Änderungen nicht sofort zu übernehmen, werden die Änderungen in die Warteschlange für ausstehende Änderungen aufgenommen. Während des nächsten Wartungsfensters, werden alle ausstehenden Änderungen in der Warteschlange angewandt.

Important

Wenn eine der ausstehenden Änderungen eine Ausfallzeit erfordert, kann die Auswahl der sofortigen Übernahme von Änderungen einen unerwarteten Ausfall für die entsprechende DB-Instance verursachen. Es gibt keine Ausfallzeiten für die anderen DB-Instances im DB-Cluster.

Note

Wenn Sie einen DB-Cluster in Neptune ändern, wirkt sich die Einstellung Sofort anwenden nur auf Änderungen für die DB-Cluster-ID und die IAM-DB-Authentifizierung aus. Alle anderen Änderungen werden sofort übernommen, unabhängig vom Wert der Einstellung für Apply Immediately (Sofort anwenden).

So ändern Sie einen DB-Cluster mithilfe der Konsole

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die Amazon Neptune Neptune-Konsole zu Hausehttps://console.aws.amazon.com/neptune/.
2. Wählen Sie im Navigationsbereich Cluster und anschließend den DB-Cluster aus, den Sie ändern möchten.
3. Wählen Sie Aktionen und dann Cluster ändern aus. Anschließend wird die Seite DB-Cluster ändern angezeigt.
4. Ändern Sie alle Einstellungen nach Bedarf.

> **Note**
>
> In der Konsole werden einige Änderungen auf Instance-Ebene nur auf die aktuelle DB-Instance angewendet, während andere auf den gesamten DB-Cluster angewendet werden. Um eine Einstellung zu ändern, die das gesamte DB-Cluster auf Instance-Ebene in der Konsole ändert, befolgen Sie die Anweisungen unter Ändern einer DB-Instance in einem DB-Cluster.

5. Nachdem Sie die gewünschten Änderungen vorgenommen haben, wählen Sie Continue (Weiter) und überprüfen Sie die Zusammenfassung.
6. Klicken Sie auf Apply immediately, um die Änderungen sofort anzuwenden.
7. Überprüfen Sie auf der Bestätigungsseite Ihre Änderungen. Wenn sie korrekt sind, wählen Sie Modify cluster (Cluster ändern) aus, um Ihre Änderungen zu speichern.

   Um Ihre Änderungen zu bearbeiten, wählen Sie Back (Zurück), oder um die Änderungen zu verwerfen, wählen Sie Cancel (Abbrechen).

Ändern einer DB-Instance in einem DB-Cluster

So ändern Sie eine DB-Instance in einem DB-Cluster mit der Konsole

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die Amazon Neptune Neptune-Konsole zu Hausehttps://console.aws.amazon.com/neptune/.
2. Wählen Sie im Navigationsbereich Instances aus und dann die DB-Instance, die Sie ändern möchten.
3. Wählen Sie Instance-Aktionen und anschließend Ändern aus. Die Seite DB-Instance ändern wird angezeigt.
4. Ändern Sie alle Einstellungen nach Bedarf.

> **Note**
>
> Einige Einstellungen gelten für das gesamte DB-Cluster und müssen auf Cluster-Ebene geändert werden. Um diese Einstellungen zu ändern, folgen Sie den Anweisungen in Ändern eines Neptune-DB-Clusters über die Konsole.

In der AWS-Managementkonsole gelten einige Änderungen auf Instance-Ebene nur für die aktuelle DB-Instance, während andere für den gesamten DB-Cluster gelten.

5. Nachdem Sie die gewünschten Änderungen vorgenommen haben, wählen Sie Continue (Weiter) und überprüfen Sie die Zusammenfassung.
6. Klicken Sie auf Apply immediately, um die Änderungen sofort anzuwenden.
7. Überprüfen Sie auf der Bestätigungsseite Ihre Änderungen. Wenn sie korrekt sind, wählen Sie Modify DB Instance (DB-Instance ändern) aus, um Ihre Änderungen zu speichern.

   Um Ihre Änderungen zu bearbeiten, wählen Sie Back (Zurück), oder um die Änderungen zu verwerfen, wählen Sie Cancel (Abbrechen).

Leistung und Skalierung in Amazon Neptune

Neptune-DB-Cluster und -Instances werden auf drei verschiedenen Ebenen skaliert:

- Speicherskalierung
- Skalieren von Instances;
- Skalieren von Lesevorgängen

Speicherskalierung in Neptune

Der Neptune-Speicher wird automatisch mit den Daten in Ihrem Cluster-Volume skaliert. Wenn Ihre Daten wachsen, wächst Ihr Cluster-Volumenspeicher auf bis zu 128 TiB in allen unterstützten Regionen außer China und GovCloud wo er auf 64 TiB begrenzt ist.

Die Größe Ihres Cluster-Volumes wird stündlich überprüft, um die Kosten für Ihren Speicherplatz zu bestimmen.

Der Speicherplatz, der von Ihrer Neptune-Datenbank verbraucht wird, wird pro GB-Monat und in Schritten abgerechnet, den Ihre Neptune-Datenbank verbrauchtI/Os consumed are billed in per million request increments. You pay only for the storage and I/Os, und Sie müssen ihn nicht im Voraus bereitstellen.

Informationen zu Preisen finden Sie auf der Neptune-Produktseite.

Skalieren von Instances in Neptune

Sie können Ihr Neptune-DB-Cluster wie notwendig skalieren, indem Sie die DB-Instance-Klasse für jede DB-Instance im DB-Cluster ändern. Neptune unterstützt mehrere optimierte DB-Instance-Klassen.

Skalieren von Lesevorgängen in Neptune

Sie können Lesevorgänge für Ihren Neptune-DB-Cluster skalieren, indem Sie bis zu 15 Neptune-Replikate im DB-Cluster erstellen. Jedes Neptune-Replikat gibt mit minimaler Replikatverzögerung die gleichen Daten aus dem Cluster-Volume zurück (häufig in weniger als 100 Millisekunden, nachdem die primäre Instance eine Aktualisierung geschrieben hat). Wenn der Lesevorgang-Datenverkehr zunimmt, können Sie zusätzliche Neptune-Replikate erstellen und direkte

Verbindungen mit ihnen herstellen, um die Leseauslastung für Ihren DB-Cluster zu verteilen. Neptune-Replikate müssen nicht zur selben DB-Instance-Klasse wie die primäre Instance gehören.

Informationen zum Hinzufügen von Neptune-Replikaten zu einem DB-Cluster finden Sie unter Hinzufügen von Reader-Instances.

Auto Scaling der Anzahl der Replikate in einem Amazon Neptune-DB-Cluster

Mit Neptune Auto Scaling können Sie die Anzahl der Neptune-Replikate in einem DB-Cluster automatisch an Ihre Konnektivitäts- und Workload-Anforderungen anpassen. Mit Auto Scaling kann Ihr Neptune-DB-Cluster einen steigenden Workload bewältigen. Wenn der Workload anschließend wieder abnimmt, entfernt Auto Scaling nicht notwendige Replikate, damit Sie nicht für ungenutzte Kapazitäten zahlen müssen.

Sie können Auto Scaling nur mit einem Neptune-DB-Cluster verwenden, der bereits über eine primäre Writer-Instance und mindestens eine Read-Replica-Instance verfügt (siehe Amazon-Neptune-DB-Cluster und -Instances). Außerdem müssen alle Read-Replica-Instances im Cluster verfügbar sein. Wenn eine Read-Replica nicht verfügbar ist, führt Neptune Auto Scaling keine Aktion aus, bis jede Read-Replica im Cluster verfügbar ist.

Informationen zum Erstellen eines neuen Clusters finden Sie unter Neptun-Cluster erstellen.

Mithilfe von definieren Sie eine Skalierungsrichtlinie und wenden sie auf den DB-Cluster an. AWS CLI Sie können die auch verwenden AWS CLI , um Ihre Auto-Scaling-Richtlinie zu bearbeiten oder zu löschen. Die Richtlinie gibt die folgenden Auto-Scaling-Parameter an:

- Die Mindest- und Höchstzahl von Replikaten im Cluster.
- Ein `ScaleOutCooldown` Intervall zwischen der Skalierungsaktivität beim Hinzufügen von Replikaten und einem `ScaleInCooldown` Intervall zwischen der Skalierungsaktivität beim Löschen von Replikaten.
- Die CloudWatch Metrik und der Metrik-Triggerwert für die Hoch- oder Herabskalierung.

Die Häufigkeit der Neptune-Auto-Scaling-Aktionen wird auf verschiedene Arten gedämpft:

- Damit ein Leser mittels Auto-Scaling hinzugefügt oder gelöscht wird, muss zunächst für `CPUUtilization` der obere Alarmschwellenwert mindestens 3 Minuten überschritten oder der untere Alarmschwellenwert mindestens 15 Minuten unterschritten werden.
- Nach dem ersten Hinzufügen oder Löschen wird die Häufigkeit der nachfolgenden Neptune-Auto-Scaling-Aktionen durch die Einstellungen für `ScaleOutCooldown` und `ScaleInCooldown` in der AutoSscaling-Richtlinie begrenzt.

Wenn die von Ihnen verwendete CloudWatch Metrik den hohen Schwellenwert erreicht, den Sie in Ihrer Richtlinie angegeben haben, und wenn das `ScaleOutCooldown` Intervall seit der letzten Auto-Scaling-Aktion abgelaufen ist und Ihr DB-Cluster noch nicht über die von Ihnen festgelegte maximale Anzahl von Replikaten verfügt, erstellt Neptune auto-scaling ein neues Replikat, das denselben Instance-Typ wie die primäre Instance des DB-Clusters verwendet.

Wenn die Metrik den von Ihnen angegebenen unteren Schwellenwert erreicht, das Intervall für `ScaleInCooldown` seit der letzten Auto-Scaling-Aktion verstrichen ist und Ihr DB-Cluster mehr als die von Ihnen festgelegte Mindestanzahl von Replikaten besitzt, löscht Neptune Auto Scaling ein Replikat.

> **Note**
>
> Neptune Auto Scaling entfernt nur Replikate, die es selbst erstellt hat. Bereits vorhandene Replikate werden nicht entfernt.

Mithilfe des DB-Cluster-Parameters neptune_autoscaling_config können Sie auch den Instance-Typ der neuen, von Neptune Auto Scaling erstellten Read-Replicas, die Wartungsfenster für diese Read-Replicas sowie die Tags angeben, die mit den neuen Read-Replicas verknüpft werden sollen. Sie geben diese Konfigurationseinstellungen in einer JSON-Zeichenfolge als Wert des Parameters `neptune_autoscaling_config` wie folgt an:

```
"{
  \"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\"
}"
```

Beachten Sie, dass alle Anführungszeichen in der JSON-Zeichenfolge mit einem Backslash-Zeichen (\) maskiert werden müssen. Alle Leerzeichen in der Zeichenfolge sind wie üblich optional.

Wenn eine dieser drei Konfigurationseinstellungen nicht im Parameter `neptune_autoscaling_config` angegeben ist, wird sie aus der Konfiguration der primären Writer-Instance des DB-Clusters kopiert.

Wenn Auto Scaling eine neue Read-Replica-Instance hinzufügt, wird der DB-Instance-ID das Präfix `autoscaled-reader` vorangestellt (z. B. `autoscaled-reader-7r7t7z3lbd-20210828`). Außerdem wird jeder erstellten Read-Replica ein Tag mit dem Schlüssel `autoscaled-reader` und dem Wert `TRUE` hinzugefügt. Sie finden dieses Tag auf Tags auf der Detailseite der DB-Instance in der AWS-Managementkonsole.

```
"key" : "autoscaled-reader",  "value" : "TRUE"
```

Die Heraufstufung aller mittels Auto Scaling erstellten Read-Replica-Instances hat die niedrigste Priorität, standardmäßig 15. Daher werden während eines Failovers alle Replikate mit einer höheren Priorität, beispielsweise manuell erstellte Replikate, zuerst heraufgestuft. Siehe Fehlertoleranz für einen Neptune-DB-Cluster.

Die auto-scaling von Neptune wird mithilfe von Application Auto Scaling mit einer Zielverfolgungs-Skalierungsrichtlinie implementiert, die eine `CPUUtilization` CloudWatch Neptune-Metrik als vordefinierte Metrik verwendet.

Verwenden von Auto Scaling in einem Neptune-Serverless-DB-Cluster

Neptune Serverless reagiert sehr viel schneller als Neptune Auto Scaling, wenn die Nachfrage die Kapazität einer Instance überschreitet, und skaliert die Instance hoch, statt eine weitere Instance hinzuzufügen. Während Auto Scaling für vergleichsweise stabile Zunahmen oder Abnahmen des Workloads vorgesehen ist, bewältigt Serverless kurzfristige Spitzen und Schwankungen beim Bedarf.

Sie können Auto Scaling und Serverless auf der Basis ihrer jeweiligen Vorteile kombinieren und so eine flexible Infrastruktur schaffen, um Workload-Änderungen effizient zu bewältigen, dem Bedarf gerecht zu werden und die Kosten zu minimieren.

Um Auto Scaling und Serverless effektiv zu kombinieren, müssen Sie für Ihr Serverless-Cluster `maxNCU` hoch genug festlegen, um Spitzen und kurzfristige Änderungen des Bedarfs zu bewältigen. Andernfalls lösen vorübergehende Änderungen keine Serverless-Skalierung aus, sodass Auto Scaling möglicherweise viele unnötige zusätzliche Instances erstellt. Wenn `maxNCU` hoch genug festgelegt ist, kann die Serverless-Skalierung diese Änderungen schneller und kostengünstiger bewältigen.

Aktivieren von Auto Scaling für Amazon Neptune

Auto Scaling kann für Neptune-DB-Cluster nur über die AWS CLI aktiviert werden. Sie können Auto Scaling nicht über die AWS-Managementkonsole aktivieren.

Außerdem wird Auto Scaling in den folgenden Amazon-Regionen nicht unterstützt:

- Afrika (Kapstadt): `af-south-1`
- Naher Osten (VAE): `me-central-1`
- AWS GovCloud (USA-Ost): `us-gov-east-1`
- AWS GovCloud (US-West): `us-gov-west-1`

Die Auto-Scaling-Aktivierung für Neptune-DB-Cluster umfasst drei Schritte:

1. Registrieren des DB-Clusters bei Application Auto Scaling

Der erste Schritt bei der Aktivierung von auto-scaling für einen Neptune DB-Cluster besteht darin, den Cluster bei Application Auto Scaling zu registrieren, indem Sie das AWS CLI oder eines der Application Auto Scaling verwenden. SDKs Der Cluster muss bereits über eine primäre Instance und mindestens eine Read-Replica-Instance verfügen:

Um beispielsweise einen Cluster für die automatische Skalierung mit einem bis acht zusätzlichen Replikaten zu registrieren, könnten Sie den AWS CLI `register-scalable-target`Befehl wie folgt verwenden:

```
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
```

Dies entspricht der Verwendung der Application-Auto-Scaling-API-Operation `RegisterScalableTarget`.

Der Befehl AWS CLI `register-scalable-target` verwendet die folgenden Parameter:

- **`service-namespace`** – Legen Sie diesen Parameter auf `neptune` fest.

  Dieser Parameter entspricht dem Parameter `ServiceNamespace` der Application-Auto-Scaling-API.
- **`resource-id`** – Legen Sie diesen Parameter auf die Ressourcen-ID für Ihren Neptune-DB-Cluster fest. Der Ressourcentyp ist `cluster`, gefolgt von einem Doppelpunkt (`:`) und dem Namen Ihres DB-Clusters.

Dieser Parameter entspricht dem Parameter `ResourceID` der Application-Auto-Scaling-API.

- **`scalable-dimension`** – Da die skalierbare Dimension in diesem Fall die Anzahl der Replikat-Instances im DB-Cluster ist, legen Sie diesen Parameter auf `neptune:cluster:ReadReplicaCount` fest.

  Dieser Parameter entspricht dem Parameter `ScalableDimension` der Application-Auto-Scaling-API.
- **`min-capacity`** – Die Mindestanzahl der DB-Reader-Replikat-Instances, die von Application Auto Scaling verwaltet werden sollen. Dieser Parameter sollte auf einen Wert zwischen 0 und 15 festgelegt werden und gleich oder kleiner als der Wert sein, der in `max-capacity` für die maximal Anzahl von Neptune-Replikaten angegeben ist. Es muss mindestens ein Leser im DB-Cluster vorhanden sein, damit Auto Scaling funktioniert.

  Dieser Parameter entspricht dem Parameter `MinCapacity` der Application-Auto-Scaling-API.
- **`max-capacity`** – Die maximale Anzahl von DB-Reader-Replikat-Instances im DB-Cluster, einschließlich bereits vorhandener Instances und neuer Instances, die von Application Auto Scaling verwaltet werden. Dieser Parameter muss auf einen Wert zwischen 0 und 15 festgelegt werden und gleich oder kleiner als der Wert sein, der in `min-capacity` für die Mindestanzahl von Neptune-Replikaten angegeben ist.

  Der `max-capacity` AWS CLI Parameter entspricht dem `MaxCapacity` Parameter in der Application Auto Scaling API.

Wenn Sie Ihren DB-Cluster registrieren, erstellt Application Auto Scaling die serviceverknüpfte Rolle `AWSServiceRoleForApplicationAutoScaling_NeptuneCluster`. Weitere Informationen finden Sie unter Serviceverknüpfte Rollen für Application Auto Scaling im Benutzerhandbuch für Application Auto Scaling.

2. Definieren einer Auto-Scaling-Richtlinie für Ihren DB-Cluster

Eine Skalierungsrichtlinie zur Zielverfolgung ist als JSON-Textobjekt definiert, das auch in einer Textdatei gespeichert werden kann. Für Neptune kann diese Richtlinie derzeit nur die `CPUUtilization` CloudWatch Neptune-Metrik als vordefinierte Metrik mit dem Namen verwenden. `NeptuneReaderAverageCPUUtilization`

Dies ist ein Beispiel für die Konfiguration einer Skalierungsrichtlinie zur Zielnachverfolgung für Neptune:

```
{
  "PredefinedMetricSpecification": { "PredefinedMetricType":
 "NeptuneReaderAverageCPUUtilization" },
  "TargetValue": 60.0,
  "ScaleOutCooldown" : 600,
  "ScaleInCooldown" : 600
}
```

Das Element **`TargetValue`** gibt den Prozentsatz der CPU-Auslastung an, ab dem Auto Scaling aufskaliert (d. h. weitere Replikate hinzufügt) und unter dem Auto-Scaling abskaliert (d. h. Replikate löscht). In diesem Fall ist der Zielprozentsatz für die Auslösung der Skalierung `60.0` %.

Das Element **`ScaleInCooldown`** gibt den Zeitraum in Sekunden an, der zwischen zwei Abskalierungen verstreichen muss. Standardmäßig ist ein Zeitraum von 300 Sekunden festgelegt. Hier gibt der Wert 600 an, dass zwischen dem Abschluss einer Replikatlöschung und dem Start einer anderen Replikatlöschung mindestens zehn Minuten liegen müssen.

Das Element **`ScaleOutCooldown`** gibt den Zeitraum in Sekunden an, der zwischen zwei Aufskalierungen verstreichen muss. Standardmäßig ist ein Zeitraum von 300 Sekunden festgelegt. Hier gibt der Wert 600 an, dass zwischen dem Abschluss einer Replikathinzufügung und dem Start einer anderen Replikathinzufügung mindestens zehn Minuten liegen müssen.

Das Element **`DisableScaleIn`** ist ein boolescher Wert. Wenn vorhanden und auf festgelegt, wird die Abskalierung vollständig deaktiviert. Das bedeutet, dass Auto Scaling Replikate hinzufügt, aber niemals entfernt. Standardmäßig ist die Abskalierung aktiviert und `DisableScaleIn` ist `false`.

Nach der Registrierung des Neptune-DB-Clusters bei Application Auto Scaling und der Definition einer JSON-Skalierungsrichtlinie in einer Textdatei wenden Sie die Skalierungsrichtlinie auf den registrierten DB-Cluster an. Sie können dazu den AWS CLI `put-scaling-policy`Befehl mit Parametern wie den folgenden verwenden:

```
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)
```

Nach der Anwendung der Auto-Scaling-Richtlinie ist Auto Scaling für Ihren DB-Cluster aktiviert.

Sie können den AWS CLI `put-scaling-policy`Befehl auch verwenden, um eine bestehende Auto-Scaling-Richtlinie zu aktualisieren.

Weitere Informationen finden Sie auch PutScalingPolicyin der API-Referenz für Application Auto Scaling.

Entfernen von Auto Scaling aus einem Neptune-DB-Cluster

Verwenden Sie die Befehle und, um die auto-scaling aus einem Neptune-DB-Cluster zu entfernen. AWS CLI delete-scaling-policyderegister-scalable-target

Warten eines Amazon-Neptune-DB-Clusters

Neptune führt regelmäßig Wartungsaktionen für alle verwendeten Ressourcen aus, einschließlich:

- Ersetzen der zugrunde liegende Hardware bei Bedarf. Dies geschieht im Hintergrund, ohne dass Sie etwas unternehmen müssen, und hat im Allgemeinen keine Auswirkungen auf Ihren Betrieb.
- Aktualisieren des zugrunde liegenden Betriebssystems. Betriebssystem-Upgrades der Instances in Ihrem DB-Cluster dienen der Verbesserung von Leistung und Sicherheit. Sie sollten sie daher generell so schnell wie möglich abschließen. In der Regel dauern die Updates etwa 10 Minuten. Betriebssystemupdates ändern nicht die DB-Engine-Version oder die DB-Instance-Klasse einer DB-Instance.

  Im Allgemeinen ist es am besten, zuerst die Reader-Instances in einem DB-Cluster und dann die Writer-Instance zu aktualisieren. Die gleichzeitige Aktualisierung der Reader und des Writers kann bei einem Failover zu Ausfallzeiten führen. Beachten Sie, dass DB-Instances vor einem Betriebssystem-Update nicht automatisch gesichert werden. Erstellen Sie daher unbedingt manuelle Backups, bevor Sie ein Betriebssystem-Update anwenden.
- Aktualisieren der Neptune-Datenbank-Engine. Neptune veröffentlicht regelmäßig verschiedene Engine-Updates, um neue Features und Verbesserungen einzuführen und Fehler zu beheben.

Engine-Versionsnummern

Versionsnummerierung vor Engine-Version 1.3.0.0

Vor November 2019 unterstützte Neptune nur jeweils eine Engine-Version gleichzeitig und alle Engine-Versionsnummern hatten die Form `1.0.1.0.200<xxx>`, wobei `xxx` die Patch-Nummer war. Alle neuen Engine-Versionen wurden als Patches für frühere Versionen veröffentlicht.

Seit November 2019 unterstützt Neptune mehrere Versionen, sodass Kunden eine bessere Kontrolle über ihre Upgrade-Pfade haben. Infolgedessen wurde die Nummerierung der Engine Releases geändert.

Ab November 2019 bis zur Engine-Version 1.3.0.0 bestehen die Engine-Versionsnummern aus 5 Teilen. Ein Beispiel ist `1.0.2.0.R2`:

- Der erste Teil war immer 1.
- Der zweite Teil (`0` in`1.0.2.0.R2`) war die Hauptversionsnummer der Datenbank.
- Der dritte und der vierte Teil (`2.0` in `1.0.2.0.R2`) waren beide Nebenversionsnummern.

- Der fünfte Teil (`R2` in `1.0.2.0.R2`) war die Patch-Nummer.

Bei den meisten Updates handelte es sich um Patch-Updates und die Unterscheidung zwischen Patches und Nebenversionsupdates war nicht immer deutlich.

Versionsnummerierung seit Engine-Version 1.3.0.0

Beginnend mit Engine-Version 1.3.0.0 wurde in Neptune die Art und Weise geändert, wie Engine-Updates nummeriert und verwaltet werden.

Die Versionsnummern der Engines bestehen nun aus vier Teilen, die jeweils einem Releasetyp entsprechen:

product-version.*major-version*.*minor-version*.*patch-version*

Non-breaking Änderungen, die zuvor als Patches veröffentlicht wurden, werden jetzt als Nebenversionen veröffentlicht, die Sie mit der `AutoMinorVersionUpgrade`Instanzeinstellung verwalten können.

Wenn Sie jedes Mal, wenn eine neue Nebenversion veröffentlicht wird, eine Benachrichtigung erhalten möchten, können Sie daher das `RDS-EVENT-0156`-Ereignis abonnieren (siehe Abonnieren von Neptune-Ereignisbenachrichtigungen).

Patch-Veröffentlichungen sind jetzt für dringende gezielte Korrekturen vorbehalten und werden mit dem letzten Teil der Versionsnummer nummeriert (`*.*.*.1`, `*.*.*.2` usw.).

Verschiedene Arten von Engine-Veröffentlichungen in Amazon Neptune

Die vier Arten von Engine-Veröffentlichungen, die den vier Teilen einer Engine-Versionsnummer entsprechen, lauten wie folgt:

- Produktversion: Diese ändert sich nur, wenn tiefgreifende, grundlegende Änderungen der Funktionalität oder der Benutzeroberfläche des Produkts erfolgen. Die aktuelle Neptune-Produktversion ist 1.
- Hauptversion: Hauptversionen führen wichtige neue Features und grundlegende Änderungen ein und haben in der Regel eine Nutzungsdauer von mindestens zwei Jahren.
- Nebenversion: Nebenversionen können neue Features, Verbesserungen und Fehlerkorrekturen enthalten, umfassen jedoch keine grundlegenden Änderungen. Sie können wählen, ob sie im nächsten Wartungsfenster automatisch angewendet werden sollen. Außerdem können Sie festlegen, ob Sie benachrichtigt werden möchten, wenn eine solche Version veröffentlicht wird.

- Patch-Version: Patch-Versionen werden nur veröffentlicht, um dringende Fehlerbehebungen oder kritische Sicherheitsupdates bereitzustellen. Sie enthalten selten grundlegende Änderungen und werden automatisch im nächsten Wartungsfenster nach ihrer Veröffentlichung angewendet.

Hauptversionsupdates von Amazon Neptune

Ein Hauptversionsupdate führt in der Regel eines oder mehrere wichtige neue Features ein und enthält häufig grundlegende Änderungen. Es hat in der Regel eine Support-Laufzeit von etwa zwei Jahren. Die Hauptversionen von Neptune sind in den Engine-Versionen zusammen mit dem Datum ihrer Veröffentlichung und ihrem voraussichtlichen Lebenszyklusende aufgeführt.

Updates für Hauptversionen sind rein optional, bis die von Ihnen verwendete Hauptversion das Ende ihrer Lebensdauer erreicht hat. Wenn Sie sich für ein Upgrade auf eine neue Hauptversion entscheiden, müssen Sie die neue Version selbst mithilfe der AWS CLI oder der Neptune-Konsole installieren, wie unter beschrieben. Hauptversions-Upgrades

Wenn die Hauptversion, die Sie verwenden, allerdings das Ende ihrer Lebensdauer erreicht, werden Sie darüber informiert, dass Sie auf eine neuere Hauptversion aktualisieren müssen. Wenn Sie dann nicht innerhalb einer Übergangsfrist nach der Benachrichtigung ein Upgrade durchführen, wird automatisch im nächsten Wartungsfenster ein Upgrade auf die neueste Hauptversion geplant. Weitere Informationen finden Sie unter Lebensdauern der Engine-Version.

Aktualisierungen der Nebenversion von Amazon Neptune

Bei den meisten Neptune-Engine-Updates handelt es sich um Updates der Nebenversion. Sie kommen relativ häufig vor und enthalten keine grundlegenden Änderungen.

Wenn Sie in der Writer-Instance (primären Instance) Ihres DB-Clusters das Feld `AutoMinorVersionUpgrade` auf `true` festgelegt haben, werden Nebenversionsupdates im nächsten Wartungsfenster nach ihrer Veröffentlichung automatisch auf alle Instances in Ihrem DB-Cluster angewendet.

Wenn Sie das Feld `AutoMinorVersionUpgrade` in der Writer-Instance Ihres DB-Clusters auf `false` festgelegt haben, werden sie nur angewendet, wenn Sie sie explizit installieren.

> **Note**
>
> Updates für Nebenversionen sind eigenständig (sie hängen nicht von vorherigen Nebenversionsupdates für dieselbe Hauptversion ab) und kumulativ (sie enthalten alle

Features und Korrekturen, die in früheren Nebenversionsupdates eingeführt wurden). Das bedeutet, dass Sie jedes beliebige Nebenversionsupdate installieren können, unabhängig davon, ob Sie frühere Updates installiert haben oder nicht.

Sie können ganz einfach den Überblick über Veröffentlichungen von Nebenversionen behalten, indem Sie das `RDS-EVENT-0156`-Ereignis abonnieren (siehe Abonnieren von Neptune-Ereignisbenachrichtigungen). Sie werden dann jedes Mal benachrichtigt, wenn eine neue Nebenversion veröffentlicht wird.

Unabhängig davon, ob Sie Benachrichtigungen abonniert haben, können Sie jederzeit überprüfen, welche Updates noch ausstehen.

Updates der Patch-Version von Amazon Neptune

Bei Sicherheitsproblemen oder anderen schwerwiegenden Fehlern, die die Zuverlässigkeit der Instance beeinträchtigen, stellt Neptune obligatorische Patches bereit. Sie werden während des nächsten Wartungsfensters auf alle Instances in Ihrem DB-Cluster angewendet, ohne dass Sie eingreifen müssen.

Eine Patch-Version wird nur bereitgestellt, wenn die Risiken einer Nichtbereitstellung die mit der Bereitstellung verbundenen Risiken und Ausfallzeiten übersteigen. Diese Patch-Veröffentlichungen erfolgen in unregelmäßigen Abständen (in der Regel alle paar Monate) und nehmen selten mehr als einen Bruchteil Ihres Wartungsfensters in Anspruch.

Planung für die Lebensdauer der Hauptversion der Amazon-Neptune-Engine

Neptun-Engine-Versionen erreichen fast immer am Ende eines Kalenderquartals das Ende ihres Lebenszyklus. Ausnahmen treten nur auf, wenn wichtige Sicherheits- oder Verfügbarkeitsprobleme auftreten.

Wenn eine Engine-Version das Ende ihres Lebenszyklus de erreicht, müssen Sie Ihre Neptune-Datenbank auf eine neuere Version aktualisieren.

Im Allgemeinen sind Neptune-Engine-Versionen weiterhin wie folgt verfügbar:

- Engine-Nebenversionen: Engine-Nebenversionen bleiben nach ihrer Veröffentlichung mindestens 6 Monate lang verfügbar.

- Engine-Hauptversionen: Engine-Hauptversionen bleiben nach ihrer Veröffentlichung mindestens 12 Monate lang verfügbar.

Mindestens 3 Monate, bevor eine Engine-Version ihr Lebensende erreicht, AWS sendet eine automatische E-Mail-Benachrichtigung an die mit Ihrem AWS Konto verknüpfte E-Mail-Adresse und veröffentlicht dieselbe Nachricht in Ihrem AWS Health Dashboard. So haben Sie Zeit, das Upgrade zu planen und vorzubereiten.

Wenn eine Engine-Version das Ende ihres Lebenszyklus erreicht, können Sie mit dieser Version keine neuen Cluster oder Instances mehr erstellen und auch Autoscaling kann keine Instances mit dieser Version erstellen.

Eine Engine-Version, die das Ende ihres Lebenszyklus erreicht hat, wird während eines Wartungszeitfensters automatisch aktualisiert. Die Nachricht, die Sie 3 Monate vor dem Ende des Lebenszyklus der Engine-Version erhalten, enthält Informationen darüber, was dieses automatische Update beinhalten würde, einschließlich der Version, auf die Sie automatisch aktualisiert werden würden, der Auswirkungen auf Ihre DB-Cluster und der von uns empfohlenen Maßnahmen.

Important

Sie sind dafür verantwortlich, Ihre Datenbank-Engine-Versionen auf dem neuesten Stand zu halten. AWS fordert alle Kunden nachdrücklich auf, ihre Datenbanken auf die jeweils neueste Engine-Version zu aktualisieren, um von den aktuellsten Sicherheits-, Datenschutz- und Verfügbarkeitsvorkehrungen zu profitieren. Wenn Sie Ihre Datenbank nach dem Verfallsdatum auf einer Engine oder Software betreiben, die nicht mehr unterstützt wird („Legacy Engine“), ist die Wahrscheinlichkeit von Sicherheits-, Datenschutz- und Betriebsrisiken, einschließlich Ausfallzeiten, höher.
Der Betrieb Ihrer Datenbank auf einer beliebigen Engine unterliegt der Vereinbarung, die Ihre Nutzung der AWS Dienste regelt. Ältere Engines sind nicht allgemein verfügbar. AWS bietet keinen Support mehr für die Legacy Engine an und AWS kann den Zugriff auf oder die Nutzung einer Legacy Engine jederzeit einschränken, wenn AWS festgestellt wird, dass die Legacy Engine ein Sicherheits- oder Haftungsrisiko oder ein Schadensrisiko für die Dienste AWS, ihre verbundenen Unternehmen oder Dritte darstellt. Ihre Entscheidung, Ihre Inhalte weiterhin in einer Legacy Engine auszuführen, könnte dazu führen, dass Ihre Inhalte nicht mehr verfügbar, beschädigt oder nicht wiederherstellbar sind. Für Datenbanken, die auf einer Legacy Engine ausgeführt werden, gelten Ausnahmen für Service Level Agreements (SLA). DATENBANKEN UND ZUGEHÖRIGE SOFTWARE, DIE AUF EINER ÄLTEREN ENGINE AUSGEFÜHRT WERDEN, ENTHALTEN BUGS, FEHLER, DEFEKTE UND AND/

OR SCHÄDLICHE KOMPONENTEN. DEMENTSPRECHEND UND UNGEACHTET ANDERSLAUTENDER BESTIMMUNGEN IN DER VEREINBARUNG ODER DEN SERVICEBEDINGUNGEN AWS WIRD DIE ÄLTERE ENGINE „WIE BESEHEN“ BEREITGESTELLT.

Verwalten von Engine-Updates für Ihren Neptune-DB-Cluster

> **Note**
>
> Updates werden auf alle Instances in einem DB-Cluster gleichzeitig angewendet. Ein Update erfordert einen Datenbankneustart auf diesen Instances. Daher kommt es zu einer Ausfallzeit von 20 bis 30 Sekunden bis zu mehreren Minuten. Anschließend können Sie die Nutzung des DB-Clusters fortsetzen. In seltenen Fällen kann ein Multi-AZ Failover erforderlich sein, damit ein Wartungsupdate für eine Instanz abgeschlossen werden kann.
> Bei Haupversionsupgrades, deren Anwendung länger dauern kann, können Sie eine Blau/Grün-Bereitstellungsstrategie verwenden, um Ausfallzeiten zu minimieren.

Ermitteln der derzeit verwendeten Engine-Version

Sie können den AWS CLI `get-engine-status`Befehl verwenden, um zu überprüfen, welche Engine-Release-Version Ihr DB-Cluster derzeit verwendet:

```
aws neptunedata get-engine-status
```

Die JSON-Ausgabe enthält ein `"dbEngineVersion"`-Feld wie dieses:

```
  "dbEngineVersion": "1.3.0.0",
```

Prüfen der ausstehenden und verfügbaren Updates

Sie können ausstehende Aktualisierungen Ihres DB-Clusters über die Neptune-Konsole überprüfen. Wählen Sie in der linken Spalte Datenbanken und dann Ihren DB-Cluster im Datenbankbereich aus. Ausstehende Updates sind in der Spalte Wartung aufgeführt. Wenn Sie Aktionen und dann Wartung auswählen, haben Sie drei Möglichkeiten:

- Jetzt Upgrade ausführen

- Im nächsten Wartungsfenster Upgrade ausführen
- Upgrade verschieben

Sie können ausstehende Engine-Updates AWS CLI wie folgt auflisten:

```
aws neptune describe-pending-maintenance-actions \
  --resource-identifier (ARN of your DB cluster)
  --region (your region) \
  --engine neptune
```

Sie können verfügbare Engine-Updates auch AWS CLI wie folgt auflisten:

```
aws neptune describe-db-engine-versions \
  --region (your region) \
  --engine neptune
```

Die Liste der verfügbaren Engine-Releases enthält nur Releases mit einer Versionsnummer, die höher als die aktuelle Versionsnummer ist. Außerdem muss ein Upgrade-Pfad definiert sein.

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Ein Nebenversionsupgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können, auch ohne dass es grundlegende Änderungen gibt.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Am besten testen Sie eine neue Version, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, indem Sie die Blue-Green Neptune-Bereitstellungslösung verwenden. Auf diese Weise können Sie Anwendungen und Abfragen in der neuen Version ausführen, ohne dass Ihr Produktions-DB-Cluster beeinträchtigt wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

Neptune-Wartungsfenster

Das wöchentliche Wartungsfenster umfasst einen Zeitraum von 30 Minuten, in dem geplante Engine-Updates und andere Systemänderungen vorgenommen werden. Die meisten Wartungsereignisse werden während des 30-minütigen Wartungsfensters abgeschlossen, wobei größere Wartungsereignisse manchmal länger dauern können.

Jeder DB-Cluster hat ein wöchentliches Wartungsfenster von 30 Minuten. Wenn Sie beim Erstellen des DB-Clusters keine bevorzugte Uhrzeit angeben, legt Neptune zufällig einen Wochentag fest und weist dann zufällig einen 30-minütigen Zeitraum aus einem 8-Stunden-Block zu, der je nach Region variiert.

Dies sind beispielsweise die 8-Stunden-Zeitblöcke für Wartungsfenster, die in verschiedenen AWS -Regionen verwendet werden:

Region	Zeitblock
Region USA West (Oregon)	06:00 bis 14:00 Uhr UTC
Region USA West (Nordkalifornien)	06:00 bis 14:00 Uhr UTC
Region USA Ost (Ohio)	03:00 bis 11:00 Uhr UTC
Region Europa (Irland)	22:00 bis 06:00 Uhr UTC

Das Wartungsfenster bestimmt, wann ausstehende Operationen beginnen, und die meisten Wartungsarbeiten werden innerhalb dieses Zeitfensters abgeschlossen. Größere Wartungsaufgaben können jedoch auch nach Ablauf des Zeitfensters fortgesetzt werden.

Verschieben des DB-Cluster-Wartungsfensters

Im Idealfall sollte Ihr Wartungsfenster in einer Zeit liegen, in der der Cluster am wenigsten genutzt wird. Wenn dies für Ihr aktuelles Zeitfenster nicht zutrifft, können Sie es zu einer günstigeren Zeit verschieben:

So ändern Sie das DB-Cluster-Wartungsfenster

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die Amazon Neptune Neptune-Konsole unter. https://console.aws.amazon.com/neptune/home
2. Wählen Sie im Navigationsbereich Datenbanken aus.
3. Wählen Sie das DB-Cluster aus, für das Sie den Wartungszeitraum ändern möchten.
4. Wählen Sie Ändern aus.
5. Wählen Sie unten auf der Seite Cluster ändern die Option Mehr anzeigen aus.
6. Stellen Sie im Abschnitt Bevorzugtes Wartungsfenster den Tag, die Uhrzeit und die Dauer des Wartungsfensters wie gewünscht ein.
7. Wählen Sie Weiter aus.

   Überprüfen Sie auf der Bestätigungsseite Ihre Änderungen.
8. Um die Änderungen am Wartungszeitraum sofort zu übernehmen, klicken Sie auf Sofort anwenden.
9. Wählen Sie Senden aus, um Ihre Änderungen zu übernehmen.

   Wenn Sie Ihre Änderungen bearbeiten möchten, wählen Sie Zurück aus. Wenn Sie die Änderungen verwerfen möchten, wählen Sie Abbrechen aus.

Wird AutoMinorVersionUpgrade zur Steuerung von automatischen Updates für kleinere Versionen verwendet

> **Important**
>
> `AutoMinorVersionUpgrade` kann nur für Nebenversionsupgrades ab Engine-Version 1.3.0.0 verwendet werden.

Automatische Upgrades kleinerer Versionen gelten nicht für Neptune-Cluster, die Teil einer globalen Datenbank sind. Sie können diese Option für eine DB-Instance in einem globalen Datenbankcluster festlegen, sie hat jedoch keine Auswirkung.

Wenn Sie in der Writer-Instance (primären Instance) Ihres DB-Clusters das Feld `AutoMinorVersionUpgrade` auf `true` festgelegt haben, werden Nebenversionsupdates automatisch im nächsten Wartungsfenster nach ihrer Veröffentlichung auf alle Instances in Ihrem DB-Cluster angewendet.

Sofern Sie das Feld `AutoMinorVersionUpgrade` in der Writer-Instance Ihres DB-Clusters auf `false` festgelegt haben, werden sie nur angewendet, wenn Sie sie explizit installieren.

Note

Patch-Releases (`*.*.*.1`, `*.*.*.2` usw.) werden immer automatisch während Ihres nächsten Wartungsfensters installiert, unabhängig davon, wie der `AutoMinorVersionUpgrade`-Parameter festgelegt wurde.

Sie können sie AWS-Managementkonsole wie folgt festlegen`AutoMinorVersionUpgrade`:

So richten Sie **`AutoMinorVersionUpgrade`** über die Neptune-Konsole ein

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die Amazon Neptune Neptune-Konsole unter. https://console.aws.amazon.com/neptune/home
2. Wählen Sie im Navigationsbereich Datenbanken aus.
3. Wählen Sie die primäre Instance (Writer) des DB-Clusters aus, für den Sie `AutoMinorVersionUpgrade` einrichten möchten.
4. Wählen Sie Ändern aus.
5. Wählen Sie unten auf der Seite Cluster ändern die Option Mehr anzeigen aus.
6. Wählen Sie unten auf der erweiterten Seite entweder Automatisches Upgrade von Unterversionen einschalten oder Automatisches Upgrade von Unterversionen ausschalten aus.
7. Wählen Sie Weiter aus.

   Überprüfen Sie auf der Bestätigungsseite Ihre Änderungen.

8. Zum Übernehmen der Änderungen an automatischen Nebenversionsupgrades klicken Sie auf Sofort anwenden.
9. Wählen Sie Senden aus, um Ihre Änderungen zu übernehmen.

   Wenn Sie Ihre Änderungen bearbeiten möchten, wählen Sie Zurück aus. Wenn Sie die Änderungen verwerfen möchten, wählen Sie Abbrechen aus.

Sie können das Feld auch verwenden AWS CLI , um das `AutoMinorVersionUpgrade` Feld festzulegen. Verwenden Sie beispielsweise einen Befehl wie diesen, um es auf `true` festzulegen:

```
aws neptune modify-db-instance \
  --db-instance-identifier (the ID of your cluster's writer instance) \
  --auto-minor-version-upgrade \
  --apply-immediately
```

Zum Festlegen auf `false` verwenden Sie entsprechend einen Befehl wie diesen:

```
aws neptune modify-db-instance \
  --db-instance-identifier (the ID of your cluster's writer instance) \
  --no-auto-minor-version-upgrade \
  --apply-immediately
```

Manuelles Installieren von Updates für Ihre Neptune-Engine

Installieren eines Engine-Hauptversionsupgrades

Engine-Hauptversionen müssen immer manuell installiert werden. Um Ausfallzeiten zu minimieren und genügend Zeit für Tests und Validierungen zur Verfügung zu haben, ist die Installation einer neuen Hauptversion in der Regel die Blue-Green Neptune-Bereitstellungslösung.

In einigen Fällen können Sie auch die CloudFormation Vorlage verwenden, mit der Sie Ihren DB-Cluster erstellt haben, um ein Upgrade der Hauptversion zu installieren (siehe Verwenden Sie eine CloudFormation Vorlage, um die Engine-Version Ihres Neptune DB-Clusters zu aktualisieren).

Wenn Sie ein Hauptversionsupdate sofort installieren möchten, können Sie einen CLI-Befehl wie den folgenden verwenden:

```
aws neptune modify-db-cluster \
```

```
  --db-cluster-identifier (identifier for your neptune cluster) \
  --engine neptune \
  --engine-version (the new engine version) \
  --apply-immediately
```

Sie müssen die Engine-Version angeben, auf die das Upgrade erfolgen soll. Wenn Sie dies nicht tun, wird die Engine möglicherweise nicht auf die neueste oder auf eine nicht erwartete Version aktualisiert.

Statt `--apply-immediately` können Sie `--no-apply-immediately` angeben.

Wenn Ihr Cluster eine benutzerdefinierte Cluster-Parametergruppe verwendet, müssen Sie diesen Parameter einschließen, um sie anzugeben:

```
  --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Ebenso sollte für Instances im Cluster, die eine benutzerdefinierte DB-Parametergruppe verwenden, dieser Parameter eingeschlossen werden, um sie zu spezifizieren:

```
  ---db-instance-parameter-group-name (name of the custom instance parameter group)
```

Installation eines Engine-Upgrades für eine Nebenversion mithilfe der AWS-Managementkonsole

Ausführen eines Nebenversionsupgrades mit der Neptune-Konsole

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die Amazon Neptune Neptune-Konsole unter. https://console.aws.amazon.com/neptune/home
2. Wählen Sie im Navigationsbereich Datenbanken und dann den zu ändernden DB-Cluster aus.
3. Wählen Sie Ändern aus.
4. Wählen Sie unter Instance-Spezifikationen die neue Version aus, auf die das Upgrade erfolgen soll.
5. Wählen Sie Weiter aus.
6. Wählen Sie Sofort anwenden aus, um die Änderungen sofort anzuwenden.
7. Wählen Sie Senden aus, um Ihren DB-Cluster zu aktualisieren.

Installation eines Engine-Upgrades für eine kleinere Version mit dem AWS CLI

Sie können einen Befehl wie den folgenden verwenden, um ein Nebenversionsupgrade auszuführen, ohne auf das nächste Wartungsfenster warten zu müssen:

```
aws neptune modify-db-cluster \
  --db-cluster-identifier (your-neptune-cluster) \
  --engine-version (new-engine-version) \
  --apply-immediately
```

Wenn Sie ein manuelles Upgrade mit dem durchführen AWS CLI, stellen Sie sicher, dass Sie die Engine-Version angeben, auf die Sie das Upgrade durchführen möchten. Tun Sie dies nicht, wird die Engine möglicherweise auf einen andere als die neueste oder nicht auf die erwartete Version upgegradet.

Upgrade zu einer Engine-Version 1.2.0.0 oder höher von einer früheren Engine-Version als 1.2.0.0

Mit der Engine-Version 1.2.0.0 wurden mehrere wichtige Änderungen eingeführt, die das Upgrade von einer früheren Version erschweren können:

- Mit der Engine-Version 1.2.0.0 wurde ein neues Format für benutzerdefinierte Parametergruppen und benutzerdefinierte Cluster-Parametergruppen eingeführt. Wenn Sie also von einer Engine-Version vor 1.2.0.0 auf Engine-Version 1.2.0.0 oder höher aktualisieren, müssen Sie alle vorhandenen benutzerdefinierten Parametergruppen und benutzerdefinierten Cluster-Parametergruppen mithilfe der Parametergruppenfamilie `neptune1.2` neu erstellen. In früheren Versionen wurde die Parametergruppenfamilie `neptune1` verwendet und diese Parametergruppen funktionieren nicht mit Version 1.2.0.0 und höher. Weitere Informationen finden Sie unter Amazon-Neptune-Parametergruppen.
- Mit der Engine-Version 1.2.0.0 wurde ein neues Format für Undo-Logs eingeführt. Wenn Sie also von einer Version vor 1.2.0.0 auf Version 1.2.0.0 oder höher aktualisieren, muss die `UndoLogListSize`Metrik unter einem bestimmten Schwellenwert liegen. Andernfalls wird der Patch zurückgesetzt und schlägt fehl. Die Schwellenwerte hängen vom Instance-Typ ab: Das Standardlimit liegt bei 40.000 für 4xlarge- oder größere Instances und 10.000 für Instances, die kleiner als 4xlarge sind. Wenn beim Versuch, ein Upgrade durchzuführen, das Limit `UndoLogListSize` überschritten wird, wird der Patch-Vorgang zurückgesetzt, das Upgrade wird abgebrochen und ein Ereignis mit dem Grund wird auf der Cluster-Ereignisseite angezeigt. Diese Grenzwerte können sich aus betrieblichen Gründen ohne vorherige Warnung ändern.

Sie können die Löschgeschwindigkeit beschleunigen, indem Sie die Writer-Instance des Clusters aktualisieren, in der das Löschen stattfindet. Wenn Sie dies tun, bevor Sie versuchen, ein Upgrade durchzuführen, kann dies dazu beitragen, den `UndoLogListSize` geltenden Schwellenwert zu unterschreiten. Wenn Sie die Größe des Writers auf einen 24XL-Instance-Typ erhöhen, kann sich Ihre Löschrate auf mehr als eine Million Datensätze pro Stunde erhöhen.

Wenn die `UndoLogListSize` CloudWatch Metrik extrem umfangreich ist, kann Ihnen die Eröffnung eines Support-Falls dabei helfen, zusätzliche Strategien zu finden, um sie unter das erforderliche Limit zu senken.

- Schließlich wurde mit Version 1.2.0.0 eine bahnbrechende Änderung eingeführt, die sich auf früheren Code auswirkt, der das Bolt-Protokoll mit IAM-Authentifizierung verwendete. Ab Version 1.2.0.0 benötigt Bolt einen Ressourcenpfad für die IAM-Signatur. In Java könnte das Festlegen des Ressourcenpfads so aussehen: `request.setResourcePath("/openCypher"));`. In anderen Sprachen kann `/openCypher` an den Endpunkt-URI angehängt werden. Beispiele finden Sie unter Verwenden des Bolt-Protokolls.

Verwenden Sie eine CloudFormation Vorlage, um die Engine-Version Ihres Neptune DB-Clusters zu aktualisieren

Sie können die CloudFormation Neptune-Vorlage, mit der Sie Ihren Neptune-DB-Cluster erstellt haben, wiederverwenden, um dessen Engine-Version zu aktualisieren.

Updates der Neptune-Engine-Version können Neben- oder Haupt-Updates sein. Die Verwendung einer CloudFormation Vorlage kann bei größeren Versionsupgrades hilfreich sein, die häufig erhebliche Änderungen beinhalten. Haupt-Updates können Datenbankänderungen enthalten, die nicht mit vorhandenen Anwendungen abwärtskompatibel sind. Daher müssen Sie möglicherweise auch Änderungen für Ihre Anwendungen ausführen. Führen Sie vor Updates stets Tests durch. Wir empfehlen Ihnen außerdem nachdrücklich, vor Updates einen manuellen Snapshot Ihres DB-Clusters zu erstellen.

Beachten Sie, dass Sie für jede Hauptversion ein eigenes Engine-Update durchführen müssen. Sie können eine Hauptversion nicht überspringen und direkt auf die folgende Hauptversion aktualisieren.

Wenn Sie vor dem 17. Mai 2023 den CloudFormation Neptune-Stack zum Upgrade Ihrer Engine-Version verwendet haben, hat er einfach einen neuen, leeren DB-Cluster anstelle Ihres aktuellen erstellt. Seit dem 17. Mai 2023 unterstützt der CloudFormation Neptune-Stack jedoch jetzt direkte Engine-Upgrades, bei denen Ihre vorhandenen Daten erhalten bleiben.

> **Note**
>
> Wenn Sie die verwenden, stellen Sie sicher AWS Cloud Development Kit (AWS CDK), dass die verwendete AWS CDK Version 2.82.0 oder höher ist. Versionen vor 2.82.0 unterstützen keine direkten Neptune-Engine-Upgrades.

Für ein Update auf eine Hauptversion sollte Ihre Vorlage die folgenden Eigenschaften im `DBCluster` festlegen:

- `DBClusterParameterGroup` (Benutzerdefiniert/Standard)
- `DBInstanceParameterGroupName`
- `EngineVersion`

In ähnlicher Weise sollten Sie für DBInstances attached to Folgendes einstellen DBCluster :

- `DBParameterGroup` (Benutzerdefiniert/Standard)

Alle Parametergruppen müssen in der Vorlage definiert werden, sowohl benutzerdefinierte als auch Standard-Parametergruppen.

Stellen Sie für benutzerdefinierte Parametergruppen sicher, dass die Familie der vorhandenen benutzerdefinierten Parametergruppe mit der neuen Engine-Version kompatibel ist. Engine-Versionen vor 1.2.0.0 verwenden die Parametergruppenfamilie `neptune1`. Engine-Versionen ab 1.2.0.0 erfordern die Parametergruppenfamilie `neptune1.2`. Weitere Informationen finden Sie unter Amazon-Neptune-Parametergruppen.

Geben Sie bei Updates auf Engine-Hauptversionen eine Parametergruppe mit der entsprechenden Familie in das `DBCluster`-Feld `DBInstanceParameterGroupName` ein.

Eine Standard-Parametergruppe sollte auf eine Parametergruppe aktualisiert werden, die mit der neuen Engine-Version kompatibel ist.

Beachten Sie, dass Neptune DB-Instances nach einem Engine-Update automatisch neu startet.

Themen

- Beispiel: Engine-Update von der Nebenversion 1.2.0.1 auf die Nebenversion 1.2.0.2
- Beispiel: Hauptversions-Update von 1.1.1.0 auf 1.2.0.2 mit Standard-Parametergruppen
- Beispiel: Hauptversions-Update von 1.1.1.0 auf 1.2.0.2 mit benutzerdefinierten Parametergruppen
- Beispiel: Hauptversions-Update von 1.1.1.0 auf 1.2.0.2 mit einer Mischung aus benutzerdefinierten und Standard-Parametergruppen

Beispiel: Engine-Update von der Nebenversion 1.2.0.1 auf die Nebenversion 1.2.0.2

Suchen Sie den DB-Cluster, den Sie aktualisieren möchten, und die Vorlage, die Sie zur Erstellung verwendet haben. Beispiel:

```
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
```

Aktualisieren Sie die Eigenschaft `EngineVersion` von `1.2.0.1` auf `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
```

Verwenden Sie jetzt CloudFormation , um die überarbeitete Vorlage auszuführen.

Beispiel: Hauptversions-Update von 1.1.1.0 auf 1.2.0.2 mit Standard-Parametergruppen

Suchen Sie den `DBCluster`, den Sie aktualisieren möchten, und die Vorlage, die Sie zur Erstellung verwendet haben. Beispiel:

```
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
```

- Aktualisieren Sie die Standard-`DBClusterParameterGroup` auf eine Parametergruppe in der Familie, die von der neuen Engine-Version verwendet wird (hier `default.neptune1.2`).

- Aktualisieren Sie für jede `DBInstance`, die dem `DBCluster` angefügt ist, die Standard-`DBParameterGroup` auf eine Parametergruppe in der Familie, die von der neuen Engine-Version verwendet wird (hier `default.neptune1.2`).
- Legen Sie die Eigenschaft `DBInstanceParameterGroupName` auf die Standardparametergruppe in dieser Familie fest (hier `default.neptune1.2`).
- Aktualisieren Sie die Eigenschaft `EngineVersion` von `1.1.0.0` auf `1.2.0.2`.

Die Vorlage sollte wie folgt aussehen:

```
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:
```

Wird jetzt verwendet CloudFormation , um die überarbeitete Vorlage auszuführen.

Beispiel: Hauptversions-Update von 1.1.1.0 auf 1.2.0.2 mit benutzerdefinierten Parametergruppen

Suchen Sie den `DBCluster`, den Sie aktualisieren möchten, und die Vorlage, die Sie zur Erstellung verwendet haben. Beispiel:

```
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
```

- Aktualisieren Sie die benutzerdefinierte `DBClusterParameterGroup` Familie auf die Familie, die von der neuen Engine-Version verwendet wird `default.neptune1.2` (hier).
- Aktualisieren Sie für jede `DBInstance` angehängte Familie die benutzerdefinierte `DBParameterGroup` Familie auf die Familie`DBCluster`, die von der neuen Engine-Version verwendet wird (hier`default.neptune1.2`).
- Legen Sie die Eigenschaft `DBInstanceParameterGroupName` auf die benutzerdefinierte Parametergruppe in dieser Familie fest (hier `default.neptune1.2`).
- Aktualisieren Sie die Eigenschaft `EngineVersion` von `1.1.0.0` auf `1.2.0.2`.

Die Vorlage sollte wie folgt aussehen:

```
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
```

Verwenden Sie jetzt CloudFormation , um die überarbeitete Vorlage auszuführen.

Beispiel: Hauptversions-Update von 1.1.1.0 auf 1.2.0.2 mit einer Mischung aus benutzerdefinierten und Standard-Parametergruppen

Suchen Sie den `DBCluster`, den Sie aktualisieren möchten, und die Vorlage, die Sie zur Erstellung verwendet haben. Beispiel:

```
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
```

- Aktualisieren Sie für eine benutzerdefinierte Cluster-Parametergruppe die `DBClusterParameterGroup`-Familie auf die von der neuen Engine-Version verwendete Parametergruppenfamilie (`neptune1.2`).
- Aktualisieren Sie für eine Standard-Cluster-Parametergruppe die `DBClusterParameterGroup`-Familie auf die von der neuen Engine-Version verwendete Standard-Parametergruppenfamilie (`default.neptune1.2`).
- Aktualisieren Sie für jede `DBInstance`, die dem `DBCluster` angefügt ist, eine Standard-`DBParameterGroup` auf eine Parametergruppe in der Familie, die von der neuen Engine-Version verwendet wird (hier `default.neptune1.2`). Aktualisieren Sie eine benutzerdefinierte Parametergruppe auf eine Parametergruppe in der Familie, die von der neuen Engine-Version unterstützt wird (hier `neptune1.2`).
- Legen Sie die Eigenschaft `DBInstanceParameterGroupName` auf eine Parametergruppe in der Familie fest, die von der neuen Engine-Version unterstützt wird.

Die Vorlage sollte wie folgt aussehen:

```
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
```

Wird jetzt verwendet CloudFormation , um die überarbeitete Vorlage auszuführen.

Klonen von Datenbanken in Neptune

Mit dem Klonen von Datenbanken können Sie schnell und kosteneffizient Klone aller Ihrer Datenbanken in Amazon Neptune erstellen. Die Klon-Datenbanken erfordern nur eine kleine Menge an zusätzlichem Speicherplatz beim ersten Erstellungsvorgang. Beim Klonen von Datenbanken wird ein copy-on-write Protokoll verwendet. Die Daten werden zu dem Zeitpunkt, an dem sie sich ändern, entweder in den Quelldatenbanken oder in den Klondatenbanken kopiert. Sie können mehrere Klone aus demselben DB-Cluster erstellen. Es ist auch möglich, zusätzliche Klone aus anderen Klonen zu erstellen. Weitere Informationen zur Funktionsweise des copy-on-write Protokolls im Kontext von Neptune Storage finden Sie unter. Copy-on-Write Protocol (Protokoll)

Sie können das Klonen von Datenbanken in einer Vielzahl von Anwendungsfällen einsetzen, insbesondere dann, wenn Sie keine Auswirkungen auf Ihre Produktionsumgebung haben wollen, wie zum Beispiel in den folgenden Fällen:

- Experimentieren Sie mit Änderungen und bewerten Sie die Auswirkungen, wie z. B. die Änderung von Schemata oder Parametergruppen.
- Führen Sie Workload-intensive Vorgänge durch, wie zum Beispiel das Exportieren von Daten oder das Ausführen von analytischen Abfragen.
- Erstellen Sie eine Kopie eines Produktions-DB-Clusters in einer Nicht-Produktionsumgebung zu Entwicklungs- oder Testzwecken.

Um einen Klon eines DB-Clusters mit dem zu erstellen AWS-Managementkonsole

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die Amazon Neptune Neptune-Konsole zu Hausehttps://console.aws.amazon.com/neptune/.
2. Wählen Sie im Navigationsbereich Instances aus. Wählen Sie eine primäre Instance für das DB-Cluster aus, für das Sie einen Klon erstellen möchten.
3. Wählen Sie Instance actions (Instance-Aktionen) und anschließend Create clone (Klon erstellen) aus.
4. Geben Sie auf der Seite Create Clone (Klon erstellen) einen Namen für die primäre Instance des DB-Klon-Clusters als DB Instance Identifier (DB-Instance-ID) ein.

   Wenn Sie möchten, können Sie weitere beliebige Einstellungen für das Klon-DB-Cluster konfigurieren. Weitere Informationen zu den unterschiedlichen DB-Cluster-Einstellungen finden Sie unter Starten über die Konsole.

5. Wählen Sie Create Clone (Klon erstellen) aus, um das DB-Klon-Cluster zu starten.

Um einen Klon eines DB-Clusters zu erstellen, verwenden Sie AWS CLI

- Rufen Sie den point-in-time AWS CLI Befehl Neptune restore-db-cluster-to- auf und geben Sie die folgenden Werte ein:
  - `--source-db-cluster-identifier` – Der Name des DB-Quell-Clusters, dessen Klon erstellt werden soll.
  - `--db-cluster-identifier` – Der Name des DB-Klon-Clusters.
  - `--restore-type copy-on-write` – Der Wert für `copy-on-write` gibt an, dass ein DB-Klon-Cluster erstellt werden soll.
  - `--use-latest-restorable-time` – Dies gibt an, dass die letzte Sicherung verwendet werden soll, die wiederhergestellt werden kann.

> **Note**
>
> Der point-in-time AWS CLI Befehl restore-db-cluster-to- klont nur den DB-Cluster, nicht die DB-Instances für diesen DB-Cluster.

Das folgende Linux/UNIX Beispiel erstellt einen Klon aus dem `source-db-cluster-id` DB-Cluster und benennt den Klon`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
```

Das gleiche Beispiel funktioniert unter Windows, wenn das \-Zeilenende-Escape-Zeichen durch das Windows-^-Äquivalent ersetzt wird:

```
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
```

Einschränkungen

Das DB-Klonen in Neptune unterliegt den folgenden Einschränkungen:

- Sie können keine Klondatenbanken AWS regionsübergreifend erstellen. Die Klon-Datenbanken müssen in derselben Region wie die Quelldatenbanken erstellt worden sein.
- Eine Klon-Datenbank verwendet stets den neuesten Patch der Neptune-Engine-Version der Datenbank, deren Klon sie ist. Dies gilt auch dann, wenn die Quelldatenbank noch nicht auf diese Patch-Version aktualisiert wurde. Die Engine-Version selbst ändert sich jedoch nicht.
- Zurzeit können Sie maximal 15 Klone pro Kopie Ihres Neptune-DB-Clusters erstellen, einschließlich Klonen von anderen Klonen. Wenn Sie dieses Limit erreicht haben, müssen Sie eine weitere Kopie Ihrer Datenbank erstellen, anstatt sie zu klonen. Wenn Sie eine neue Kopie erstellen, kann diese jedoch erneut bis zu 15 Klone haben.
- Die kontoübergreifende DB-Klonung wird aktuell nicht unterstützt.
- Sie können eine andere Virtual Private Cloud (VPC) für Ihren Klon anbieten. Die Subnetze in diesen VPCs müssen jedoch derselben Gruppe von Availability Zones zugeordnet sein.

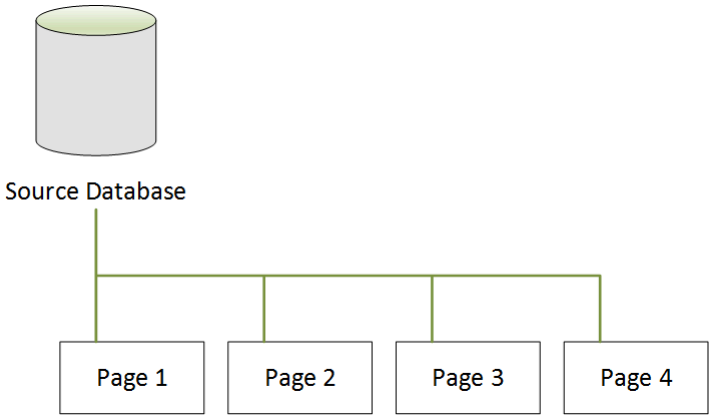
Copy-on-Write Protokoll für das DB-Klonen

Die folgenden Szenarien veranschaulichen, wie das copy-on-write Protokoll funktioniert.

- Neptune-Datenbank vor dem Klonen
- Neptune-Datenbank nach dem Klonen
- Bei einer Änderung an der Quelldatenbank
- Bei einer Änderung an der Klondatenbank

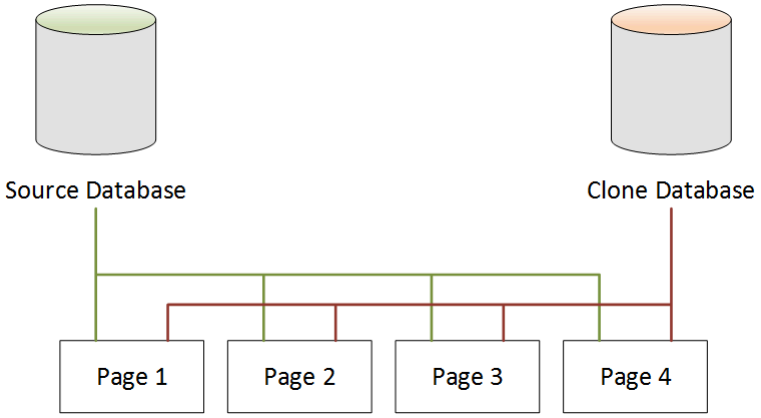
Neptune-Datenbank vor dem Klonen

Die Daten in einer Quelldatenbank werden in Seiten gespeichert. Im folgenden Diagramm hat die Quelldatenbank vier Seiten.



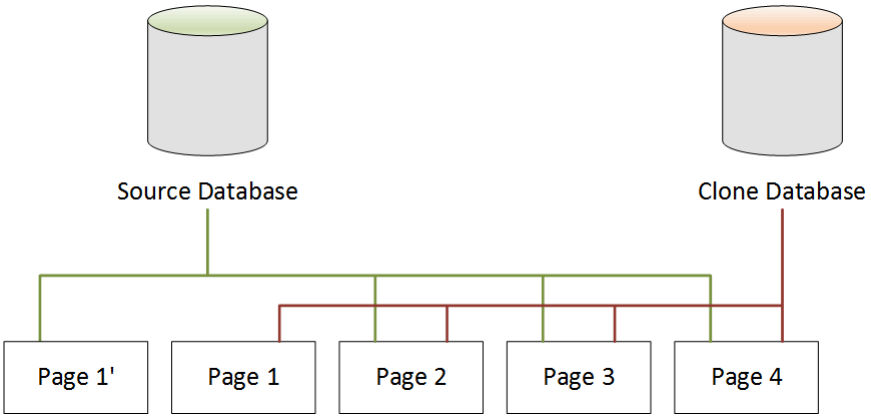
Neptune-Datenbank nach dem Klonen

Nach der Datenbankklonung gibt es, wie das folgende Diagramm zeigt, in der Quelldatenbank keine Änderungen. Sowohl die Quelldatenbank als auch die Klondatenbank verweisen auf dieselben vier Seiten. Es wurden keine Seiten physikalisch kopiert, daher wird kein zusätzlicher Speicherplatz benötigt.



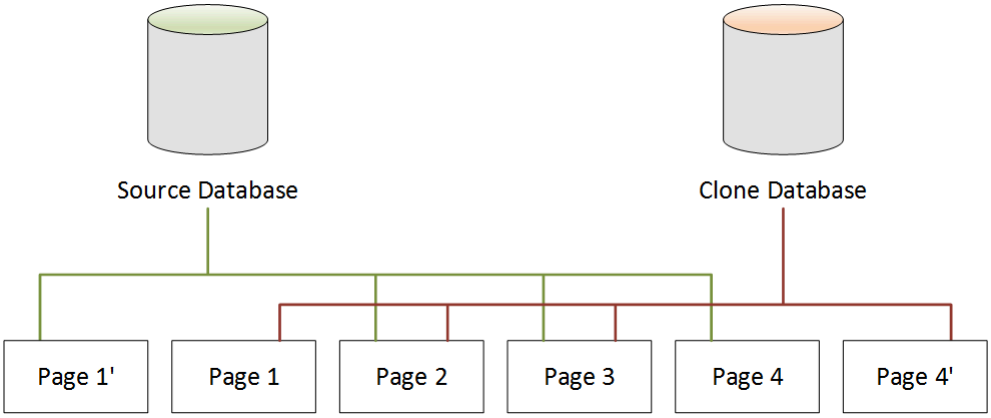
Bei einer Änderung an der Quelldatenbank

Im folgenden Beispiel wird in der Quelldatenbank eine Änderung an den Daten auf `Page 1` vorgenommen. Anstatt in den ursprünglichen `Page 1` zu schreiben, wird zusätzlicher Speicher verwendet, um eine neue Seite mit dem Namen `Page 1'` zu erstellen. Die Quelldatenbank verweist nun auf die neue `Page 1'` und auch auf `Page 2`, `Page 3` und `Page 4`. Die Klondatenbank verweist weiterhin auf `Page 1` über `Page 4`.



Bei einer Änderung an der Klondatenbank

Im folgenden Diagramm hat sich auch die Klondatenbank geändert, diesmal auf `Page 4`. Anstatt in den ursprünglichen `Page 4` zu schreiben, wird zusätzlicher Speicher verwendet, um eine neue Seite mit dem Namen `Page 4'` zu erstellen. Die Quelldatenbank verweist weiter auf `Page 1'` und auch auf `Page 2` über `Page 4`. Die Klondatenbank verweist jetzt jedoch auf `Page 1` über `Page 3` und auch auf `Page 4'`.



Wie im zweiten Szenario gezeigt, ist nach der Datenbankklonung zum Zeitpunkt der Klon-Erstellung kein zusätzlicher Speicherplatz erforderlich. Da jedoch Änderungen sowohl an der Quell- als auch an der Klondatenbank vorgenommen werden, werden nur die geänderten Seiten erstellt, wie im dritten und vierten Szenario erklärt wird. Mit den über die Zeit zunehmenden Änderungen in der Quell- und Klondatenbank werden Sie zunehmend mehr Speicherplatz benötigen, um Änderungen zu erfassen und zu speichern.

Löschen einer Quelldatenbank

Das Löschen einer Quelldatenbank wirkt sich nicht auf die zugehörigen Klondatenbanken aus. Die Klondatenbanken verweisen weiterhin auf die Seiten, die vorher zur Quelldatenbank gehörten.

Verwalten von Amazon-Neptune-Instances

In den folgenden Abschnitten finden Sie Informationen zu Vorgängen auf Instance-Ebene.

Themen

- Neptune-Burstable-T3-Instance-Klasse
- Ändern einer Neptune-DB-Instance (und sofortige Anwendung)
- Umbenennen einer Neptune-DB-Instance
- Neustarten einer DB-Instance in Amazon Neptune
- Löschen einer DB-Instance in Amazon Neptune

Neptune-Burstable-T3-Instance-Klasse

Zusätzlich zu Instance-Klassen mit fester Leistung wie R5 und R6 ermöglicht Amazon Neptune die Verwendung einer Burstable Performance Instance. Während der Entwicklung Ihrer Diagrammanwendung sollte Ihre Datenbank schnell und responsiv sein, aber Sie benötigen sie nicht laufend. Die Neptune-Instance-Klasse `db.t3.medium` ist genau die Klasse, die Sie in dieser Situation verwenden sollten. Sie ist deutlich kostengünstiger als die kostengünstigste Instance-Klasse mit fester Leistung.

Eine Burstable Instance wird mit einer CPU-Leistung auf Baseline-Stufe ausgeführt, bis ein Workload mehr Leistung benötigt. In diesem Fall wird die Leistung mit einem Burst so lange weit über die Baseline hinaus gesteigert, wie dies für den Workload erforderlich ist. Die Bursts sind im Stundenpreis enthalten, sofern die durchschnittliche CPU-Auslastung die Baseline über einen 24-stündigen Zeitraum hinweg nicht überschreitet. Für die meisten Entwicklungs- und Testsituationen bedeutet das eine gute Leistung zu niedrigen Kosten.

Wenn Sie mit einer T3 Instance-Klasse beginnen, können Sie später problemlos zu einer Instance-Klasse mit fester Leistung wechseln, wenn Sie bereit sind, in die Produktion zu gehen, indem Sie die, AWS-Managementkonsole AWS CLI, oder eine der folgenden Optionen verwenden. AWS SDKs

T3-Bursting wird durch CPU-Guthaben geregelt

Ein CPU-Guthaben repräsentiert die einminütige volle Auslastung eines virtuellen CPU Cores (vCPU). Dies kann auch zu einer 50-prozentigen Auslastung einer vCPU für zwei Minuten oder einer 25-prozentigen Auslastung von zwei V CPUs für zwei Minuten usw. führen.

Eine T3-Instance verdient CPU-Guthaben, wenn sie im Leerlauf ist, und verbraucht sie, wenn sie aktiv ist, wobei beides in Millisekundenauflösung gemessen wird. Die `db.t3.medium` Instance-Klasse hat zwei VCPUs, von denen jede im Leerlauf 12 CPU-Credits pro Stunde verdient. Dies bedeutet, dass eine 20%ige Auslastung jeder vCPU zu einem CPU-Nullguthaben führt. Die 12 verdienten CPU-Guthaben werden durch 20%ige Auslastung der vCPU aufgebraucht (da 20 % von 60 Minuten ebenfalls 12 ergibt). Diese 20%ige Auslastung ist somit die Baseline-Auslastung, die weder ein positives noch ein negatives CPU-Guthaben bewirkt.

Leerlaufzeit (CPU-Auslastung unter 20 % des verfügbaren Gesamtbetrags) bewirkt, dass CPU-Guthaben in einem Guthaben-Saldo-Bucket gespeichert werden, und zwar bis zu dem Limit 576 für eine `db.t3.medium`-Instance-Klasse (dies ist die maximale Anzahl von CPU-Guthaben, die innerhalb von 24 Stunden auflaufen können, nämlich 2 x 12 x 24). Oberhalb dieses Grenzwerts werden CPU-Guthaben einfach verworfen.

Bei Bedarf kann sich die CPU-Auslastung mit einem Burst so lange auf bis zu 100 % steigern, wie dies für einen Workload erforderlich ist, auch wenn das CPU-Guthaben unter Null fällt. Wenn die Instance 24 Stunden lang einen negativen Saldo aufrechterhält, fällt eine zusätzliche Gebühr in Höhe von 0,05 USD pro -60 CPU-Guthaben in diesem Zeitraum an. Bei den meisten Entwicklungs- und Test-Workloads wird das Bursting üblicherweise jedoch durch Leerlaufzeit vor oder nach dem Burst abgedeckt.

> **Note**
>
> Die T3 Instance-Klasse von Neptune ist wie der Amazon EC2 Unlimited-Modus konfiguriert.

Verwenden von AWS-Managementkonsole , um eine T3-Burstable-Instance zu erstellen

In der AWS-Managementkonsole können Sie eine primäre DB-Cluster-Instance oder eine Read-Replica-Instance erstellen, die die Instance-Klasse verwendet, oder Sie können eine vorhandene `db.t3.medium` Instance so ändern, dass sie die Instance-Klasse verwendet. `db.t3.medium`

Beispielsweise erstellen Sie eine neue primäre DB-Cluster-Instance wie folgt über die Konsole:

- Wählen Sie Datenbank erstellen aus.
- Wählen Sie eine DB-Engine-Version gleich oder höher als `1.0.2.2` aus.
- Wählen Sie unter Zweck die Option Entwicklung und Testen aus.
- Akzeptieren Sie als DB-Instance-Klasse den Standardwert: `db.t3.medium — 2 vCPU, 4 GiB RAM`.

Verwenden von, AWS CLI um eine T3-Burstable-Instance zu erstellen

Sie können das auch verwenden AWS CLI , um dasselbe zu tun:

```
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
```

Ändern einer Neptune-DB-Instance (und sofortige Anwendung)

Sie können die meisten Änderungen sofort auf eine Amazon-Neptune-DB-Instance anwenden oder bis zum nächsten Wartungsfenster aufschieben. Einige Änderungen, wie Änderungen der Parametergruppe, erfordern einen manuellen Neustart Ihrer DB-Instance, damit die Änderungen wirksam werden.

> **Important**
>
> Änderungen führen zu einem Ausfall, wenn Neptune Ihre DB-Instance neu starten muss, damit die Änderungen wirksam werden. Überprüfen Sie die Auswirkungen auf Ihre Datenbank und Ihre Anwendungen, bevor Sie die Einstellungen für die DB-Instance ändern.

Häufige Einstellungen und Auswirkungen von Ausfallzeiten

Die folgende Tabelle beinhaltet Einzelheiten zu Einstellungen, die Sie ändern können, wann die Änderungen angewandt werden können und ob diese Änderungen eine Ausfallzeit für die DB-Instance verursachen.

DB-Instance-Einstellung	Hinweise zur Ausfallzeit	
DB-Instance-Klasse	Diese Änderung führt zu einem Ausfall, unabhängig davon, ob sie sofort oder während des nächsten Wartungszeitraums angewendet wird.	
DB-Instance-Kennung	Die DB-Instance wird neu gestartet und es kommt zu einem Ausfall während dieser Änderung, unabhängig davon, ob sie sofort oder während des nächsten Wartungszeitraums angewendet wird.	

DB-Instance-Einstellung	Hinweise zur Ausfallzeit	
Subnetzgruppe	Die DB-Instance wird neu gestartet und es kommt zu einem Ausfall während dieser Änderung, unabhängig davon, ob sie sofort oder während des nächsten Wartungsz eitraums angewendet wird.	
Sicherheitsgruppe	Die Änderung wird so schnell wie möglich asynchron angewendet, unabhängig von dem Zeitpunkt, den Sie für die Änderung festlegen, und es kommt zu keinem Ausfall.	–
Zertifizierungsstelle	Die DB-Instance wird standardmäßig neu gestartet , wenn Sie eine neue Zertifizi erungsstelle zuweisen.	
Database Port (Datenbankport	Die Änderung wird stets sofort ausgeführt. Daher wird die DB-Instance neu gestartet und es kommt zu einem Ausfall.	

DB-Instance-Einstellung	Hinweise zur Ausfallzeit
DB-Parametergruppe	Das Ändern dieser Einstellung führt nicht zu einem Nutzungsausfall. Der Name der Parametergruppe wird zwar direkt geändert, aber die tatsächlichen Parameteränderungen werden erst angewendet, wenn Sie die Instance ohne Failover neu starten. In diesem Fall wird die DB-Instance nicht automatisch neu gestartet und die Parameteränderungen werden während des nächsten Wartungsfensters nicht übernommen. Wenn Sie jedoch dynamische Parameter in der neu zugeordneten DB-Parametergruppe ändern, werden diese Änderungen sofort ohne Neustart angewendet. Weitere Informationen finden Sie unter Neustarten einer DB-Instance in Amazon Neptune.
DB-Cluster-Parametergruppe	Der Name der DB-Parametergruppe wird sofort geändert.

DB-Instance-Einstellung	Hinweise zur Ausfallzeit
Aufbewahrungszeitraum für Backups	Wenn Sie angeben, dass Änderungen sofort ausgeführt werden sollen, wird diese Änderung sofort wirksam. Wenn Sie die Einstellung von einem Nicht-Null-Wert in einen anderen Nicht-Null-Wert ändern, wird die Änderung sobald wie möglich asynchron angewendet. Alle anderen Änderungen werden während des nächsten Wartungsfensters ausgeführt. Wenn Sie den Wert von Null in einen Nicht-Null-Wert ändern oder umgekehrt, verursachen Sie einen Ausfall.
Prüfungsprotokoll	Wählen Sie Auditprotokoll, wenn Sie die Auditprotokollierung über CloudWatch Logs verwenden möchten. Sie müssen auch den Parameter `neptune_enable_audit_log` in der DB-Cluster-Parametergruppe auf `enable` (1) festlegen, um die Prüfprotokollierung zu aktivieren.

DB-Instance-Einstellung	Hinweise zur Ausfallzeit	
Automatische Nebenversions-Updates	Wählen Sie Automatische Nebenversions-Updates aus, wenn Ihr Neptune-DB-Cluster Nebenversions-Updates automatisch erhalten soll, sobald sie verfügbar sind. Die Option Automatische Nebenversions-Updates kann nur auf Nebenversions-Updates der Engine Ihres Amazon-Neptune-DB-Clusters angewendet werden. Sie kann nicht auf regelmäßige Patches zur Aufrechterhaltung der Systemstabilität angewendet werden.	

Umbenennen einer Neptune-DB-Instance

Sie können eine Amazon-Neptune-DB-Instance über die AWS-Managementkonsole umbenennen. Das Umbenennen einer DB-Instance kann weitgehende Auswirkungen haben. Im Folgenden finden Sie wissenswerte Punkte, die Sie beim Umbenennen einer DB-Instance berücksichtigen sollten.

- Wenn Sie eine DB-Instance umbenennen, ändert sich der Endpunkt der DB-Instance, da die URL den der DB-Instance zugewiesenen Namen beinhaltet. Sie sollten den Datenverkehr immer von der alten URL auf die neue umleiten.
- Wenn Sie eine DB-Instance umbenennen, wird der alte DNS-Name der DB-Instance sofort gelöscht (obwohl er noch einige Minuten im Cache verbleiben könnte). Der neue DNS-Name für die umbenannte DB-Instance wird nach etwa 10 Minuten übernommen. Die umbenannte DB-Instance ist erst verfügbar, wenn der neue Name übernommen wurde.
- Sie können keinen bestehenden DB-Instance-Namen verwenden, wenn Sie eine Instance umbenennen.
- Alle Read Replicas, die der DB-Instance zugeordnet sind, bleiben auch nach der Umbenennung zugeordnet. Angenommen, Sie nutzen eine DB-Instance als Produktionsdatenbank und haben der Instance mehrere Read Replicas zugeordnet. Wenn Sie die DB-Instance umbenennen und diese anschließend in der Produktionsumgebung durch einen DB-Snapshot ersetzen, bleiben die Read Replicas der umbenannten DB-Instance weiterhin zugeordnet.
- Metriken und Ereignisse, die dem Namen der DB-Instance zugeordnet sind, bleiben erhalten, wenn Sie einen DB-Instance-Namen erneut verwenden. Wenn Sie beispielsweise ein Lesereplikat heraufstufen und es in den Namen der vorherigen primären Instance umbenennen, werden die der primären Instance zugeordneten Ereignisse und Metriken der umbenannten Instance zugeordnet.
- DB-Instance-Tags für die DB-Instance bleiben erhalten, ungeachtet einer Umbenennung.
- DB-Snapshots werden für eine umbenannte DB-Instance aufbewahrt.

Umbenennen einer DB-Instance über die Neptune-Konsole

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die Amazon Neptune Neptune-Konsole zu Hausehttps://console.aws.amazon.com/neptune/.
2. Wählen Sie im Navigationsbereich Datenbanken aus.
3. Wählen Sie das Optionsfeld neben der DB-Instance aus, die Sie umbenennen möchten.
4. Wählen Sie im Menü Instance actions (Instance-Aktionen) die Option Modify (Ändern).

5. Geben Sie im Textfeld DB-Instance-Kennung einen neuen Namen ein. Wählen Sie Apply immediately (Sofort anwenden) und anschließend Continue (Weiter) aus.
6. Klicken Sie auf Modify DB instance (DB-Instance ändern), um die Änderungen abzuschließen.

Neustarten einer DB-Instance in Amazon Neptune

In einigen Fällen ist ein Neustart der Instance erforderlich, damit Änderungen angewendet werden, z. B. wenn Sie eine Amazon-Neptune-DB-Instance, die mit einer Instance verknüpfte DB-Parametergruppe oder einen statischen DB-Parameter in einer von der Instance verwendeten Parametergruppe ändern.

Durch das Neustarten einer DB-Instance, wird der Datenbank-Engine-Dienst neugestartet. Bei einem Neustart der DB-Instance werden auch sämtliche noch ausstehende Änderungen für die DB-Parametergrupp übernommen. Das Neustarten einer DB-Instance bewirkt einen vorübergehenden Nutzungsausfall dieser Instance, wobei der Status für diese Instance auf rebooting gesetzt wird. Wenn die Neptune-Instance für die Multi-AZ-Bereitstellung konfiguriert ist, könnte der Neustart auch über ein Failover ausgeführt werden. Nach Abschluss des Neustarts wird ein Neptune-Ereignis erstellt.

Wenn Ihre DB-Instance eine Multi-AZ-Bereitstellung ist, können Sie durch Auswahl der Option Reboot (Neu starten) ein Failover aus einer Availability Zone in eine andere erzwingen. Wenn Sie ein Failover Ihrer DB-Instance erzwingen, wechselt Neptune automatisch zu einem Standby-Replikat in einer anderen Availability Zone. Anschließend wird ein Update des DNS-Datensatzes für die DB-Instance durchgeführt, damit dieser auf die Standby-DB-Instance hinweist. Als Folge davon müssen Sie alle bestehenden Verbindungen zur DB-Instance beenden und neu herstellen.

Die Option Reboot with failover (Neustart mit Failover) ist sinnvoll, wenn Sie zu Testzwecken einen Ausfall der DB-Instance simulieren oder nach einem Ausfall Vorgänge auf der ursprünglichen Availability Zone wiederherstellen möchten. Weitere Informationen finden Sie unter Hochverfügbarkeit (Multi-AZ) im Amazon-RDS-Benutzerhandbuch. Wenn Sie einen DB-Cluster neu starten, erfolgt ein Failover auf das Standby-Replica. Das Neustarten eines Neptune-Replikats leitet kein Failover ein.

Die für den Neustart erforderliche Zeit ist eine Funktion der Absturzwiederherstellung. Wir empfehlen, dass Sie die Datenbank-Aktivitäten während des Neustartvorgangs möglichst gering halten, um Rollback-Aktivität für laufende Transaktionen zu reduzieren und somit die Dauer des Neustart zu verbessern.

In der Konsole kann die Option Neu starten deaktiviert werden, wenn eine DB-Instance nicht den Status Verfügbar hat. Dies kann mehrere Gründe haben, wie zum Beispiel eine laufende Sicherung, eine vom Kunden angeforderte Änderung oder eine Maßnahme im Wartungsfenster.

> **Note**
>
> Vor Release: 1.2.0.0 (21.07.2022) wurden bei einem Neustart der primären Instance (Writer-Instance) alle Read-Replicas in einem DB-Cluster automatisch neu gestartet.
> Ab Release: 1.2.0.0 (21.07.2022) führt ein Neustart der primären Instance nicht zum Neustart von Replikaten. Wenn Sie einen Cluster-Parameter ändern, müssen Sie daher jede Instance getrennt neu starten, um die Parameteränderung zu übernehmen (siehe Parametergruppen).

Umbenennen einer DB-Instance über die Neptune-Konsole

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die Amazon Neptune Neptune-Konsole zu Hausehttps://console.aws.amazon.com/neptune/.
2. Wählen Sie im Navigationsbereich Datenbanken aus.
3. Wählen Sie die DB-Instance aus, die Sie neu starten möchten.
4. Wählen Sie Instance actions (Instance-Aktionen) und anschließend Reboot (Neu starten).
5. Um ein Failover von einer Availability Zone zur anderen zu erzwingen, markieren Sie das Kontrollkästchen Reboot with failover? (Neustart mit Failover?) im Dialogfeld Reboot DB Instance (DB-Instance neu starten).
6. Wählen Sie Reboot. Um den Neustart abzubrechen, klicken Sie auf Cancel (Abbrechen).

Löschen einer DB-Instance in Amazon Neptune

Sie können eine Amazon Neptune DB-Instance in jedem Status und zu jeder Zeit löschen, sofern die Instance gestartet wurde.

> **Warning**
>
> Wenn Sie die letzte verbleibende Instance in einem Cluster mithilfe der Webkonsole löschen, wird auch das zugrunde liegende Cluster-Speichervolume gelöscht.

Erstellen eines abschließenden Snapshots Ihrer DB-Instance vor dem Löschen

Um eine DB-Instance zu löschen, geben Sie den Instance-Namen an und legen fest, ob für diese Instance ein abschließender DB-Snapshot erstellt werden soll. Wenn die zu löschende DB-Instance den Status Wird erstellt hat, ist ein abschließender DB-Snapshot nicht möglich. Wenn sich die DB-Instance in einem Fehlerzustand mit dem Status failed, incompatible-restore oder incompatible-network befindet, können Sie die Instance nur löschen, wenn der Parameter `SkipFinalSnapshot` auf `true` gesetzt ist.

Wenn Sie alle Neptune-DB-Instances in einem DB-Cluster mithilfe von löschen AWS-Managementkonsole, wird der gesamte DB-Cluster automatisch gelöscht. Wenn Sie das AWS CLI oder SDK verwenden, müssen Sie den DB-Cluster manuell löschen, nachdem Sie die letzte Instance gelöscht haben.

> **Important**
>
> Wenn Sie einen vollständigen DB-Cluster löschen, werden auch alle automatisierten Sicherungen gelöscht und können nicht wiederhergestellt werden. Daher können Sie die DB-Instance später nicht zum abschließenden Zustand wiederherstellen, es sei denn, Sie erstellen manuell einen abschließenden DB-Snapshot. Manuelle Snapshots einer Instance werden nicht gelöscht, wenn der Cluster gelöscht wird.

Sofern für die zu löschende DB-Instance ein Read Replica vorhanden ist, müssen Sie das Read Replica entweder hochstufen oder löschen.

In den folgenden Beispielen, löschen Sie eine DB-Instance mit und ohne einen abschließenden DB-Snapshot.

Löschen einer DB-Instance ohne abschließenden Snapshot

Um eine DB-Instance schnell zu entfernen, können Sie die Erstellung eines abschließenden DB-Snapshots überspringen. Wenn Sie eine DB-Instance löschen, werden alle automatischen Backups gelöscht und können nicht mehr wiederhergestellt werden. Manuelle Snapshots werden nicht gelöscht.

Löschen einer DB-Instance über die Neptune-Konsole ohne abschließenden DB-Snapshot

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die Amazon Neptune Neptune-Konsole zu Hausehttps://console.aws.amazon.com/neptune/.
2. Wählen Sie im Navigationsbereich Datenbanken aus.
3. Aktivieren Sie in der Liste Instances das Optionsfeld neben der zu löschenden DB-Instance.
4. Wählen Sie Instance-Aktionen und dann Löschen aus.
5. Wählen Sie im Feld Abschließenden Snapshot erstellen? die Option Nein aus.
6. Wählen Sie Löschen aus.

Löschen einer DB-Instance mit abschließendem Snapshot

Wenn eine gelöschte DB-Instance später wiederhergestellt werden soll, können Sie einen abschließenden DB-Snapshot erstellen. Alle automatisch erstellten Sicherungen werden ebenfalls gelöscht und können nicht wiederhergestellt werden. Manuelle Snapshots werden nicht gelöscht.

Löschen einer DB-Instance über die Neptune-Konsole mit abschließendem DB-Snapshot

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die Amazon Neptune Neptune-Konsole zu Hausehttps://console.aws.amazon.com/neptune/.
2. Wählen Sie im Navigationsbereich Datenbanken aus.
3. Aktivieren Sie in der Liste Instances das Optionsfeld neben der zu löschenden DB-Instance.
4. Wählen Sie Instance-Aktionen und dann Löschen aus.
5. Wählen Sie im Feld Abschließenden Snapshot erstellen? die Option Ja aus.
6. Geben Sie im Feld Name des abschließenden Snapshots den Namen des abschließenden DB-Snapshots ein.
7. Wählen Sie Löschen aus.

Sie können den Zustand einer Instance überprüfen, ermitteln, um welche Art von Instance es sich handelt, herausfinden, welche Engine-Version Sie derzeit installiert haben, und andere Informationen zu einer Instance mithilfe der Instance-Status-APIabrufen.

Erfassen von Diagrammänderungen in Echtzeit mit Neptune-Streams

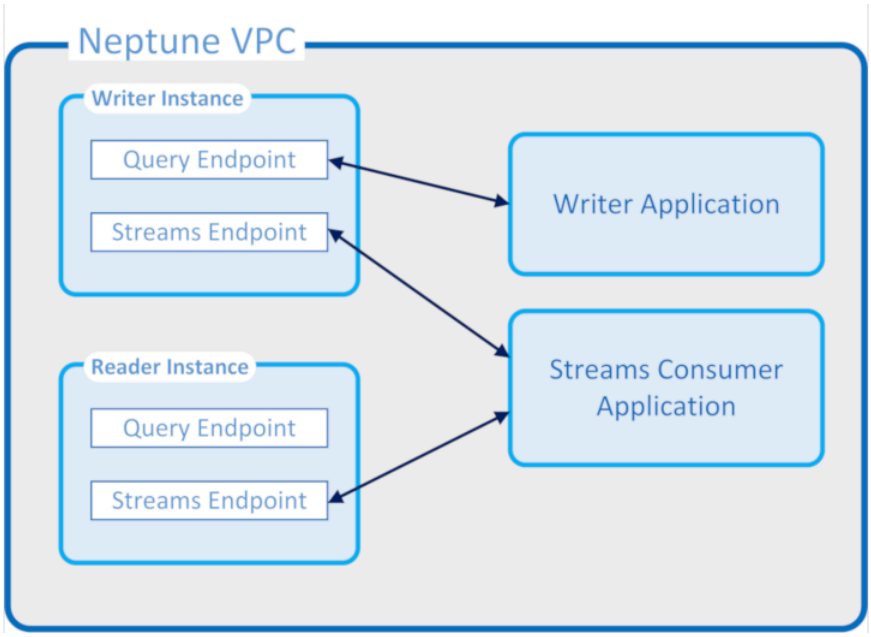
Neptune-Streams protokolliert vollständig verwaltet jede Änderung für Ihr Diagramm, sobald sie erfolgt, und in der Reihenfolge, in der sie erfolgt. Nach der Aktivierung von Streams übernimmt Neptune die Verantwortung für Verfügbarkeit, Sicherung, Sicherheit und Ablauf.

Im Folgenden finden Sie einige der zahlreichen Anwendungsfälle, in denen Sie Änderungen an einem Diagramm erfassen können, sobald sie auftreten:

- Möglicherweise möchten Sie, dass Ihre Anwendung Personen automatisch benachrichtigt, wenn bestimmte Änderungen vorgenommen werden.
- Möglicherweise möchten Sie eine aktuelle Version Ihrer Grafikdaten auch in einem anderen Datenspeicher wie Amazon OpenSearch Service, Amazon oder Amazon ElastiCache Simple Storage Service (Amazon S3) verwalten.

Neptune verwendet denselben nativen Speicher für den Änderungsprotokoll-Stream wie für Diagrammdaten. Es schreibt Änderungsprotokolleinträge synchron mit der Transaktion, die diese Änderungen vornimmt. Sie rufen diese Änderungsdatensätze mithilfe einer HTTP-REST-API aus dem Protokoll-Stream ab. (Informationen finden Sie unter Aufrufen der Streams-API.)

Das folgende Diagramm zeigt, wie Änderungsprotokolldaten aus Neptune-Streams abgerufen werden können.



Neptune-Streams-Garantien

- Änderungen, die von einer Transaktion vorgenommen werden, sind sofort für das Lesen von Writern und Readern verfügbar, sobald die Transaktion abgeschlossen ist (abgesehen von normalen Replikationsverzögerungen bei Readern).
- Änderungsdatensätze werden streng sequenziell in der Reihenfolge angezeigt, in der sie aufgetreten sind (einschließlich der Änderungen, die innerhalb einer Transaktion vorgenommen wurden).
- Die Änderungsstreams enthalten keine Duplikate. Jede Änderung wird nur einmal protokolliert.
- Die Änderungsstreams sind vollständig. Es gehen keine Änderungen verloren oder werden ausgelassen.
- Die Änderungsstreams enthalten alle Informationen, die erforderlich sind, um den vollständigen Status der Datenbank selbst zu einem beliebigen Zeitpunkt zu bestimmen, sofern der Startstatus bekannt ist.
- Streams kann jederzeit aktiviert oder deaktiviert werden.

Operative Eigenschaften von Neptune-Streams

- Der Änderungsprotokoll-Stream ist vollständig verwaltet.
- Change-log Daten werden synchron als Teil derselben Transaktion geschrieben, die eine Änderung vornimmt.

- Wenn Neptune Streams aktiviert sind, fallen für die I/O Änderungsprotokolldaten Speichergebühren an.
- Änderungsdatensätze werden automatisch eine Woche nach ihrer Erstellung gelöscht. Ab Engine-Version 1.2.0.0 kann dieser Aufbewahrungszeitraum mithilfe des DB-Cluster-Parameters neptune_streams_expiry_days in jede Anzahl von Tagen zwischen 1 und 90 geändert werden.
- Die Leseleistung der Streams wird mit Instances skaliert.
- Mit Read Replicas können Sie hohe Verfügbarkeit und einen hohen Lesedurchsatz erzielen. Es gibt keine Beschränkung für die Anzahl der Stream-Reader, die Sie gleichzeitig erstellen und verwenden können.
- Change-log Daten werden über mehrere Availability Zones hinweg repliziert, wodurch sie äußerst robust sind.
- Die Protokolldaten sind so sicher wie Ihre Diagrammdaten selbst. Sie können während der Übertragung und im Ruhezustand verschlüsselt werden. Der Zugriff kann mit IAM, Amazon VPC und AWS Key Management Service ()AWS KMS gesteuert werden. Wie die Diagrammdaten können sie gesichert und später mit Point-in-Time-Wiederherstellungen (PITR) wiederhergestellt werden.
- Das synchrone Schreiben von Stream-Daten als Teil jeder Transaktion führt zu einer leichten Herabsetzung der allgemeinen Schreibleistung.
- Stream-Daten sind nicht in Shards organisiert, da Neptune nur ein Shard zulässt.
- Die Protokoll-Stream-API `GetRecords` verwendet dieselben Ressourcen wie alle anderen Neptune-Diagrammoperationen. Dies bedeutet, dass Clients einen Lastausgleich zwischen Stream-Anforderungen und anderen DB-Anforderungen durchführen müssen.
- Wenn Streams deaktiviert sind, sind alle Protokolldaten sofort unzugänglich. Dies bedeutet, dass Sie alle Protokolldaten lesen müssen, die für Sie wichtig sind, bevor Sie die Protokollierung deaktivieren.
- Derzeit gibt es keine native Integration mit. AWS Lambda Der Protokoll-Stream generiert kein Ereignis, das eine Lambda-Funktion auslösen kann.

Topics

- Verwenden von Neptune-Streams
- Serialisierungsformate in Neptune-Streams
- Beispiele für Neptune-Streams

- Verwenden AWS CloudFormation um die Neptune-to-Neptune Replikation mit der Streams-Consumer-Anwendung einzurichten
- Verwenden der regionsübergreifenden Neptune-Streams-Replikation zur Notfallwiederherstellung

Verwenden von Neptune-Streams

Mit dem Neptune-Streams-Feature können Sie eine vollständige Abfolge von Änderungsprotokolleinträgen generieren, die jede Änderung an Ihren Diagrammdaten aufzeichnen, sobald sie erfolgt. Eine Übersicht über diese Funktion finden Sie unter Erfassen von Diagrammänderungen in Echtzeit mit Neptune-Streams.

Themen

- Verwenden von Neptune-Streams
- Deaktivieren von Neptune-Streams
- Aufrufen der Neptune-Streams-REST-API
- Neptune-Streams-API-Antwortformat
- Neptun-Stream-Daten lesen
- Neptune-Streams-API-Ausnahmen

Verwenden von Neptune-Streams

Sie können Neptune-Streams jederzeit aktivieren oder deaktivieren, indem Sie den `neptune_streams`-DB-Cluster-Parameter festlegen. Wenn Sie den Parameter auf 1 setzen, wird Streams aktiviert, und wenn Sie ihn auf `0` setzen, wird Streams deaktiviert.

> **Note**
>
> Nach dem Ändern des `neptune_streams`-DB-Cluster-Parameters müssen Sie alle DB-Instances im Cluster neu starten, damit die Änderungen wirksam werden.

Mittels des DB-Cluster-Parameters neptune_streams_expiry_days können Sie festlegen, wie viele Tage (von 1 bis 90) Stream-Datensätze auf dem Server gespeichert werden, bevor sie gelöscht werden. Der Standardwert ist 7.

Neptune Streams wurde ursprünglich als experimentelles Feature eingeführt, das Sie im Labormodus mittels des DB-Cluster-Parameters `neptune_lab_mode` aktivieren oder deaktivieren konnten (siehe Neptune-Labor-Modus). Die Verwendung des Labormodus zum Aktivieren von Streams ist jetzt veraltet und wird in Zukunft deaktiviert.

Deaktivieren von Neptune-Streams

Sie können Neptune-Streams jederzeit während der Ausführung deaktivieren.

Um Streams zu deaktivieren, aktualisieren Sie die DB-Cluster-Parametergruppe so, dass der Wert des Parameters `neptune_streams` auf 0 gesetzt wird.

> **Important**
>
> Sobald Streams deaktiviert ist, können Sie nicht mehr auf die Änderungsprotokolldaten zugreifen. Lesen Sie unbedingt, was für Sie wichtig ist, bevor Sie Streams deaktivieren.

Aufrufen der Neptune-Streams-REST-API

Sie greifen auf Neptune-Streams über eine REST-API zu, die eine HTTP-GET-Anforderung an einen der folgenden lokalen Endpunkte sendet:

- Für eine SPARQL-Diagramm-DB: `https://Neptune-DNS:8182/sparql/stream`.
- Für eine Gremlin- oder openCypher-Diagramm-DB: `https://Neptune-DNS:8182/propertygraph/stream` oder `https://Neptune-DNS:8182/pg/stream`.

> **Note**
>
> Der Gremlin-Stream-Endpunkt (`https://Neptune-DNS:8182/gremlin/stream`) ist zusammen mit dem zugehörigen Ausgabeformat () veraltet. `GREMLIN_JSON` Aus Gründen der Abwärtskompatibilität wird er weiter unterstützt, wird jedoch in zukünftigen Versionen möglicherweise entfernt.

Es ist nur eine HTTP-`GET`-Operation zulässig.

Neptune unterstützt die `gzip`-Komprimierung der Antwort, vorausgesetzt, dass die HTTP-Anforderung einen `Accept-Encoding`-Header enthält, der `gzip` als akzeptiertes Komprimierungsformat angibt (d. h. `"Accept-Encoding: gzip"`).

Parameters

- `limit` – long, optional. Reichweite: 1–100 000. Standard: 10.

  Gibt die maximale Anzahl der zurückzugebenden Datensätze an. Es gibt auch eine Größenbeschränkung von 10 MB für die Antwort, die nicht geändert werden kann und Vorrang vor der Anzahl der Datensätze hat, die im `limit`-Parameter angegeben ist. Die Antwort enthält einen Schwellenwertüberschreitungsdatensatz, wenn das 10 MB-Limit erreicht wurde.

- `iteratorType` – String, optional.

  Folgende Parameterwerte sind möglich:

  - `AT_SEQUENCE_NUMBER`(Standard) – Gibt an, dass der Lesevorgang ab der Ereignissequenznummer beginnen soll, die von beiden Parametern `commitNum` und `opNum` angegeben wird.
  - `AFTER_SEQUENCE_NUMBER` – Gibt an, dass der Lesevorgang direkt nach der Ereignissequenznummer beginnen soll, die von beiden Parametern `commitNum` und `opNum` angegeben wird.
  - `TRIM_HORIZON` – Gibt an, dass der Lesevorgang mit dem letzten nicht gekürzten Datensatz im System beginnen soll. Dies ist der älteste nicht abgelaufene (noch nicht gelöschte) Datensatz im Änderungsprotokoll-Stream. Dieser Modus ist während des Anwendungsstarts nützlich, wenn Sie keine bestimmte Startsequenznummer haben.
  - `LATEST` – Gibt an, dass der Lesevorgang mit dem neuesten Datensatz im System beginnen soll. Dies ist der letzte nicht abgelaufene (noch nicht gelöschte) Datensatz im Änderungsprotokoll-Stream. Dies ist nützlich, wenn Datensätze aus den aktuellen Top-Streams gelesen werden müssen, ohne ältere Datensätze zu verarbeiten, z. B. bei einer Notfallwiederherstellung oder einem Upgrade ohne Ausfallzeiten. Beachten Sie, dass in diesem Modus höchstens ein Datensatz zurückgegeben wird.

- `commitNum` – long, erforderlich, wenn iteratorType `AT_SEQUENCE_NUMBER` oder `AFTER_SEQUENCE_NUMBER` ist.

  Die Commit-Nummer des Startdatensatzes, der aus dem Änderungsprotokoll-Stream gelesen werden soll.

Dieser Parameter wird ignoriert, wenn `iteratorType` den Wert `TRIM_HORIZON` oder `LATEST` hat.

- `opNum` – long, optional (Standardwert ist 1).

  Die Operationssequenznummer innerhalb des angegebenen Commitments, ab der in den Änderungsprotokoll-Streamdaten gelesen werden soll.

Operationen, die SPARQL-Diagrammdaten ändern, generieren in der Regel nur einen einzigen Änderungsdatensatz pro Operation. Operationen, die Gremlin-Diagrammdaten ändern, können jedoch mehrere Änderungsdatensätze pro Operation generieren, wie in den folgenden Beispielen dargestellt:

- `INSERT` – Ein Gremlin-Eckpunkt kann mehrere Bezeichnungen und ein Gremlin-Element kann mehrere Eigenschaften haben. Für jede Bezeichnung und Eigenschaft wird ein separater Änderungsdatensatz generiert, wenn ein Element eingefügt wird.
- `UPDATE` – Wenn eine Gremlin-Elementeigenschaft geändert wird, werden zwei Änderungsdatensätze generiert: einer zum Entfernen des vorherigen Werts und einer zum Einfügen des neuen Werts.
- `DELETE` – Für jede gelöschte Elementeigenschaft wird ein eigener Änderungsdatensatz generiert. Wenn beispielsweise eine Gremlin-Grenze mit Eigenschaften gelöscht wird, wird für jede der Eigenschaften ein Änderungsdatensatz generiert. Danach wird ein Änderungsdatensatz zum Löschen der Grenzbezeichnung generiert.

  Wenn ein Gremlin-Eckpunkt gelöscht wird, werden zuerst alle eingehenden und ausgehenden Grenzeigenschaften gelöscht, dann die Grenzbezeichnungen, dann die Eckpunkteigenschaften und schließlich die Eckpunktbezeichnungen. Jeder dieser Löschvorgänge generiert einen Änderungsdatensatz.

Neptune-Streams-API-Antwortformat

Eine Antwort auf eine Neptune-Streams-REST-API-Anforderung enthält die folgenden Felder:

- `lastEventId` – Sequenz-ID der letzten Änderung in der Stream-Antwort. Eine Ereignis-ID besteht aus zwei Feldern: Ein `commitNum` identifiziert eine Transaktion, die das Diagramm geändert hat, und ein `opNum` identifiziert eine bestimmte Operation innerhalb dieser Transaktion. Dies wird im folgenden Beispiel veranschaulicht.

```
  "eventId": {
    "commitNum": 12,
    "opNum": 1
  }
```

- `lastTrxTimestamp` – Der Zeitpunkt, zu dem das Commit für die Transaktion angefordert wurde, in Millisekunden ab der Unix-Epoche.
- `format` – Serialisierungsformat für die zurückgegebenen Änderungsdatensätze. Die möglichen Werte sind `PG_JSON` für Gremlin- oder openCypher-Änderungsdatensätze und `NQUADS` für SPARQL-Änderungsdatensätze.
- `records` – Ein Array serialisierter Änderungsprotokoll-Stream-Datensätze, die in der Antwort enthalten sind. Jeder Datensatz im `records`-Array enthält die folgenden Felder:
  - `commitTimestamp` – Der Zeitpunkt, zu dem das Commit für die Transaktion angefordert wurde, in Millisekunden ab der Unix-Epoche.
  - `eventId` – Die Sequenz-ID des Stream-Änderungsdatensatzes.
  - `data`— Der serialisierte Gremlin-, SPARQL- oder Change-Record. OpenCypher Die Serialisierungsformate für jeden Datensatz werden im nächsten Abschnitt (Serialisierungsformate in Neptune-Streams) ausführlicher beschrieben.
  - `op` – Die Operation, die die Änderung erstellt hat. Folgende sind gültige Werte:
    - `ADD`— Zeigt an, dass dem Diagramm ein neues Element (z. B. ein Scheitelpunkt, eine Kante, eine Eigenschaft oder ein Tripel) hinzugefügt wurde.
    - `REMOVE`— Zeigt an, dass ein vorhandenes Element aus dem Diagramm entfernt wurde.

    Neptune drückt eine Änderung an einem vorhandenen Element als Datensatz aus, gefolgt von einem `REMOVE ADD` Datensatz.
  - `isLastOp` – Nur vorhanden, wenn diese Operation die letzte in ihrer Transaktion ist. Wenn vorhanden, ist sie auf `true` festgelegt. Nützlich, um sicherzustellen, dass die gesamte Transaktion genutzt wird.
- `totalRecords` – Die Gesamtanzahl der Datensätze in der Antwort.

Die folgende Antwort gibt beispielsweise Gremlin-Änderungsdaten für eine Transaktion zurück, die mehr als eine Operation enthält:

```
{
  "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
}
```

Die folgende Antwort gibt SPARQL-Änderungsdaten für die letzte Operation in einer Transaktion zurück (die Operation, die `EventId(97, 1)` in Transaktionsnummer 97 identifiziert hat).

```
{
  "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
}
```

Neptun-Stream-Daten lesen

Die folgenden Beispiele zeigen, wie Datensätze vom Stream-Endpunkt des Eigenschaftsdiagramms gelesen werden. Sie können das AWS CLI, das AWS SDK for Python (Boto3)awscurl, oder verwenden. curl

AWS CLI

```
aws neptunedata get-propertygraph-stream \
  --endpoint-url https://your-neptune-endpoint:port \
  --iterator-type TRIM_HORIZON \
  --limit 10
```

Verwenden Sie die Parameter und, um von einer bestimmten Position im Stream aus zu lesen: `--commit-num` `--iterator-type`

```
aws neptunedata get-propertygraph-stream \
  --endpoint-url https://your-neptune-endpoint:port \
  --limit 10 \
  --commit-num 1 \
  --iterator-type AT_SEQUENCE_NUMBER
```

Weitere Informationen finden Sie unter get-propertygraph-stream in der Befehlsreferenz. 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))
```

Um durch den Stream zu paginieren, verwenden Sie in der Antwort das `commitNum` und opNum aus dem: `lastEventId`

```
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**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

Weitere Informationen zur Verwendung awscurl mit der IAM-Authentifizierung finden Sie unter. [Verwenden von `awscurl` mit temporären Anmeldeinformationen, um eine sichere Verbindung zu einem DB-Cluster mit aktivierter IAM-Authentifizierung herzustellen]

curl

```
curl https://your-neptune-endpoint:port/propertygraph/stream?limit=10
```

> **Note**
>
> Die Verwendung curl ohne Sigv4-Signatur funktioniert nur, wenn die IAM-Authentifizierung auf Ihrem Neptune-Cluster deaktiviert ist. Wenn die IAM-Authentifizierung aktiviert ist, verwenden Sie stattdessen oder. awscurl AWS CLI

Ersetzen Sie SPARQL-Diagramme `/propertygraph/stream` durch `/sparql/stream` in den vorherigen Beispielen. Wenn Sie das SDK AWS CLI oder verwenden, verwenden Sie `client.get_sparql_stream()` stattdessen `get-sparql-stream` or.

Stream-Daten in einem Graph-Notizbuch anzeigen

Wenn Sie Neptune Graph Notebooks verwenden, bietet `%stream_viewer` Line Magic eine visuelle Oberfläche für das interaktive Durchsuchen von Stream-Datensätzen, ohne Code schreiben zu müssen. Führen Sie Folgendes in einer Notebook-Zelle aus:

```
%stream_viewer
```

Sie können optional die Abfragesprache und die Seitengröße angeben:

```
%stream_viewer sparql --limit 20
```

Weitere Informationen finden Sie unter Die Zeilen-Magics `%stream_viewer`.

> **Note**
>
> The `%stream_viewer` Magic wird nur in den Engine-Versionen 1.0.5.1 und früher vollständig unterstützt.

Neptune-Streams-API-Ausnahmen

Die folgende Tabelle beschreibt Streams-Ausnahmen.

Fehlercode	HTTP-Code	Erneut versuchen?	Fehlermeldung
`InvalidParameterException`	400	Nein	Für den Eingabeparameter wurde ein ungültiger Wert oder ein Wert angegeben, der sich außerhalb des gültigen Bereichs befindet.
`ExpiredStreamException`	400	Nein	Alle angeforderten Datensätze überschreiten das maximal zulässige Alter und sind abgelaufen.
`ThrottlingException`	500	Ja	Die Anforderungsrate übersteigt den maximalen Durchsatz.
`StreamRecordsNotFoundException`	404	Nein	Die angeforderte Ressource wurde nicht gefunden. Der Stream ist möglicherweise nicht korrekt angegeben.
`MemoryLimitExceededException`	500	Ja	Die Anforderungsverarbeitung war nicht erfolgreich, weil nicht genügend Arbeitsspeicher vorhanden ist. Sie kann aber wiederholt werden, wenn der Server nicht mehr so stark ausgelastet ist.

Serialisierungsformate in Neptune-Streams

Amazon Neptune verwendet zwei verschiedene Formate für die Serialisierung von Diagrammänderungsdaten in Protokollstreams, je nachdem, ob das Diagramm mit Gremlin oder SPARQL erstellt wurde.

Beide Formate verwenden das gleiche Format für die Serialisierung von Datensätzen, wie in Neptune-Streams-API-Antwortformat beschrieben, das die folgenden Felder enthält:

- `commitTimestamp` – Der Zeitpunkt, zu dem das Commit für die Transaktion angefordert wurde, in Millisekunden ab der Unix-Epoche.
- `eventId` – Die Sequenz-ID des Stream-Änderungsdatensatzes.
- `data`— Der serialisierte Gremlin-, SPARQL- oder Change-Record. OpenCypher Die Serialisierungsformate für jeden Datensatz werden in den nächsten Abschnitten ausführlicher beschrieben.
- `op` – Die Operation, die die Änderung erstellt hat.

Themen

- PG_JSON-Änderungsserialisierungsformat
- SPARQL NQUADS-Änderungsserialisierungsformat

PG_JSON-Änderungsserialisierungsformat

> **Note**
>
> Das vom Gremlin-Stream-Endpunkt (`GREMLIN_JSON`) ausgegebene Gremlin-Stream-Ausgabeformat () ist veraltet. `https://Neptune-DNS:8182/gremlin/stream` Es wird durch PG_JSON ersetzt, das zurzeit mit `GREMLIN_JSON` identisch ist.

Ein Gremlin- oder openCypher-Änderungsdatensatz, der im Feld `data` einer Protokoll-Stream-Antwort enthalten ist, hat die folgenden Felder:

- `id` – Zeichenfolge, erforderlich.

  Die ID des Gremlin- oder openCypher-Elements.

- `type` – Zeichenfolge, erforderlich.

  Der Typ dieses Gremlin- oder openCypher-Elements. Muss einer der folgenden sein:

  - `vl` – Eckpunktbezeichnung für Gremlin; Knotenbezeichnung für openCypher.
  - `vp` – Eckpunkteigenschaften für Gremlin; Knoteneigenschaften für openCypher.
  - `e` – Kante und Kantenbezeichnung für Gremlin; Beziehung und Beziehungstyp für openCypher.
  - `ep` – Kanteneigenschaften für Gremlin; Beziehungseigenschaften für openCypher.
- `key` – Zeichenfolge, erforderlich.

  Der Name der Eigenschaft. Für Elementbezeichnungen ist dies „label“.
- `value` – `value`-Objekt, erforderlich.

  Dies ist ein JSON-Objekt, das ein `value`-Feld für den Wert selbst und ein `datatype`-Feld für den JSON-Datentyp dieses Werts enthält.

```
  "value": {
    "value": "the new value",
    "dataType": "the JSON datatype of the new value"
  }
```

- `from` – Zeichenfolge, optional.

  Wenn dies eine Kante ist (Typ = „e“), die ID des entsprechenden Eckpunkts oder Quellknotens Von.
- `to` – Zeichenfolge, optional.

  Wenn dies eine Kante ist (Typ = „e“), die ID des entsprechenden Eckpunkts oder Zielknotens Bis.

Gremlin-Beispiele

- Das folgende Beispiel zeigt eine Gremlin-Eckpunktbezeichnung.

```
{
  "id": "an ID string",
  "type": "vl",
  "key": "label",
  "value": {
    "value": "the new value of the vertex label",
    "dataType": "String"
```

```
  }
}
```

- Das folgende Beispiel zeigt eine Gremlin-Eckpunkteigenschaft.

```
{
  "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"
  }
}
```

- Das folgende Beispiel zeigt eine Gremlin-Kante.

```
{
  "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"
}
```

openCypher-Beispiele

- Das folgende Beispiel zeigt eine openCypher-Knotenbezeichnung.

```
{
  "id": "an ID string",
  "type": "vl",
  "key": "label",
  "value": {
    "value": "the new value of the node label",
    "dataType": "String"
  }
```

```
}
```

- Das folgende Beispiel zeigt eine openCypher-Knoteneigenschaft.

```
{
  "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"
  }
}
```

- Das folgende Beispiel zeigt eine openCypher-Beziehung.

```
{
  "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"
}
```

SPARQL NQUADS-Änderungsserialisierungsformat

Neptune protokolliert Änderungen an SPARQL-Quads im Diagramm mithilfe der Resource Description Framework (RDF) `N-QUADS` -Sprache, die in der W3C RDF 1.1-Spezifikation definiert ist. N-Quads

Das `data` Feld im Änderungsdatensatz enthält einfach ein `stmt` Feld, das eine N-QUADS Anweisung enthält, die das geänderte Quadrat ausdrückt, wie im folgenden Beispiel.

```
  "stmt" : "<https://test.com/s> <https://test.com/p> <https://test.com/o> .\n"
```

Beispiele für Neptune-Streams

Die folgenden Beispiele zeigen, wie Sie auf Änderungsprotokoll-Stream-Daten in Amazon Neptune zugreifen.

Themen

- AT_SEQUENCE_NUMBER Änderungsprotokoll
- AFTER_SEQUENCE_NUMBER-Änderungsprotokoll
- TRIM_HORIZON Änderungsprotokoll
- AKTUELLES Änderungsprotokoll
- Komprimierungsänderungsprotokoll

AT_SEQUENCE_NUMBER Änderungsprotokoll

Das folgende Beispiel zeigt ein Gremlin- oder openCypher-AT_SEQUENCE_NUMBER-Änderungsprotokoll.

```
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
}
```

Dies ist ein SPARQL-Beispiel eines AT_SEQUENCE_NUMBER-Änderungsprotokolls.

```
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
}
```

AFTER_SEQUENCE_NUMBER-Änderungsprotokoll

Das folgende Beispiel zeigt ein Gremlin- oder openCypher-AFTER_SEQUENCE_NUMBER-Änderungsprotokoll.

```
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
}
```

TRIM_HORIZON Änderungsprotokoll

Das folgende Beispiel zeigt ein Gremlin- oder openCypher-TRIM_HORIZON-Änderungsprotokoll.

```
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
}
```

AKTUELLES Änderungsprotokoll

Das folgende Beispiel zeigt ein Gremlin- oder openCypher-LATEST-Änderungsprotokoll. Beachten Sie, dass die API-Parameter `limit`, `commitNum` und `opNum` vollständig optional sind.

```
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
}
```

Komprimierungsänderungsprotokoll

Das folgende Beispiel zeigt ein Gremlin- oder openCypher-Komprimierungsänderungsprotokoll.

```
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
}
```

Verwenden AWS CloudFormation um die Neptune-to-Neptune Replikation mit der Streams-Consumer-Anwendung einzurichten

Sie können eine CloudFormation Vorlage verwenden, um die Neptune Streams-Consumer-Anwendung zur Unterstützung Neptune-to-Neptune der Replikation einzurichten.

Themen

- Wählen Sie eine CloudFormation Vorlage für Ihre Region
- Hinzufügen von Details zu dem von Ihnen erstellten Neptune-Streams-Consumer-Stack
- Führen Sie den CloudFormation Vorlage
- Aktualisieren des Stream-Pollers mit den neuesten Lambda-Artefakten

Wählen Sie eine CloudFormation Vorlage für Ihre Region

Um den entsprechenden CloudFormation Stack auf der CloudFormation Konsole zu starten, wählen Sie je nach AWS Region, die Sie verwenden möchten, eine der Schaltflächen „Stack starten“ in der folgenden Tabelle.

Region	Anzeigen	Starten
USA Ost (Nord-Virginia)	Anzeigen	
USA Ost (Ohio)	Anzeigen	
USA West (Nordkalifornien)	Anzeigen	
USA West (Oregon)	Anzeigen	
Kanada (Zentral)	Anzeigen	
Südamerika (São Paulo)	Anzeigen	
Europa (Stockholm)	Anzeigen	
Europa (Irland)	Anzeigen	
Europa (London)	Anzeigen	
Europa (Paris)	Anzeigen	
Europa (Frankfurt)	Anzeigen	
Middle East (Bahrain)	Anzeigen	

Region	Anzeigen	Starten
Naher Osten (VAE)	Anzeigen	Launch Stack
Israel (Tel Aviv)	Anzeigen	Launch Stack
Afrika (Kapstadt)	Anzeigen	Launch Stack
Asien-Pazifik (Tokio)	Anzeigen	Launch Stack
Asien-Pazifik (Hongkong)	Anzeigen	Launch Stack
Asien-Pazifik (Seoul)	Anzeigen	Launch Stack
Asien-Pazifik (Singapur)	Anzeigen	Launch Stack
Asien-Pazifik (Sydney)	Anzeigen	Launch Stack
Asien-Pazifik (Mumbai)	Anzeigen	Launch Stack
China (Peking)	Anzeigen	Launch Stack
China (Ningxia)	Anzeigen	Launch Stack
AWS GovCloud (US-West)	Anzeigen	Launch Stack
AWS GovCloud (US-East)	Anzeigen	Launch Stack

Wählen Sie auf der Seite Create Stack (Stack erstellen) die Option Next (Weiter) aus.

Hinzufügen von Details zu dem von Ihnen erstellten Neptune-Streams-Consumer-Stack

Auf der Seite Stackdetails angeben finden Sie Eigenschaften und Parameter für die Einrichtung der Anwendung steuern können:

Stack-Name — Der Name des neuen CloudFormation Stacks, den Sie erstellen. Sie können in der Regel den Standardwert `NeptuneStreamPoller` verwenden.

Geben Sie unter Parameter, Folgendes ein:

Netzwerkkonfiguration für die VPC, in der der Streams-Consumer ausgeführt wird

- **`VPC`** – Geben Sie den Namen der VPC an, in der die abfragende Lambda-Funktion ausgeführt werden wird.
- **`SubnetIDs`** – Die Subnetze, für die eine Netzwerkschnittstelle eingerichtet ist. Fügen Sie Subnetze hinzu, die Ihrem Neptune-Cluster entsprechen.
- **`SecurityGroupIds`** – Geben Sie die IDs von Sicherheitsgruppen an, die Schreibzugriff für den eingehenden Datenverkehr in Ihrem Neptune-DB-Cluster gewähren.
- **`RouteTableIds`** – Dies wird benötigt, um einen Amazon-DynamoDB-Endpunkt in Ihrer Neptune-VPC zu erstellen, wenn noch keiner vorhanden ist. Sie müssen eine durch Komma getrennte Liste der Routing-Tabellen-IDs angeben, die den Subnetzen zugeordnet sind.
- **`CreateDDBVPCEndPoint`** – Ein boolescher Wert, der standardmäßig auf `true` festgelegt ist und angibt, ob ein Dynamo-DB-VPC-Endpunkt erstellt werden muss. Sie müssen ihn nur in `false` ändern, wenn Sie bereits einen DynamoDB-Endpunkt in Ihrer VPC erstellt haben.
- **`CreateMonitoringEndPoint`** – Ein boolescher Wert, der standardmäßig auf `true` festgelegt ist und angibt, ob ein überwachender VPC-Endpunkt erstellt werden muss. Sie müssen ihn nur in `false` ändern, wenn Sie bereits einen Überwachungsendpunkt in Ihrer VPC erstellt haben.

Stream-Poller

- **`ApplicationName`** – Sie können diesen Satz in der Regel beim Standardwert (`NeptuneStream`) belassen. Wenn Sie einen anderen Namen verwenden, muss dieser eindeutig sein.
- **`LambdaMemorySize`** – Wird verwendet, um die Speichergröße festzulegen, die für die Lambda-Poller-Funktion verfügbar ist. Der Standardwert ist 2.048 Megabyte.

- **`LambdaRuntime`** – Die Sprache, die in der Lambda-Funktion verwendet wird, mit der Elemente aus dem Neptune-Stream abgerufen werden. Sie können dies auf `python3.9` oder `java8` festlegen.
- **`LambdaS3Bucket`** – Der Amazon-S3-Bucket, der Lambda-Code-Artefakte enthält. Lassen Sie dieses Feld leer, es sei denn, Sie verwenden eine benutzerdefinierte Lambda-Polling-Funktion, die aus einem anderen Amazon-S3-Bucket geladen wird.
- **`LambdaS3Key`** – Der Amazon-S3-Schlüssel, der Ihren Lambda-Code-Artefakten entspricht. Lassen Sie dieses Feld leer, es sei denn, Sie verwenden eine benutzerdefinierte Lambda-Polling-Funktion.
- **`LambdaLoggingLevel`** – Belassen Sie diesen Satz grundsätzlich beim Standardwert, `INFO`.
- **`ManagedPolicies`** – Listet die verwalteten Richtlinien auf, die für die Ausführung Ihrer Lambda-Funktion verwendet werden sollen. Lassen Sie dieses Feld grundsätzlich leer, es sei denn, Sie verwenden eine benutzerdefinierte Lambda-Polling-Funktion.
- **`StreamRecordsHandler`** – Lassen Sie dieses Feld grundsätzlich leer, es sei denn, Sie verwenden einen benutzerdefinierten Handler für die Datensätze in Neptune-Streams.
- **`StreamRecordsBatchSize`** – Die maximale Anzahl von Datensätzen, die aus dem Stream abgerufen werden sollen. Sie können mit diesem Parameter die Leistung optimieren. Der Standardwert (`5000`) ist ein guter Anfangspunkt. Der maximal zulässige Wert ist 10.000. Je höher die Zahl ist, desto weniger Netzwerkaufrufe werden benötigt, um Datensätze aus dem Stream zu lesen. Allerdings ist umso mehr Speicher ist erforderlich, um die Datensätze zu verarbeiten. Niedrigere Werte dieses Parameters führen zu einem geringeren Durchsatz.
- **`MaxPollingWaitTime`** – Die maximale Wartezeit zwischen zwei Abfragen (in Sekunden). Legt die Häufigkeit fest, mit der der Lambda-Poller aufgerufen wird, um die Neptune-Streams abzufragen. Legen Sie diesen Wert für kontinuierliche Abfragen auf „0“ fest. Die Höchstwert beträgt 3.600 Sekunden (1 Stunde). Der Standardwert (60 Sekunden) ist ein guter Ausgangspunkt, je nachdem, wie schnell sich Ihre Graphdaten ändern.
- **`MaxPollingInterval`** – Die maximale kontinuierliche Polling-Dauer (in Sekunden). Mit dieser Einstellung können Sie ein Timeout für die Lambda-Polling-Funktion festlegen. Der Wert sollte im Bereich zwischen 5 und 900 Sekunden liegen. Der Standardwert (600 Sekunden) ist ein guter Ausgangspunkt.
- **`StepFunctionFallbackPeriod`**— Die Anzahl der Einheiten der Step-Function-Fallback-Periode, um auf den Poller zu warten. Danach wird die Step-Funktion über Amazon CloudWatch Events aufgerufen, um sich nach einem Ausfall zu erholen. Der Standardwert (5 Minuten) ist ein guter Anfangspunkt.

- **`StepFunctionFallbackPeriodUnit`** – Die Zeiteinheiten, die verwendet wurden, um den vorhergehenden Wert für `StepFunctionFallbackPeriodUnit` (`minutes`, `hours`, oder `days`) zu messen. Der Standardwert (`minutes`) ist im Allgemeinen ausreichend.
- **`StartingCheckpoint`**— Der Start-Checkpoint für den Stream-Poller. Die Standardeinstellung ist `0:0`, was bedeutet, dass mit dem Anfang des Neptune-Streams begonnen wird.
- **`StreamPollerInitialState`**— Der Ausgangszustand des Pollers. Die Standardeinstellung ist `ENABLED`, was bedeutet, dass die Stream-Replikation beginnt, sobald die gesamte Stack-Erstellung abgeschlossen ist.

Neptune-Stream

- **`NeptuneStreamEndpoint`** – (Erforderlich) Der Endpunkt des Neptun-Quellstreams. Dies erfolgt auf eine der folgenden beiden Arten:
  - **`https://`*`your DB cluster`*`:`*`port`*`/propertygraph/stream`** (oder dessen Alias, `https://`*`your DB cluster`*`:`*`port`*`/pg/stream`).
  - **`https://`*`your DB cluster`*`:`*`port`*`/sparql/stream`**.
- **`Neptune Query Engine`** – Wählen Sie Gremlin, openCypher oder SPARQL aus.
- **`IAMAuthEnabledOnSourceStream`** – Wenn Ihr Neptune-DB-Cluster die IAM-Authentifizierung verwendet, legen Sie diesen Parameter auf `true` fest.
- **`StreamDBClusterResourceId`** – Wenn Ihr Neptune-DB-Cluster die IAM-Authentifizierung verwendet, legen Sie diesen Parameter auf die Cluster-Ressourcen-ID fest. Die Ressourcen-ID ist nicht mit der Cluster-ID identisch. Stattdessen hat sie das folgende Format: `cluster-` gefolgt von 28 alphanumerischen Zeichen. Sie finden sie unter Cluster-Details in der Neptune-Konsole.

Neptune-DB-Ziel-Cluster

- **`TargetNeptuneClusterEndpoint`** – Der Cluster-Endpunkt (nur Hostname) des Ziel-Clusters für die Sicherung.

  Beachten Sie, dass Sie nicht auch `TargetSPARQLUpdateEndpoint` angeben können, wenn Sie `TargetNeptuneClusterEndpoint` angeben.
- **`TargetNeptuneClusterPort`** – Die Port-Nummer für den Ziel-Cluster.

  Beachten Sie, dass die Einstellung für `TargetNeptuneClusterPort` ignoriert wird, wenn Sie `TargetSPARQLUpdateEndpoint` angeben.

- **`IAMAuthEnabledOnTargetCluster`** – Wird auf true festgelegt, wenn die IAM-Authentifizierung für den Ziel-Cluster aktiviert werden soll.
- **`TargetAWSRegion`**— Die AWS Region des Ziel-Backup-Clusters, z. B.`us-east-1`). Sie müssen diesen Parameter nur angeben, wenn sich die AWS Region des Ziel-Backup-Clusters von der Region des Neptune-Quell-Clusters unterscheidet, wie im Fall einer regionsübergreifenden Replikation. Wenn Quell- und Zielregion identisch sind, ist dieser Parameter optional.

  Beachten Sie, dass der Vorgang fehlschlägt, wenn der `TargetAWSRegion` Wert keine gültige AWS Region ist, die Neptune unterstützt.
- **`TargetNeptuneDBClusterResourceId`** – Optional: Dies ist nur notwendig, wenn die IAM-Authentifizierung für den DB-Ziel-Cluster aktiviert ist. Auf die Ressourcen-ID des Ziel-Clusters festgelegt.
- **`SPARQLTripleOnlyMode`** – Boolesches Flag, das festlegt, ob der Nur-Tripel-Modus aktiviert ist. Im Nur-Tripel-Modus gibt es keine Replikation mit benannten Diagrammen. Der Standardwert ist `false`.
- **`TargetSPARQLUpdateEndpoint`** – URL des Zielendpunkts für das SPARQL-Update, z. B. `https://abc.com/xyz`. Dieser Endpunkt kann jeder SPARQL-Speicher sein, der Quads oder Tripel unterstützt.

  Beachten Sie, dass Sie nicht auch `TargetNeptuneClusterEndpoint` angeben können, wenn Sie `TargetSPARQLUpdateEndpoint` angeben, und dass die Einstellung für `TargetNeptuneClusterPort` ignoriert wird.
- **`BlockSparqlReplicationOnBlankNode`** — Boolesches Flag, das, wenn es auf gesetzt ist`true`, die Replikation von Daten BlankNode in SPARQL (RDF) stoppt. Der Standardwert ist `false`.

Alarm

- **`Required to create Cloud watch Alarm`**— Setzen Sie diesen Wert auf, `true` wenn Sie einen CloudWatch Alarm für den neuen Stack erstellen möchten.
- **`SNS Topic ARN for Cloudwatch Alarm Notifications`**— Das SNS-Thema ARN, an das CloudWatch Alarmbenachrichtigungen gesendet werden sollen (nur erforderlich, wenn Alarme aktiviert sind).
- **`Email for Alarm Notifications`** – Die E-Mail-Adresse, an die Alarmbenachrichtigungen gesendet werden sollen (nur notwendig, wenn Alarme aktiviert sind).

Als Ziel der Alarmbenachrichtigung können Sie nur SNS, nur E-Mail oder sowohl SNS als auch E-Mail hinzufügen.

Führen Sie den CloudFormation Vorlage

Jetzt können Sie den Prozess für die Bereitstellung einer Neptune-Streams-Consumer-Anwendungs-Instance wie folgt abschließen:

1. Wählen Sie auf der Seite „Stack-Details angeben“ die Option Weiter aus. CloudFormation
2. Wählen Sie auf der Seite Optionen Weiter aus.
3. Aktivieren Sie auf der Seite Überprüfen das erste Kontrollkästchen, um zu bestätigen, dass CloudFormation IAM-Ressourcen erstellt. Aktivieren Sie das zweite Kontrollkästchen, um `CAPABILITY_AUTO_EXPAND` für den neuen Stack zu bestätigen.

   > **Note**
   >
   > `CAPABILITY_AUTO_EXPAND` bestätigt explizit, dass Makros ohne vorherige Überprüfung beim Erstellen des Stacks erweitert werden. Benutzer erstellen häufig einen Änderungssatz aus einer verarbeiteten Vorlage, sodass die von Makros durchgeführten Änderungen überprüft werden können, ehe der Stack tatsächlich erstellt wird. Weitere Informationen zur CloudFormation CreateStackAPI finden Sie in der AWS CloudFormation API-Referenz.

   Wählen Sie die Option Erstellen aus.

Aktualisieren des Stream-Pollers mit den neuesten Lambda-Artefakten

Sie können den Stream-Poller wie folgt mit den neuesten Lambda-Code-Artefakten aktualisieren:

1. Navigieren Sie im AWS-Managementkonsole zum übergeordneten CloudFormation Hauptstapel CloudFormation und wählen Sie ihn aus.
2. Wählen Sie die Option Aktualisieren für den Stack aus.
3. Wählen Sie Aktuelle Vorlage ersetzen aus.
4. Wählen Sie als Vorlagenquelle Amazon-S3-URL aus und geben Sie die folgende S3-URL ein:

```
https://aws-neptune-customer-samples.s3.amazonaws.com/neptune-stream/
neptune_to_neptune.json
```

5. Wählen Sie Weiter aus, ohne die CloudFormation Parameter zu ändern.
6. Wählen Sie Stack aktualisieren aus.

Der Stack aktualisiert jetzt die Lambda-Artefakte mit den neuesten Artefakten.

Verwenden der regionsübergreifenden Neptune-Streams-Replikation zur Notfallwiederherstellung

Neptune stellt zwei Möglichkeiten für die Implementierung regionsübergreifender Failover-Funktionen bereit:

- Cross-region Snapshot kopieren und wiederherstellen
- Verwenden von Neptune-Streams zur Replikation von Daten zwischen zwei Clustern in zwei verschiedenen Regionen.

Cross-region Das Kopieren und Wiederherstellen von Snapshots hat den geringsten Betriebsaufwand für die Wiederherstellung eines Neptune-Clusters in einer anderen Region. Das Kopieren eines Snapshots zwischen Regionen kann jedoch eine erhebliche Datenübertragungszeit erfordern, da ein Snapshot eine vollständige Sicherung des Neptune-Clusters darstellt. Daher kann das regionsübergreifende Kopieren und Wiederherstellen von Snapshots für Szenarien verwendet werden, die lediglich ein Recovery Point Objective (RPO) von einigen Stunden und ein Recovery Time Objective (RTO) von einigen Stunden erfordern.

Ein Recovery Point Objective (RPO) wird anhand der Zeit zwischen Sicherungen gemessen. Es definiert, wie viele Daten zwischen dem Zeitpunkt der letzten Sicherung und dem Zeitpunkt, an dem die Datenbank wiederhergestellt wird, möglicherweise verloren gingen.

Ein Recovery Time Objective (RTO) wird anhand der Zeit gemessen, die für die Durchführung einer Wiederherstellung benötigt wird. Dies ist die Zeit, die der DB-Cluster benötigt, um nach einem Ausfall einen Failover zu einer wiederhergestellten Datenbank auszuführen.

Neptune-Streams ermöglichen die kontinuierliche Synchronisierung von Neptune-Sicherungs-Clustern mit dem primären Produktions-Cluster. Wenn ein Fehler auftritt, erfolgt ein Failover Ihrer

Datenbank zum Sicherungs-Cluster. Dies reduziert RPO und RTO auf Minuten, da Daten ständig zum Sicherungs-Cluster kopiert werden, der jederzeit sofort als Failover-Ziel verfügbar ist.

Der Nachteil dieser Verwendung von Neptune-Streams besteht darin, dass der operative Aufwand für die Wartung der Replikationskomponenten und die Kosten für die kontinuierliche Ausführung eines zweiten Neptune-DB-Clusters erheblich sein können.

Replikation einrichten Neptune-to-Neptune

Ihr primärer Produktions-DB-Cluster befindet sich in einer VPC in einer bestimmten Quellregion. Zur Notfallwiederherstellung müssen Sie vor allem drei Dinge in einer anderen Wiederherstellungsregion replizieren oder emulieren:

- Die im Cluster gespeicherten Daten.
- Die Konfiguration des primären Clusters. Dies würde Aspekte wie IAM-Authentifizierung, Verschlüsselung, DB-Cluster-Parameter, Instance-Parameter, Instance-Größen usw. umfassen.
- Die verwendete Netzwerktopologie, einschließlich der Ziel-VPC, ihrer Sicherheitsgruppen usw.

Sie können Neptune-Verwaltungs-APIs wie die folgenden verwenden, um diese Informationen zu sammeln:

- `DescribeDBClusters`
- `DescribeDBInstances`
- `DescribeDBClusterParameters`
- `DescribeDBParameters`
- `DescribeVpcs`

Mit den gesammelten Informationen können Sie das folgende Verfahren verwenden, um einen Sicherungs-Cluster in einer anderen Region einzurichten, zu dem Ihr Produktionscluster bei einem Fehler einen Failover ausführen kann.

Neptun-Streams aktivieren

Mit Modifizieren DBCluster ParameterGroup können Sie den Parameter `neptune_streams` auf 1 festlegen. Starten Sie anschließend alle Instances im DB-Cluster neu, damit die Änderung wirksam wird.

Sie sollten nach der Aktualisierung von Neptune-Streams mindestens einen Vorgang zum Hinzufügen oder Aktualisieren für den DB-Quell-Cluster ausführen. Dies füllt den Änderungs-Stream mit Datenpunkten, auf die später verwiesen werden kann, wenn der Produktions-Cluster erneut mit dem Sicherungs-Cluster synchronisiert wird.

Erstellen Sie eine neue VPC in der Region, in der Sie Ihren Backup-Cluster einrichten möchten

Bevor Sie einen neuen Neptune-DB-Cluster in einer anderen Region als der Region Ihres primären Clusters erstellen, müssen Sie in der Zielregion eine neue VPC einrichten, um den Cluster zu hosten. Die Konnektivität zwischen primären und Sicherungs-Clustern wird mittels VPC-Peering hergestellt, bei dem Datenverkehr über private Subnetze in verschiedenen VPCs verwendet wird. Um jedoch ein VPC-Peering zwischen zwei VPCs einzurichten, dürfen sich ihre CIDR-Blöcke oder IP-Adressräume nicht überschneiden. Das bedeutet, dass Sie nicht einfach die Standard-VPC in beiden Regionen verwenden können, da der CIDR-Block für eine Standard-VPC stets derselbe ist (`172.31.0.0/16`).

Sie können eine vorhandene VPC in der Zielregion verwenden, solange sie die folgenden Bedingungen erfüllt:

- Sie besitzt keinen CIDR-Block, der sich mit dem CIDR-Block der VPC überschneidet, in der sich der primäre Cluster befindet.
- Es besteht noch kein Peering mit einer anderen VPC, die denselben CIDR-Block wie die VPC hat, in dem sich der primäre Cluster befindet.

Wenn in der Zielregion keine geeignete VPC verfügbar ist, müssen Sie eine VPC mittels der Amazon-EC2-API `CreateVpc` erstellen.

Erstellen Sie einen Snapshot Ihres primären Clusters und stellen Sie ihn in der Ziel-Backup-Region wieder her

Erstellen Sie jetzt einen neuen Neptune-Cluster in einer geeigneten VPC in der Sicherungszielregion als Kopie des Produktions-Clusters:

Erstellen einer Kopie des Produktions-Clusters in der Sicherungsregion

1. Erstellen Sie in der Sicherungszielregion ermeut die Parameter und Parametergruppen, die vom DB-Produktions-Cluster verwendet werden. Sie können dies mittels `CreateDBClusterParameterGroup`, `CreateDBParameterGroup`, `ModifyDBClusterParameterGroup` und `ModifyDBParameterGroup` ausführen.

Beachten Sie, dass die APIs `CopyDBParameterGroup` und `CopyDBClusterParameterGroup` zurzeit kein regionsübergreifendes Kopieren unterstützen.

2. Verwenden Sie `CreateDBClusterSnapshot`, um einen Snapshot des Produktions-Clusters in der VPC in der Produktionsregion zu erstellen.
3. Verwenden Sie `CopyDBClusterSnapshot`, um den Snapshot zur VPC in der Zielregion der Sicherung zu kopieren.
4. Verwenden Sie `RestoreDBClusterFromSnapshot`, um mittels des kopierten Snapshots einen neuen DB-Cluster in der VPC in der Sicherungszielregion zu erstellen. Verwenden Sie die Konfigurationseinstellungen und Parameter, die Sie aus dem primären Produktions-Cluster kopiert haben.
5. Der neue Neptune-Cluster ist jetzt vorhanden, enthält jedoch keine Instances. Verwenden Sie diese Option, `CreateDBInstance`um eine neue primary/writer Instanz zu erstellen, die denselben Instance-Typ und dieselbe Größe wie die Writer-Instance Ihres Produktionsclusters hat. Zu diesem Zeitpunkt müssen keine zusätzlichen Read-Replicas erstellt werden, es sei denn, Ihre Backup-Instance wird vor einem Failover für I/O Lesevorgänge in der Zielregion verwendet.

Richten Sie VPC-Peering zwischen der VPC Ihres primären Clusters und der VPC Ihres neuen Backup-Clusters ein

Mit dem Einrichten von VPC-Peering ermöglichen Sie der VPC des primären Clusters die Kommunikation mit der VPC des Sicherungs-Clusters, als ob sie sich im selben privaten Netzwerk befinden würden. Führen Sie hierzu die folgenden Schritte aus:

1. Rufen Sie von der VPC des Produktions-Clusters aus die API `CreateVpcPeeringConnection` auf, um die Peering-Verbindung herzustellen.
2. Rufen Sie von der VPC des Sicherungs-Ziel-Clusters aus die API `AcceptVpcPeeringConnection` auf, um die Peering-Verbindung anzunehmen.
3. Verwenden Sie von der VPC des Produktions-Clusters aus die API `CreateRoute`, um der Routing-Tabelle der VPC eine Route hinzuzufügen, die den gesamten Datenverkehr zum CIDR-Block der Ziel-VPC umleitet, sodass dieser die VPC-Peering-Präfixliste verwendet.
4. Verwenden Sie auf ähnliche Weise von der VPC des Sicherungs–Ziel-Clusters aus die API `CreateRoute`, um der Routing-Tabelle der VPC eine Route hinzuzufügen, die den Datenverkehr zur VPC des primären Clusters leitet.

Richten Sie die Neptune Streams-Replikationsinfrastruktur ein

Nachdem beide Cluster bereitgestellt sind und die Netzwerkkommunikation zwischen beiden Regionen eingerichtet wurde, verwenden Sie die Neptune-to-Neptune CloudFormation Vorlage, um die Neptune Streams-Consumer-Lambda-Funktion mit der zusätzlichen Infrastruktur bereitzustellen, die die Datenreplikation unterstützt. Führen Sie dies in der VPC des primären Produktions-Clusters aus.

Die Parameter, die Sie für diesen CloudFormation Stack angeben müssen, sind:

- **`NeptuneStreamEndpoint`** – Der Stream-Endpunkt für den primären Cluster im URL-Format. Beispiel: `https://(cluster name):8182/pg/stream`.
- **`QueryEngine`** – Dies muss `gremlin`, `sparql` oder `openCypher` sein.
- **`RouteTableIds`** – Ermöglicht Ihnen das Hinzufügen von Routen für einen DynamoDB-VPC-Endpunkt und für einen VPC-Überwachungsendpunkt.

  Sie müssen zwei weitere Parameter (`CreateMonitoringEndpoint` und `CreateDynamoDBEndpoint`) auf true festlegen, wenn sie nicht bereits in der VPC des primären Clusters vorhanden sind. Wenn sie bereits existieren, stellen Sie sicher, dass sie auf False gesetzt sind. Andernfalls schlägt die CloudFormation Erstellung fehl.
- **`SecurityGroupIds`** – Gibt die Sicherheitsgruppe an, die vom Lambda-Consumer für die Kommunikation mit dem Neptune-Stream-Endpunkt des primären Clusters verwendet wird.

  Fügen Sie im Sicherungs-Ziel-Cluster eine Sicherheitsgruppe hinzu, die Datenverkehr von dieser Sicherheitsgruppe zulässt.
- **`SubnetIds`** – Eine Liste der Subnetz-IDs in der VPC des primären Clusters, die vom Lambda-Consumer für die Kommunikation mit dem primären Cluster verwendet werden können.
- **`TargetNeptuneClusterEndpoint`** – Der Cluster-Endpunkt (nur Hostname) des Ziel-Clusters für die Sicherung.
- **`TargetAWSRegion`**— Die AWS Region des Ziel-Backup-Clusters, z. B.`us-east-1`). Sie müssen diesen Parameter nur angeben, wenn sich die AWS Region des Ziel-Backup-Clusters von der Region des Neptune-Quell-Clusters unterscheidet, wie im Fall einer regionsübergreifenden Replikation. Wenn Quell- und Zielregion identisch sind, ist dieser Parameter optional.

  Beachten Sie, dass der Vorgang fehlschlägt, wenn der `TargetAWSRegion` Wert keine gültige AWS Region ist, die Neptune unterstützt.
- **`VPC`** – Die ID der VPC des primären Clusters.

Alle anderen Parameter können bei ihren Standardwerten belassen werden.

Sobald die CloudFormation Vorlage bereitgestellt wurde, beginnt Neptune damit, alle Änderungen vom primären Cluster auf den Backup-Cluster zu replizieren. Sie können diese Replikation in den von der Lambda-Consumer-Funktion generierten CloudWatch Protokollen überwachen.

Weitere Überlegungen

- Wenn Sie die IAM-Authentifizierung zwischen dem Primär- und dem Backup-Cluster verwenden müssen, können Sie diese auch beim Aufrufen der Vorlage einrichten. CloudFormation
- Wenn für den primären Cluster die Verschlüsselung im Ruhezustand aktiviert ist, sollten Sie überlegen, wie Sie die zugehörigen KMS-Schlüssel verwalten, wenn Sie den Snapshot in die Zielregion kopieren und in der Zielregion einen neuen KMS-Schlüssel zuordnen.
- Es stellt eine bewährte Methode dar, den Neptune-Endpunkten in Ihren Anwendungen DNS-CNAMEs voranzustellen. Wenn Sie dann manuell ein Failover auf den Ziel-Backup-Cluster durchführen müssen, können diese CNAMes so geändert werden, dass sie auf die Ziel-Cluster-Instanzendpunkte verweisen. and/or

Volltextsuche in Amazon Neptune mit Amazon Service OpenSearch

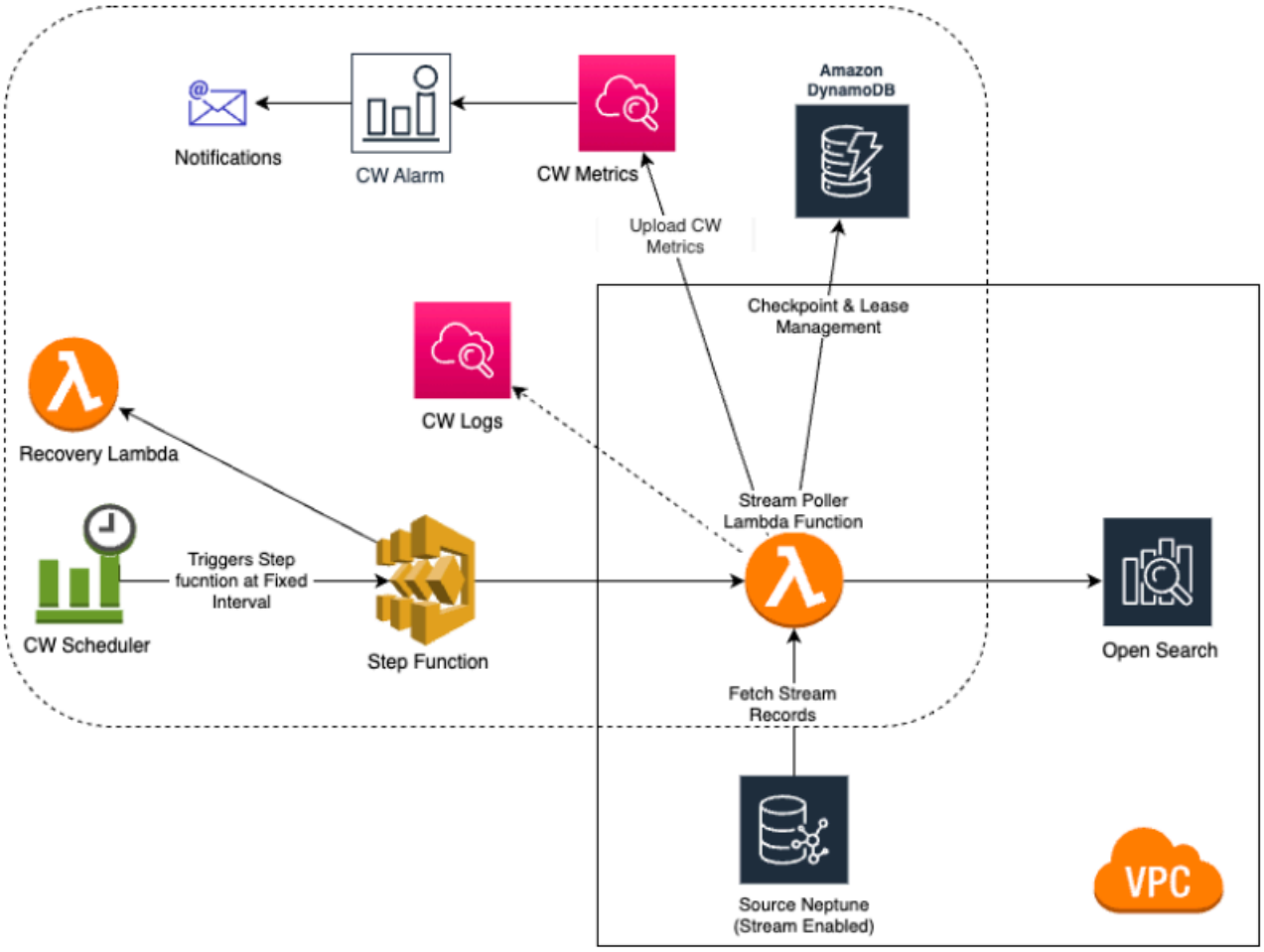
Neptune ist in Amazon OpenSearch Service (OpenSearch Service) integriert, um die Volltextsuche sowohl in Gremlin- als auch in SPARQL-Abfragen zu unterstützen.

Ab Engine-Version 1.3.0.0 unterstützt Amazon Neptune die Verwendung von Amazon OpenSearch Service Serverless für die Volltextsuche in Gremlin- und SPARQL-Abfragen.

> **Note**
>
> Für die Integration mit Amazon OpenSearch Service benötigt Neptune Elasticsearch Version 7.1 oder höher und funktioniert mit OpenSearch 2.3, 2.5 und höher. Neptune funktioniert auch mit OpenSearch Serverless.

Sie können Neptune mit einem vorhandenen OpenSearch Service-Cluster verwenden, der gemäß den gefüllt wurde. Neptun-Datenmodell für Daten OpenSearch Oder Sie können mithilfe eines Stacks eine mit Neptune verknüpfte OpenSearch Service-Domain erstellen. AWS CloudFormation

> **Important**
>
> Der hier beschriebene OpenSearch Neptun-zu-Replikationsprozess repliziert keine leeren Knoten. Dies ist eine wichtige Einschränkung, die es zu beachten gilt.
> Wenn Sie eine differenzierte Zugriffskontrolle auf Ihrem OpenSearch Cluster aktivieren, müssen Sie außerdem die IAM-Authentifizierung in Ihrer Neptune-Datenbank aktivieren.



Themen

- Neptune-to-OpenSearch Amazon-Replikation
- Replikation auf OpenSearch Serverless
- Abfragen von einem OpenSearch Cluster mit aktivierter Fine-Grained Access Control (FGAC)
- Verwendung der Apache-Lucene-Abfragesyntax in Neptune-Volltext-Suchabfragen
- Neptun-Datenmodell für Daten OpenSearch
- Parameter für die Neptune-Volltextsuche
- OpenSearch Indexierung ohne Zeichenketten in Amazon Neptune
- Full-text-search Ausführung von Abfragen in Amazon Neptune
- Beispiele für SPARQL-Volltextsuchabfragen in Neptune
- Verwenden der Neptune-Volltextsuche in Gremlin-Abfragen
- Fehlerbehebung bei der Neptune-Volltextsuche

Neptune-to-OpenSearch Amazon-Replikation

Amazon Neptune unterstützt die Volltextsuche in Gremlin- und SPARQL-Abfragen mithilfe von Amazon Service (Service). OpenSearch OpenSearch Sie können einen AWS CloudFormation Stack verwenden, um eine OpenSearch Service-Domain mit Neptune zu verknüpfen. Die CloudFormation Vorlage erstellt eine Streams-Consumer-Anwendungsinstanz, die die Replikation ermöglicht. Neptune-to-OpenSearch

Bevor Sie beginnen, benötigen Sie einen vorhandenen Neptune-DB-Cluster, auf dem Streams als Quelle aktiviert sind, und eine OpenSearch Service-Domain, die als Replikationsziel dient.

Wenn Sie in der VPC, in der sich Ihr Neptune-DB-Cluster befindet, bereits über eine bestehende OpenSearch Ziel-Servicedomäne verfügen, auf die Lambda zugreifen kann, kann die Vorlage diese verwenden. Andernfalls müssen Sie eine neue erstellen.

> **Note**
>
> Der OpenSearch Cluster und die Lambda-Funktion, die Sie erstellen, müssen sich in derselben VPC wie Ihr Neptune-DB-Cluster befinden, und der OpenSearch Cluster muss im VPC-Modus (nicht im Internetmodus) konfiguriert sein.

Wir empfehlen, dass Sie eine neu erstellte Neptune-Instanz für die Verwendung mit OpenSearch Service verwenden. Wenn Sie eine bestehende Instanz verwenden, die bereits Daten enthält, sollten Sie vor dem Durchführen von Abfragen eine OpenSearch Service-Datensynchronisierung durchführen, da es sonst zu Dateninkonsistenzen kommen kann. Dieses GitHub Projekt bietet ein Beispiel für die Durchführung der Synchronisation: Export Neptune to OpenSearch ()https://github.com/awslabs/amazon-neptune-tools/tree/master/export-neptune-to-elasticsearch.

> **Important**
>
> Für die Integration mit Amazon OpenSearch Service benötigt Neptune Elasticsearch Version 7.1 oder höher und funktioniert mit OpenSearch 2.3, 2.5 und future kompatiblen Opensearch-Versionen.

> **Note**
>
> Ab Engine-Version 1.3.0.0 unterstützt Amazon Neptune die Verwendung von Amazon OpenSearch Service Serverless für die Volltextsuche in Gremlin- und SPARQL-Abfragen.

Themen

- Mit einem AWS CloudFormation Vorlage zum Starten der Neptune-to-OpenSearch Replikation
- Aktivierung der Volltextsuche in bestehenden Neptune-Datenbanken
- Aktualisieren des Stream-Pollers
- Deaktivieren und erneutes Aktivieren des Stream-Poller-Prozesses

Mit einem AWS CloudFormation Vorlage zum Starten der Neptune-to-OpenSearch Replikation

Starten Sie ein CloudFormation Stack, der für Ihre Region spezifisch ist

Jede der folgenden CloudFormation Vorlagen erstellt eine Streams-Consumer-Anwendungsinstanz in einer bestimmten AWS Region. Um den entsprechenden Stack über die CloudFormation Konsole zu starten, wählen Sie je nach AWS Region, die Sie verwenden möchten, eine der Schaltflächen „Stack starten“ in der folgenden Tabelle.

Region	Anzeigen	In Designer anzeigen	Starten
USA Ost (Nord-Virginia)	Anzeigen	In Designer anzeigen	
USA Ost (Ohio)	Anzeigen	In Designer anzeigen	
USA West (Nordkalifornien)	Anzeigen	In Designer anzeigen	
USA West (Oregon)	Anzeigen	In Designer anzeigen	
Kanada (Zentral)	Anzeigen	In Designer anzeigen	

Region	Anzeigen	In Designer anzeigen	Starten
Südamerika (São Paulo)	Anzeigen	In Designer anzeigen	Launch Stack
Europa (Stockholm)	Anzeigen	In Designer anzeigen	Launch Stack
Europa (Irland)	Anzeigen	In Designer anzeigen	Launch Stack
Europa (London)	Anzeigen	In Designer anzeigen	Launch Stack
Europa (Paris)	Anzeigen	In Designer anzeigen	Launch Stack
Europa (Spain)	Anzeigen	In Designer anzeigen	Launch Stack
Europa (Frankfurt)	Anzeigen	In Designer anzeigen	Launch Stack
Middle East (Bahrain)	Anzeigen	In Designer anzeigen	Launch Stack
Naher Osten (VAE)	Anzeigen	In Designer anzeigen	Launch Stack
Israel (Tel Aviv)	Anzeigen	In Designer anzeigen	Launch Stack
Afrika (Kapstadt)	Anzeigen	In Designer anzeigen	Launch Stack
Asien-Pazifik (Hongkong)	Anzeigen	In Designer anzeigen	Launch Stack
Asien-Pazifik (Tokio)	Anzeigen	In Designer anzeigen	Launch Stack
Asien-Pazifik (Seoul)	Anzeigen	In Designer anzeigen	Launch Stack

Region	Anzeigen	In Designer anzeigen	Starten
Asien-Pazifik (Singapur)	Anzeigen	In Designer anzeigen	
Asien-Pazifik (Jakarta)	Anzeigen	In Designer anzeigen	
Asien-Pazifik (Mumbai)	Anzeigen	In Designer anzeigen	
China (Peking)	Anzeigen	In Designer anzeigen	
China (Ningxia)	Anzeigen	In Designer anzeigen	
AWS GovCloud (US-West)	Anzeigen	In Designer anzeigen	
AWS GovCloud (US-East)	Anzeigen	In Designer anzeigen	

Wählen Sie auf der Seite Create Stack (Stack erstellen) die Option Next (Weiter) aus.

Fügen Sie Details zu dem neuen OpenSearch Stack hinzu, den Sie gerade erstellen

Auf der Seite Stack-Details angeben finden Sie Eigenschaften und Parameter für die Einrichtung der Volltextsuche.

Stack-Name — Der Name des neuen CloudFormation Stacks, den Sie erstellen. Sie können in der Regel den Standardwert `NeptuneStreamPoller` verwenden.

Geben Sie unter Parameter, Folgendes ein:

Netzwerkkonfiguration für die VPC, auf der der Streams-Consumer ausgeführt wird

- **`VPC`** – Geben Sie den Namen der VPC an, in der die abfragende Lambda-Funktion ausgeführt werden wird.
- **`List of Subnet IDs`** – die Subnetze, für die eine Netzwerkschnittstelle eingerichtet ist. Fügen Sie Subnetze hinzu, die Ihrem Neptune-Cluster entsprechen.

- **`List of Security Group Ids`** – Geben Sie die IDs von Sicherheitsgruppen an, die eingehenden Schreibzugriff auf Ihren Neptune-Quell-DB-Cluster gewähren.
- **`List of Route Table Ids`** – Dies wird benötigt, um einen Amazon-DynamoDB-Endpunkt in Ihrer Neptune-VPC zu erstellen, wenn noch keiner vorhanden ist. Sie müssen eine durch Komma getrennte Liste der Routing-Tabellen-IDs angeben, die den Subnetzen zugeordnet sind.
- **`Require to create Dynamo DB VPC Endpoint`** – Ein boolescher Wert, der standardmäßig `true` lautet. Sie müssen ihn nur zu `false` ändern, wenn Sie bereits einen DynamoDB-Endpunkt in Ihrer VPC erstellt haben.
- **`Require to create Monitoring VPC Endpoint`** – Ein boolescher Wert, der standardmäßig `true` lautet. Sie müssen ihn nur in `false` ändern, wenn Sie bereits einen Überwachungsendpunkt in Ihrer VPC erstellt haben.

Stream-Poller

- **`Application Name`** – Sie können diesen Satz in der Regel auf dem Standardwert (`NeptuneStream`) belassen. Wenn Sie einen anderen Namen verwenden, muss dieser eindeutig sein.
- **`Memory size for Lambda Poller`** – Wird verwendet, um die Speichergröße festzulegen, die für die Lambda-Poller-Funktion verfügbar ist. Der Standardwert ist 2.048 Megabyte.
- **`Lambda Runtime`** – Die Sprache, die in der Lambda-Funktion verwendet wird, mit der Elemente aus dem Neptune-Stream abgerufen werden. Sie können dies entweder auf `python3.9` oder auf `java8` setzen.
- **`S3 Bucket having Lambda code artifacts`** – Lassen Sie dieses Feld leer, es sei denn, Sie verwenden eine benutzerdefinierte Lambda-Abfragefunktion, die aus einem anderen S3-Bucket geladen wird.
- **`S3 Key corresponding to Lambda Code artifacts`** – Lassen Sie dieses Feld leer, es sei denn, Sie verwenden eine benutzerdefinierte Lambda-Abfragefunktion.
- **`StartingCheckpoint`** – Der Start-Prüfpunkt für den Stream-Poller. Die Standardeinstellung ist `0:0`, was bedeutet, dass mit dem Anfang des Neptune-Streams begonnen wird.
- **`StreamPollerInitialState`** – Der anfängliche Zustand des Pollers. Die Standardeinstellung ist `ENABLED`, was bedeutet, dass die Stream-Replikation beginnt, sobald die gesamte Stack-Erstellung abgeschlossen ist.
- **`Logging level for Lambda`** – Belassen Sie diesen Satz generell auf dem Standardwert, `INFO`

- **`Managed Policies for Lambda Execution`** – Lassen Sie dieses Feld im Allgemeinen leer, es sei denn, Sie verwenden eine benutzerdefinierte Lambda-Abfragefunktion.
- **`Stream Records Handler`** – Lassen Sie dieses Feld im Allgemeinen leer, es sei denn, Sie verwenden einen benutzerdefinierten Handler für die Datensätze in Neptune-Streams.
- **`Maximum records Fetched from Stream`** – Mit diesem Parameter können Sie die Leistung optimieren. Der Standardwert (`100`) ist ein guter Anfangspunkt. Der maximal zulässige Wert ist 10.000. Je höher die Zahl ist, desto weniger Netzwerkaufrufe werden benötigt, um Datensätze aus dem Stream zu lesen. Allerdings ist umso mehr Speicher ist erforderlich, um die Datensätze zu verarbeiten.
- **`Max wait time between two Polls (in Seconds)`** – Legt fest, wie häufig der Lambda-Poller aufgerufen wird, um die Neptune-Streams abzufragen. Legen Sie diesen Wert für kontinuierliche Abfragen auf „0“ fest. Die Höchstwert beträgt 3.600 Sekunden (1 Stunde). Der Standardwert (60 Sekunden) ist ein guter Ausgangspunkt, je nachdem, wie schnell sich Ihre Graphdaten ändern.
- **`Maximum Continuous polling period (in Seconds)`** – Wird verwendet, um ein Timeout für die Lambda-Abfragefunktion festzulegen. Es sollte zwischen 5 und 900 Sekunden liegen. Der Standardwert (600 Sekunden) ist ein guter Ausgangspunkt.
- **`Step Function Fallback Period`**— Die Anzahl der Step-Function-Fallback-Perioden-Einheiten, die auf den Poller warten sollen. Danach wird die Step-Funktion über Amazon CloudWatch Events aufgerufen, um sich nach einem Ausfall zu erholen. Der Standardwert (5 Minuten) ist ein guter Anfangspunkt.
- **`Step Function Fallback Period Unit`** – Die Zeiteinheiten (Minuten, Stunden, Tage), die zum Messen der vorhergehenden `Step Function Fallback Period` verwendet werden. Der Standardwert (Minuten) ist im Allgemeinen ausreichend.
- **`Enable/disable creation of Elastic Search Mappings by Lambda`**— Markierung für die enable/disable automatische Verwaltung von Elastic Search-Mappings durch den Stream-Poller. Elastic Search-Mappings sind entscheidend für die Konvertierung zwischen Neptune- und Elastic Search-Datentypen. Dies sollte nur von erfahrenen Benutzern deaktiviert werden, die die direkte, manuelle Kontrolle über alle Mappings in ihrem Elastic Search-Index übernehmen möchten. Der Standardwert ist `true`.
- **`Data replication scope`**— Legt fest, ob sowohl Knoten als auch Kanten repliziert werden sollen oder nur Knoten zu OpenSearch (dies gilt nur für Gremlin-Engine-Daten). Der Standardwert (All (Alle)) ist in der Regel ein guter Anfangspunkt.
- **`Ignore OpenSearch missing document error`**— Markierung, um zu bestimmen, ob ein Fehler in einem fehlenden Dokument ignoriert OpenSearch werden kann. „Fehlendes Dokument“-

Fehler können vereinzelt vorkommen, erfordern jedoch ein manuelles Eingreifen, wenn sie nicht ignoriert werden. Der Standardwert (`True`) ist in der Regel ein guter Ausgangspunkt.

- **`Enable Non-String Indexing`** – Flag zum Aktivieren oder Deaktivieren der Indizierung von Feldern, die keinen Zeichenfolgeninhalt haben. Wenn dieses Kennzeichen auf gesetzt ist, werden Felder`true`, die keine Zeichenfolgen sind, indexiert OpenSearch, oder wenn`false`, werden nur Zeichenkettenfelder indexiert. Der Standardwert ist `true`.
- **`Properties to exclude from being inserted into OpenSearch`**— Eine durch Kommas getrennte Liste von Eigenschafts- oder Prädikatschlüsseln, die von der Indizierung ausgeschlossen werden sollen. OpenSearch Wenn dieser CFN-Parameterwert leer gelassen wird, werden alle Eigenschaftsschlüssel indexiert.
- **`Datatypes to exclude from being inserted into OpenSearch`**— Eine kommagetrennte Liste von Eigenschaften- oder Prädikat-Datentypen, die von der Indizierung ausgeschlossen werden sollen. OpenSearch Wenn dieser CFN-Parameterwert leer gelassen wird, werden alle Eigenschaftswerte, die sicher in Datentypen konvertiert werden können, indexiert. OpenSearch

Neptune-Stream

- **`Endpoint of source Neptune Stream`** – (Erforderlich) Dies kann eine von zwei Formen haben:
  - **`https://`*`your DB cluster`*`:`*`port`*`/propertygraph/stream`** (oder dessen Alias, `https://`*`your DB cluster`*`:`*`port`*`/pg/stream`).
  - `https://`*`your DB cluster`*`:`*`port`*`/sparql/stream`
- **`Neptune Query Engine`** – Wählen Sie Gremlin oder SPARQL.
- **`Is IAM Auth Enabled?`** – Wenn Ihr Neptune-DB-Cluster die IAM-Authentifizierung verwendet, setzen Sie diesen Parameter auf `true`.
- **`Neptune Cluster Resource Id`** – Wenn Ihr Neptune-DB-Cluster die IAM-Authentifizierung verwendet, setzen Sie diesen Parameter auf die Cluster-Ressourcen-ID. Die Ressourcen-ID ist nicht mit der Cluster-ID identisch. Stattdessen hat sie das folgende Format: `cluster-` gefolgt von 28 alphanumerischen Zeichen. Sie finden sie unter Cluster-Details in der Neptune-Konsole.

Zielcluster OpenSearch

- **`Endpoint for OpenSearch service`**— (Erforderlich) Geben Sie den Endpunkt für den OpenSearch Service in Ihrer VPC an.

- **`Number of Shards for OpenSearch Index`** – Der Standardwert (5) ist in der Regel ein guter Ausgangspunkt.
- **`Number of Replicas for OpenSearch Index`** – Der Standardwert (1) ist in der Regel ein guter Ausgangspunkt.
- **`Geo Location Fields for Mapping`** – Wenn Sie Geolocation-Felder verwenden, führen Sie hier die Eigenschaftsschlüssel auf.

Alarm

- **`Require to create Cloud watch Alarm`**— Stellen Sie diesen Wert ein, `true` wenn Sie einen CloudWatch Alarm für den neuen Stack erstellen möchten.
- **`SNS Topic ARN for Cloudwatch Alarm Notifications`**— Das SNS-Thema ARN, an das CloudWatch Alarmbenachrichtigungen gesendet werden sollen (nur erforderlich, wenn Alarme aktiviert sind).
- **`Email for Alarm Notifications`** – Die E-Mail-Adresse, an die Alarmbenachrichtigungen gesendet werden sollen (nur erforderlich, wenn Alarme aktiviert sind).

Als Ziel der Alarmbenachrichtigung können Sie nur SNS, nur E-Mail oder sowohl SNS als auch E-Mail hinzufügen.

Führen Sie das aus CloudFormation Vorlage

Jetzt können Sie den Prozess für die Bereitstellung einer Neptune-Streams-Consumer-Anwendungs-Instance wie folgt abschließen:

1. Wählen Sie auf der Seite „Stack-Details angeben“ die Option Weiter aus. CloudFormation
2. Wählen Sie auf der Seite Optionen Weiter aus.
3. Aktivieren Sie auf der Seite Überprüfen das erste Kontrollkästchen, um zu bestätigen, dass CloudFormation IAM-Ressourcen erstellt. Aktivieren Sie das zweite Kontrollkästchen, um `CAPABILITY_AUTO_EXPAND` für den neuen Stack zu bestätigen.

> **Note**
>
> `CAPABILITY_AUTO_EXPAND` bestätigt explizit, dass Makros ohne vorherige Überprüfung beim Erstellen des Stacks erweitert werden. Benutzer erstellen häufig einen Änderungssatz aus einer verarbeiteten Vorlage, sodass die von Makros durchgeführten

Änderungen überprüft werden können, ehe der Stack tatsächlich erstellt wird. Weitere Informationen finden Sie in der CloudFormation CreateStackAPI-Referenz unter AWS CloudFormation API-Vorgang.

Wählen Sie die Option Erstellen aus.

Aktivierung der Volltextsuche in bestehenden Neptune-Datenbanken

Dies sind die etablierten Ansätze, um die Volltextsuche in bestehenden Amazon Neptune Neptune-Datenbanken zu ermöglichen. Je nachdem, ob Sie Ihre Schreib-Workloads pausieren können oder nicht, können die Schritte leicht variieren. In diesem Handbuch werden die empfohlenen Schritte für beide Szenarien beschrieben — wann Sie Schreibvorgänge unterbrechen können und wann nicht. Es behandelt das Aktivieren von Neptune-Streams, das Erstellen eines Datenbankklons, das Synchronisieren von Daten mit einer OpenSearch Domain und das Einrichten kontinuierlicher Updates. Die Anleitung nutzt AWS Dienste und Open-Source-Tools, um den Prozess zu rationalisieren und Ausfallzeiten zu minimieren.

Wenn Sie Ihre Schreib-Workloads unterbrechen können

Der beste Weg, die Volltextsuche in einer vorhandenen Neptune-Datenbank zu aktivieren, ist im Allgemeinen der folgende, vorausgesetzt, Sie können Ihre Schreib-Workloads unterbrechen. Dazu müssen Sie einen Klon erstellen, die Streams mithilfe eines Cluster-Parameters aktivieren und alle Instances neu starten. Das Erstellen eines Klons ist ein relativ schneller Vorgang, so dass die damit verbundenen Ausfallzeiten nur begrenzt sind.

Die erforderlichen Schritte sind:

1. Halten Sie alle Schreib-Workloads in der Datenbank an.
2. Aktivieren Sie Streams in der Datenbank (siehe Aktivieren von Neptune-Streams).
3. Erstellen Sie einen Klon der Datenbank (siehe Klonen von Datenbanken in Neptune).
4. Setzen Sie die Schreib-Workloads fort.
5. Verwenden Sie das Tool export-Neptune-to-Elasticsearch auf Github, um eine einmalige Synchronisation von der geklonten Datenbank zur Domain durchzuführen. OpenSearch
6. Verwenden Sie die CloudFormation -Vorlage für Ihre Region, um die Synchronisation von Ihrer ursprünglichen Datenbank aus mit kontinuierlicher Aktualisierung zu starten (es sind keine Konfigurationsänderungen in der Vorlage erforderlich).

7. Löschen Sie die geklonte Datenbank und den für das Tool erstellten Stack. CloudFormation `export-neptune-to-elasticsearch`

> **Note**
>
> Wenn Sie export-neptune-to-elasticsearch mit Opensearch Serverless verwenden, muss der von export-Neptune-to-elasticsearch`LambdaExecutionRole` erstellte Cloudformation-Stack zur Opensearch-Datenzugriffsrichtlinie hinzugefügt werden, um die Daten erfolgreich zu replizieren.

Wenn Sie Ihre Schreib-Workloads nicht unterbrechen können

Wenn Sie es sich nicht leisten können, Schreib-Workloads in Ihrer Datenbank auszusetzen, finden Sie hier einen Ansatz, der noch weniger Ausfallzeiten beinhaltet als der oben empfohlene Ansatz, der jedoch vorsichtig durchgeführt werden muss:

1. Aktivieren Sie Streams in der Datenbank (siehe Aktivieren von Neptune-Streams).
2. Erstellen Sie einen Klon der Datenbank (siehe Klonen von Datenbanken in Neptune).
3. Holen Sie sich die neueste `eventID` zu den Streams in der geklonten Datenbank, indem Sie einen Befehl dieser Art für den Streams-API-Endpunkt ausführen (weitere Informationen finden Sie unter Aufrufen der Neptune-Streams-REST-API):

   ```
   curl "https://(your neptune endpoint):(port)/(propertygraph or sparql)/stream?
   iteratorType=LATEST"
   ```

   Notieren Sie sich die Werte in den Feldern `commitNum` und `opNum` im `lastEventId`-Objekt in der Antwort.
4. Verwenden Sie das Tool export-neptune-to-elasticsearch auf Github, OpenSearch um eine einmalige Synchronisation von der geklonten Datenbank zur Domain durchzuführen.
5. Verwenden Sie die CloudFormation -Vorlage für Ihre Region, um die Synchronisation von Ihrer ursprünglichen Datenbank aus mit kontinuierlicher Aktualisierung zu starten.

   Nehmen Sie bei der Erstellung des Stacks die folgende Änderung vor: Setzen Sie auf der Seite mit den Stack-Details im Abschnitt Parameter den Wert des `StartingCheckpoint` Felds auf*`commitNum`*: *`opnum`* und verwenden Sie dabei die Werte und, die Sie oben aufgezeichnet haben. `commitNum` opNum

6. Löschen Sie die geklonte Datenbank und den für das `export-neptune-to-elasticsearch` Tool erstellten CloudFormation Stack.

Aktualisieren des Stream-Pollers

Die folgenden Informationen beschreiben die Schritte, die erforderlich sind, um den Stream-Poller mithilfe der AWS Managementkonsole mit den neuesten Lambda-Artefakten zu aktualisieren.

So aktualisieren Sie den Stream-Poller mit den neuesten Lambda-Artefakten

Sie können den Stream-Poller wie folgt mit den neuesten Lambda-Artefakten aktualisieren:

1. Navigieren Sie im AWS-Managementkonsole zum übergeordneten CloudFormation CloudFormation Hauptstapel und wählen Sie ihn aus.
2. Wählen Sie die Option Aktualisieren für den Stack aus.
3. Wählen Sie Aktuelle Vorlage ersetzen aus.
4. Wählen Sie als Vorlagenquelle Amazon-S3-URL aus und geben Sie die folgende S3-URL ein:

   ```
   https://aws-neptune-customer-samples.s3.amazonaws.com/neptune-stream/
   neptune_to_elastic_search.json
   ```

5. Wählen Sie Weiter aus, ohne die CloudFormation Parameter zu ändern.
6. Wählen Sie Stack aktualisieren aus.

Der Stack aktualisiert jetzt die Lambda-Artefakte mit den neuesten Artefakten.

Erweitern des Stream-Pollers zur Unterstützung benutzerdefinierter Felder

Der aktuelle Stream-Poller kann einfach erweitert werden, um benutzerdefinierten Code für den Umgang mit benutzerdefinierten Feldern zu schreiben, wie in diesem Blogbeitrag ausführlich erläutert wird: Erfassen von Graphänderungen mit Neptune Streams.

> **Note**
>
> Achten Sie beim Hinzufügen eines benutzerdefinierten Felds darauf OpenSearch, das neue Feld als inneres Objekt eines Prädikats hinzuzufügen (siehe Neptune-Datenmodell für die Volltextsuche).

Deaktivieren und erneutes Aktivieren des Stream-Poller-Prozesses

> **Warning**
>
> Seien Sie vorsichtig, wenn Sie den Stream-Poller-Prozess deaktivieren! Es können Datenverluste auftreten, wenn der Prozess länger als das Ablaufzeitfenster für den Stream angehalten wird. Das Standardzeitfenster beträgt 7 Tage, aber ab Engine-Version 1.2.0.0 können Sie ein benutzerdefiniertes Stream-Ablaufzeitfenster von maximal 90 Tagen festlegen.

Deaktivieren (Unterbrechen) des Stream-Poller-Prozesses

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die EventBridge Amazon-Konsole unter https://console.aws.amazon.com/events/.
2. Wählen Sie im Navigationsbereich Regeln aus.
3. Wählen Sie die Regel aus, deren Name den Namen enthält, den Sie in der CloudFormation Vorlage, mit der Sie den Stream-Poller eingerichtet haben, als Anwendungsname angegeben haben.
4. Wählen Sie Disable (deaktivieren) aus.
5. Öffnen Sie die Step Functions Functions-Konsole unter https://console.aws.amazon.com/states/.
6. Wählen Sie die laufende Step-Funktion aus, die dem Stream-Poller-Prozess entspricht. Auch hier enthält der Name dieser Step-Funktion den Namen, den Sie in der CloudFormation Vorlage, mit der Sie den Stream-Poller eingerichtet haben, als Anwendungsname angegeben haben. Sie können nach dem Ausführungsstatus der Funktion filtern, um nur laufende Funktionen anzuzeigen.
7. Wählen Sie Beenden aus.

Re-enabling der Stream-Poller-Prozess

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die EventBridge Amazon-Konsole unter https://console.aws.amazon.com/events/.
2. Wählen Sie im Navigationsbereich Regeln aus.

3. Wählen Sie die Regel aus, deren Name den Namen enthält, den Sie in der CloudFormation Vorlage, mit der Sie den Stream-Poller eingerichtet haben, als Anwendungsname angegeben haben.
4. Wählen Sie Enable (Aktivieren) aus. Die auf dem angegebenen geplanten Intervall basierende Ereignisregel löst jetzt eine neue Ausführung der Step-Funktion aus.

Replikation auf OpenSearch Serverless

Ab Engine-Version 1.3.0.0 unterstützt Amazon Neptune die Verwendung von Amazon OpenSearch Service Serverless für die Volltextsuche in Gremlin- und SPARQL-Abfragen. Wenn Sie OpenSearch Serverless verwenden, müssen Sie die IAM-Authentifizierung auf Ihrem Neptune-Datenbank-Cluster aktivieren. Neptune-Datenbankcluster mit deaktivierter IAM-Authentifizierung werden von Serverless nicht unterstützt. OpenSearch

Für die AWS Lambda-Funktion des Pollers sind Änderungen erforderlich

Wenn Sie auf OpenSearch Serverless replizieren, fügen Sie die Ausführungsrolle Lambda Stream Poller zur Datenzugriffsrichtlinie für die Serverless-Sammlung hinzu. OpenSearch Der ARN für die Lambda-Stream-Poller-Ausführungsrolle hat das folgende Format:

```
arn:aws:iam::(account ID):role/stack-name-NeptuneOSReplication-
NeptuneStreamPollerExecu-(uuid)
```

Für das Massenimport-Hilfsprogramm sind Änderungen erforderlich

Wenn Sie die Synchronisierung vorhandener Daten mit OpenSearch Serverless verwenden export-neptune-to-elasticsearch, fügen Sie das `LambdaExecutionRole` From the CloudFormation Stack zur Datenzugriffsrichtlinie für die OpenSearch Serverless-Sammlung hinzu. Der ARN für `LambdaExecutionRole` hat dieses Format:

```
arn:aws:iam::012345678901:role/stack-name-LambdaExecutionRole-(id)
```

Weitere Informationen finden Sie unter Datenzugriffskontrolle für Amazon OpenSearch Serverless.

Erforderliche Änderungen an der IAM-Rolle, die für die Abfrage von Neptune verwendet wird

Die IAM-Entität (Benutzer oder Rolle), die für die Verbindung mit der Neptune-Datenbank verwendet wird, sollte über Berechtigungen sowohl für Neptune als auch für die Serverless-Sammlung verfügen. OpenSearch Das bedeutet, dass Ihrem Benutzer oder Ihrer Rolle eine serverlose Richtlinie wie die folgende beigefügt sein muss: OpenSearch

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"
    }
  ]
}
```

Weitere Informationen finden Sie unter Erstellen benutzerdefinierter IAM-Richtlinienerklärungen für den Zugriff auf Daten in Amazon Neptune.

Abfragen von einem OpenSearch Cluster mit aktivierter Fine-Grained Access Control (FGAC)

Wenn Sie eine differenzierte Zugriffskontrolle auf Ihrem OpenSearch Cluster aktiviert haben, müssen Sie die IAM-Authentifizierung auch in Ihrer Neptune-Datenbank aktivieren.

Die IAM-Entität (Benutzer oder Rolle), die für die Verbindung mit der Neptune-Datenbank verwendet wird, sollte über Berechtigungen sowohl für Neptune als auch für den Cluster verfügen. OpenSearch

Das bedeutet, dass Ihrem Benutzer oder Ihrer Rolle eine OpenSearch Servicerichtlinie wie die folgende beigefügt sein muss:

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/*"
    }
  ]
}
```

Weitere Informationen finden Sie unter Erstellen benutzerdefinierter IAM-Richtlinienerklärungen für den Zugriff auf Daten in Amazon Neptune.

Verwendung der Apache-Lucene-Abfragesyntax in Neptune-Volltext-Suchabfragen

OpenSearch unterstützt die Verwendung der Apache Lucene-Syntax für query_string-Abfragen. Dies ist besonders nützlich, um mehrere Filter in einer Abfrage zu übergeben.

Neptune verwendet eine verschachtelte Struktur zum Speichern von Eigenschaften in einem OpenSearch Dokument (siehe). Neptune-Datenmodell für die Volltextsuche Wenn Sie die Lucene-Syntax verwenden, müssen Sie die vollständigen Pfade zu den Eigenschaften in diesem verschachtelten Modell verwenden.

Hier ist ein Gremlin-Beispiel:

```
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")
```

Hier ist ein SPARQL-Beispiel:

```
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 .
}
```

Neptun-Datenmodell für Daten OpenSearch

Amazon Neptune verwendet eine einheitliche JSON-Dokumentstruktur zum Speichern von SPARQL- und Gremlin-Daten im Service. OpenSearch Jedes Dokument in OpenSearch entspricht einer Entität und speichert alle relevanten Informationen für diese Entität. Bei Gremlin werden Scheitelpunkte und Kanten als Entitäten betrachtet, sodass die entsprechenden OpenSearch Dokumente Informationen über Scheitelpunkte, Beschriftungen und Eigenschaften enthalten. Bei SPARQL können Subjekte als Entitäten betrachtet werden, sodass entsprechende OpenSearch Dokumente Informationen über alle Prädikat-Objekt-Paare in einem Dokument enthalten.

> **Note**
>
> Die Neptune-to-OpenSearch Replikationsimplementierung speichert nur Zeichenkettendaten. Sie können sie jedoch so ändern, dass andere Datentypen gespeichert werden.

Die einheitliche JSON-Dokumentstruktur sieht folgendermaßen aus:

```
{
  "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` – Eindeutige Entitäts-ID, die das Dokument repräsentiert.
  - Bei SPARQL ist dies der Subjekt-URI.
  - Für Gremlin ist dies die `Vertex_ID` oder `Edge_ID`.
- `entity_type` – Stellt eine oder mehrere Bezeichnungen für einen Vertex oder einen Edge bzw. keine oder mehrere `rdf:type`-Prädikatwerte für ein Subjekt dar.
- `document_type` – Gibt an, ob das aktuelle Dokument einen Vertex, einen Edge oder eine RDF-Ressource darstellt.
- `predicates` – Bei Gremlin werden Eigenschaften und Werte für einen Vertex oder einen Edge gespeichert. Bei SPARQL werden Prädikatobjektpaare gespeichert.

  Der Eigenschaftsname hat die Form `properties.name.value` in OpenSearch. Um ihn abfragen zu können, müssen Sie ihn in dieser Form benennen.
- `value` – Ein Eigenschaftswert für Gremlin oder ein Objektwert für SPARQL.
- `graph` – Ein benannter Graph für SPARQL.
- `language` – Ein Sprach-Tag für ein `rdf:langString`-Literal in SPARQL.

Beispiel für ein SPARQL-Dokument OpenSearch

Daten

```
@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
```

Dokumente

```
{
  "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"
}
```

Beispiel für ein Gremlin-Dokument OpenSearch

Daten

```
# 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
```

Dokumente

```
{
  "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"
      }
    ]
  }
}
```

Parameter für die Neptune-Volltextsuche

Amazon Neptune verwendet die folgenden Parameter für die Angabe von OpenSearch Volltextabfragen in Gremlin und SPARQL:

- **`queryType`**— (Erforderlich) Der Abfragetyp. OpenSearch (Eine Liste der Abfragetypen finden Sie in der OpenSearch Dokumentation). Neptune unterstützt die folgenden OpenSearch Abfragetypen:
  - simple_query_string – Gibt Dokumente zurück, die auf einer bereitgestellten Abfragezeichenfolge basieren, wobei ein Parser mit einer eingeschränkten, aber fehlertoleranten Lucene-Syntax verwendet wird. Dies ist der Standardabfragetyp.

    Diese Abfrage verwendet eine einfache Syntax, mit der die bereitgestellte Abfragezeichenfolge auf Grundlage spezieller Operatoren analysiert und in Begriffe aufgeteilt wird. Die Abfrage analysiert dann jeden Begriff unabhängig von einander, bevor übereinstimmende Dokumente zurückgegeben werden.

    Obwohl die Syntax eingeschränkter ist als die `query_string`-Abfrage, gibt die `simple_query_string`-Abfrage keine Fehler für ungültige Syntax zurück. Stattdessen werden alle ungültigen Teile der Abfragezeichenfolge ignoriert.
  - match – die `match`-Abfrage ist die Standardabfrage für das Durchführen einer Volltextsuche, einschließlich Optionen für Fuzzyübereinstimmungen.

- prefix – Gibt Dokumente zurück, die ein bestimmtes Präfix in einem angegebenen Feld enthalten.
- fuzzy – Gibt Dokumente zurück, die dem Suchbegriff ähnelnde Begriffe enthalten, wie sie anhand einer Levenshtein-Bearbeitungsdistanz gemessen werden.

  Eine Bearbeitungsdistanz ist die erforderliche Anzahl der Änderungen einzelner Zeichen, damit ein Begriff in einen anderen umgewandelt werden kann. Diese Änderungen können Folgendes umfassen:

  - Ändern eines Zeichens (Maus zu Haus).
  - Entfernen eines Zeichens (Reis zu Eis).
  - Einfügen eines Zeichens (Bauch zu Brauch).
  - Transponieren von zwei benachbarten Zeichen (Beine zu Biene).

  Für die Suche nach ähnlichen Begriffen erstellt die Fuzzyabfrage einen Satz aller möglichen Variationen und Erweiterungen des Suchbegriffs innerhalb einer angegebenen Bearbeitungsdistanz und gibt dann exakte Übereinstimmungen für jede dieser Varianten zurück.
- term – Gibt Dokumente zurück, die eine exakte Übereinstimmung eines bestimmten Begriffs in einem der angegebenen Felder enthalten.

  Mit der `term`-Abfrage können Sie nach Dokumenten suchen, die auf einem genauen Wert basieren, z, B. einem Preis, einer Produkt-ID oder einem Benutzernamen.

  > ⚠ Warning
  >
  > Vermeiden Sie die Verwendung der Begriffsabfrage für Textfelder. OpenSearch Ändert standardmäßig die Werte von Textfeldern im Rahmen seiner Analyse, was es schwierig machen kann, exakte Übereinstimmungen für Textfeldwerte zu finden.
  > Verwenden Sie stattdessen die Übereinstimmungsabfrage, um nach Textfeldwerten zu suchen.
- query_string – Gibt Dokumente zurück, die auf einer angegebenen Abfragezeichenfolge basieren, wobei ein Parser mit einer strengen Syntax (Lucene-Syntax) verwendet wird.

  Diese Abfrage verwendet eine Syntax, um die bereitgestellte Abfragezeichenfolge basierend auf Operatoren wie AND oder NOT zu analysieren und zu trennen. Die Abfrage analysiert dann jeden getrennten Text unabhängig voneinander, bevor übereinstimmende Dokumente zurückgegeben werden.

Mit der `query_string`-Abfrage können Sie eine komplexe Suche erstellen, die Platzhalterzeichen, Suchen in mehreren Feldern usw. enthält. Obwohl diese Abfrage vielseitig ist, ist Sie strikt und gibt bei einer ungültigen Syntax in der Abfragezeichenfolge einen Fehler.

> **Warning**
>
> Da bei einer ungültigen Syntax ein Fehler zurückgegeben wird, empfiehlt es sich nicht, die `query_string`-Abfrage für Suchfelder zu verwenden.
> Wenn Sie keine Abfragesyntax unterstützen müssen, sollten Sie die `match`-Abfrage verwenden. Wenn Sie die Funktionen einer Abfragesyntax benötigen, verwenden Sie die `simple_query_string`-Abfrage. Diese ist weniger strikt.

- **`field`**— Das Feld, OpenSearch in dem die Suche ausgeführt werden soll. Dies kann nur weggelassen werden, wenn es `queryType` gestattet (wie z. B. bei `simple_query_string` und `query_string`). In diesem Fall wird die Suche für alle Felder durchgeführt. In Gremlin ist dies implizit.

  Es können mehrere Felder können angegeben werden, sofern die Abfrage dies zulässt, wie z. B. `simple_query_string` und `query_string`.
- **`query`**— (Erforderlich) Die Abfrage, gegen die ausgeführt werden soll OpenSearch. Der Inhalt dieses Feldes kann je nach queryType unterschiedlich sein. Verschiedene queryTypes akzeptieren unterschiedliche Syntaxregeln, wie zum Beispiel `Regexp`. In Gremlin ist `query` implizit.
- **`maxResults`** – Die maximale Anzahl der zurückzugebenden Ergebnisse. Die Standardeinstellung ist die `index.max_result_window` OpenSearch Einstellung, die selbst standardmäßig 10.000 ist. Der `maxResults`-Parameter kann eine beliebige Zahl angeben, die niedriger ist.

  > **Important**
  >
  > Wenn Sie einen höheren Wert als den OpenSearch `index.max_result_window` Wert festlegen `maxResults` und versuchen, mehr als `index.max_result_window` Ergebnisse abzurufen, OpenSearch schlägt dies mit einem `Result window is too large` Fehler fehl. Allerdings geht Neptune damit vorsichtig um, ohne den Fehler zu propagieren. Beachten Sie dies, wenn Sie versuchen, mehr als `index.max_result_window` Ergebnisse abzurufen.
- **`minScore`** – Der Mindestwert, den ein Suchergebnis für die Rückgabe haben muss. Eine Erläuterung der Ergebnisbewertung finden Sie in der Dokumentation zur OpenSearch Relevanz.

- **`batchSize`** – Neptune ruft immer Daten in Stapeln ab (die Standardstapelgröße ist 100). Sie können mit diesem Parameter die Leistung optimieren. Die Batchgröße darf die `index.max_result_window` OpenSearch Einstellung nicht überschreiten, die standardmäßig auf 10.000 eingestellt ist.
- **`sortBy`**— Ein optionaler Parameter, mit dem Sie die zurückgegebenen OpenSearch Ergebnisse nach einer der folgenden Kriterien sortieren können:
  - Ein bestimmtes Zeichenfolgenfeld im Dokument –

    In einer SPARQL-Abfrage können Sie beispielsweise Folgendes angeben:

    ```
    neptune-fts:config neptune-fts:sortBy foaf:name .
    ```

    In einer ähnlichen Gremlin-Abfrage können Sie Folgendes angeben:

    ```
    .withSideEffect('Neptune#fts.sortBy', 'name')
    ```

  - Ein bestimmtes Nicht-Zeichenfolgenfeld (*`long`*, *`double`* usw.) –

    Beachten Sie, dass Sie beim Sortieren nach einem Nicht-Zeichenfolgenfeld `.value` an den Feldnamen anhängen müssen, um es von einem Zeichenfolgenfeld zu unterscheiden.

    In einer SPARQL-Abfrage können Sie beispielsweise Folgendes angeben:

    ```
    neptune-fts:config neptune-fts:sortBy foaf:name.value .
    ```

    In einer ähnlichen Gremlin-Abfrage können Sie Folgendes angeben:

    ```
    .withSideEffect('Neptune#fts.sortBy', 'name.value')
    ```

  - `score` – Sortieren nach Übereinstimmungsergebnis (Standardeinstellung).

    Wenn der Parameter `sortOrder`, aber nicht `sortBy` vorhanden ist, werden die Ergebnisse in der durch `sortOrder` angegebenen Reihenfolge nach `score` sortiert.

  - `id` – Sortieren nach ID, d. h. SPARQL-Subject-URI oder Gremlin-Vertex oder Edge-ID.

    In einer SPARQL-Abfrage können Sie beispielsweise Folgendes angeben:

    ```
    neptune-fts:config neptune-fts:sortBy 'Neptune#fts.entity_id' .
    ```

In einer ähnlichen Gremlin-Abfrage können Sie Folgendes angeben:

```
    .withSideEffect('Neptune#fts.sortBy', 'Neptune#fts.entity_id')
```

- `label` – Sortieren nach Etikett.

  In einer SPARQL-Abfrage können Sie beispielsweise Folgendes angeben:

  ```
      neptune-fts:config neptune-fts:sortBy 'Neptune#fts.entity_type' .
  ```

  In einer ähnlichen Gremlin-Abfrage können Sie Folgendes angeben:

  ```
      .withSideEffect('Neptune#fts.sortBy', 'Neptune#fts.entity_type')
  ```

- `doc_type` – Sortieren nach Dokumenttyp (SPARQL oder Gremlin).

  In einer SPARQL-Abfrage können Sie beispielsweise Folgendes angeben:

  ```
      neptune-fts:config neptune-fts:sortBy 'Neptune#fts.document_type' .
  ```

  In einer ähnlichen Gremlin-Abfrage können Sie Folgendes angeben:

  ```
      .withSideEffect('Neptune#fts.sortBy', 'Neptune#fts.document_type')
  ```

  Standardmäßig sind die OpenSearch Ergebnisse nicht sortiert und ihre Reihenfolge ist nicht deterministisch, was bedeutet, dass dieselbe Abfrage bei jeder Ausführung Elemente in einer anderen Reihenfolge zurückgeben kann. Wenn die Ergebnismenge größer als `max_result_window` ist, kann daher jedes Mal eine ganz andere Teilmenge der Gesamtergebnisse zurückgegeben werden, wenn eine Abfrage ausgeführt wird. Durch die Sortierung können Sie jedoch die Ergebnisse verschiedener Durchläufe direkt vergleichbar machen.

  Wenn `sortOrder` von keinem `sortBy`-Parameter begleitet wird, wird die absteigende (`DESC`) Reihenfolge vom größten zum geringsten verwendet.

- **`sortOrder`**— Ein optionaler Parameter, mit dem Sie angeben können, ob die OpenSearch Ergebnisse vom kleinsten zum größten oder vom größten zum kleinsten sortiert werden (Standardeinstellung):

- `ASC` – Aufsteigende Reihenfolge, vom geringsten zum größten.
- `DESC` – Absteigende Reihenfolge, vom größten zum geringsten.

  Dies ist der Standardwert, der verwendet wird, wenn der Parameter `sortBy` vorhanden ist, aber keine `sortOrder` angegeben wird.

Wenn `sortBy` weder noch vorhanden `sortOrder` ist, werden die OpenSearch Ergebnisse standardmäßig nicht sortiert.

OpenSearch Indexierung ohne Zeichenketten in Amazon Neptune

Die OpenSearch Nicht-String-Indizierung in Amazon Neptune ermöglicht die Replikation von Nicht-String-Werten für Prädikate mithilfe des Stream-Pollers. OpenSearch Alle Prädikatwerte, die sicher in ein entsprechendes Mapping oder einen entsprechenden Datentyp konvertiert werden können, werden anschließend repliziert. OpenSearch OpenSearch

Damit die Indexierung ohne Zeichenketten für einen neuen Stack aktiviert werden kann, muss das `Enable Non-String Indexing` Flag in der CloudFormation Vorlage auf gesetzt sein. `true` Dies ist die Standardeinstellung. Um einen vorhandenen Stack so zu aktualisieren, dass er die Nicht-Zeichenfolgen-Indizierung unterstützt, siehe unten unter Aktualisieren eines vorhandenen Stacks.

> **Note**
>
> - Es ist empfehlenswert, die Nicht-Zeichenfolgen-Indizierung nicht für Engine-Versionen zu aktivieren, die älter sind als **`1.0.4.2`**.
> - OpenSearch Abfragen, die reguläre Ausdrücke für Feldnamen verwenden, die mehreren Feldern entsprechen, von denen einige Zeichenkettenwerte und andere Nichtzeichenfolgenwerte enthalten, schlagen mit einem Fehler fehl. Dies passiert auch, wenn Volltext-Suchanfragen in Neptune diesen Typ haben.
> - Wenn Sie nach einem Nicht-Zeichenfolgenfeld sortieren, hängen Sie „.value“ an den Feldnamen an, um das Feld von einem Zeichenfolgenfeld zu unterscheiden.

Inhalt

- Aktualisieren eines vorhandenen Neptune-Volltextsuchstacks zur Unterstützung der Nicht-Zeichenfolgen-Indizierung
- Filtern, welche Felder in der Neptune-Volltextsuche indiziert werden
  - Filtern nach Eigenschaften- oder Prädikatnamen
  - Filtern nach Eigenschaften- oder Prädikatstyp
- Zuordnung von SPARQL- und Gremlin-Datentypen zu OpenSearch
- Validierung von Datenzuordnungen
- OpenSearch Beispielabfragen ohne Zeichenfolge in Neptune
  - Abrufen alle Scheitelpunkte, deren Alter über 30 liegt und deren Name mit „Si“ beginnt
  - Abrufen aller Knoten mit einem Alter zwischen 10 und 50 und einem Namen, der unscharf mit „Ronka“ übereinstimmt
  - Abrufen aller Knoten mit einem Zeitstempel, der in die letzten 25 Tage fällt
  - Abrufen aller Knoten mit einem Zeitstempel, der in ein bestimmtes Jahr und einen bestimmten Monat fällt

Aktualisieren eines vorhandenen Neptune-Volltextsuchstacks zur Unterstützung der Nicht-Zeichenfolgen-Indizierung

Wenn Sie die Neptune-Volltextsuche bereits verwenden, müssen Sie die folgenden Schritte ausführen, um die Nicht-Zeichenfolgen-Indizierung zu unterstützen:

1. Halten Sie die Lambda-Funktion des Stream-Pollers an. Dadurch wird sichergestellt, dass beim Export keine neuen Updates kopiert werden. Deaktivieren Sie dazu die Cloud-Ereignisregel, die die Lambda-Funktion aufruft:

   - Navigieren Sie im AWS-Managementkonsole zu CloudWatch.
   - Wählen Sie Regeln aus.
   - Wählen Sie die Regel mit dem Namen des Lambda-Stream-Pollers aus.
   - Wählen Sie Deaktivieren aus, um die Regel vorübergehend zu deaktivieren.

2. Löscht den aktuellen Neptun-Index in. OpenSearch Verwenden Sie die folgende `curl` Abfrage, um den `amazon_neptune` Index aus Ihrem OpenSearch Cluster zu löschen:

   ```
   curl -X DELETE "your OpenSearch endpoint/amazon_neptune"
   ```

3. Starten Sie einen einmaligen Export von Neptune nach. OpenSearch Es empfiehlt sich, an dieser Stelle einen neuen OpenSearch Stack einzurichten, sodass neue Artefakte für den Poller, der den Export durchführt, aufgenommen werden.

   Folgen Sie den hier aufgeführten Schritten GitHub, um den einmaligen Export Ihrer Neptun-Daten in zu starten. OpenSearch

4. Aktualisieren Sie die Lambda-Artefakte für den vorhandenen Stream-Poller. Nachdem der Export der Neptun-Daten nach erfolgreich abgeschlossen OpenSearch wurde, führen Sie die folgenden Schritte aus:

   - Navigieren Sie im AWS-Managementkonsole zu. CloudFormation
   - Wählen Sie den übergeordneten CloudFormation Hauptstapel aus.
   - Wählen Sie die Option Aktualisieren für diesen Stack aus.
   - Wählen Sie Aktuelle Vorlage aus den Optionen ersetzen.
   - Wählen Sie unter „Vorlagenquelle“ die Option Amazon S3 URL aus.
   - Geben Sie für die Amazon-S3-URL Folgendes ein:

     ```
     https://aws-neptune-customer-samples.s3.amazonaws.com/neptune-stream/
     neptune_to_elastic_search.json
     ```

   - Wählen Sie Weiter, ohne einen der CloudFormation Parameter zu ändern.
   - Wählen Sie Stack aktualisieren aus. CloudFormation ersetzt die Lambda-Code-Artefakte für den Stream-Poller durch die neuesten Artefakte.

5. Starten Sie den Stream-Poller erneut. Aktivieren Sie dazu die entsprechende CloudWatch Regel:

   - Navigieren Sie im AWS-Managementkonsole zu CloudWatch.
   - Wählen Sie Regeln aus.
   - Wählen Sie die Regel mit dem Namen des Lambda-Stream-Pollers aus.
   - Wählen Sie Aktivieren.

Filtern, welche Felder in der Neptune-Volltextsuche indiziert werden

In den CloudFormation Vorlagendetails gibt es zwei Felder, in denen Sie Eigenschafts- oder Prädikatschlüssel oder Datentypen angeben können, die von der Indizierung ausgeschlossen werden sollen: OpenSearch

Filtern nach Eigenschaften- oder Prädikatnamen

Sie können den optionalen CloudFormation Vorlagenparameter named verwenden, `Properties to exclude from being inserted into Elastic Search Index` um eine durch Kommas getrennte Liste von Eigenschafts- oder Prädikatschlüsseln bereitzustellen, die von der Indizierung ausgeschlossen werden sollen. OpenSearch

Angenommen, Sie haben diesen Parameter auf bob festgelegt:

```
"Properties to exclude from being inserted into Elastic Search Index" : bob
```

In diesem Fall würde der Stream-Datensatz der folgenden Gremlin-Aktualisierungsabfrage gelöscht, anstatt in den Index aufgenommen zu werden:

```
g.V("1").property("bob", "test")
```

In ähnlicher Weise könnten Sie den Parameter auf `http://my/example#bob` setzen:

```
"Properties to exclude from being inserted into Elastic Search Index" : http://my/
example#bob
```

In diesem Fall würde der Stream-Datensatz der folgenden SPARQL-Aktualisierungsabfrage gelöscht, anstatt in den Index aufgenommen zu werden:

```
PREFIX ex: <http://my/example#>
INSERT DATA { ex:s1 ex:bob "test"}.
```

Wenn Sie in diesen CloudFormation Vorlagenparameter nichts eingeben, werden alle Eigenschaftsschlüssel, die nicht anderweitig ausgeschlossen wurden, indexiert.

Filtern nach Eigenschaften- oder Prädikatstyp

Sie können den optionalen CloudFormation Vorlagenparameter named verwenden, `Datatypes to exclude from being inserted into Elastic Search Index` um eine durch Kommas getrennte Liste von Eigenschaften- oder Prädikatwert-Datentypen bereitzustellen, die von der Indizierung ausgeschlossen werden sollen. OpenSearch

Für SPARQL müssen Sie nicht den vollständigen URI vom Typ XSD auflisten, Sie können einfach das Datentyp-Token auflisten. Gültige Datentyp-Token, die Sie auflisten können, sind:

- `string`
- `boolean`
- `float`
- `double`
- `dateTime`
- `date`
- `time`
- `byte`
- `short`
- `int`
- `long`
- `decimal`
- `integer`
- `nonNegativeInteger`
- `nonPositiveInteger`
- `negativeInteger`
- `unsignedByte`
- `unsignedShort`
- `unsignedInt`
- `unsignedLong`

Für Gremlin sind folgende Datentypen zum Auflisten gültig:

- `string`
- `date`
- `bool`
- `byte`
- `short`
- `int`

- `long`
- `float`
- `double`

Angenommen, Sie haben diesen Parameter auf `string` festgelegt:

```
"Datatypes to exclude from being inserted into Elastic Search Index" : string
```

In diesem Fall würde der Stream-Datensatz der folgenden Gremlin-Aktualisierungsabfrage gelöscht, anstatt in den Index aufgenommen zu werden:

```
g.V("1").property("myStringval", "testvalue")
```

In ähnlicher Weise könnten Sie den Parameter auf `int` setzen:

```
"Datatypes to exclude from being inserted into Elastic Search Index" : int
```

In diesem Fall würde der Stream-Datensatz der folgenden SPARQL-Aktualisierungsabfrage gelöscht, anstatt in den Index aufgenommen zu werden:

```
PREFIX ex: <http://my/example#>
PREFIX xsd:<http://www.w3.org/2001/XMLSchema#>
INSERT DATA { ex:s1 ex:bob "11"^^xsd:int }.
```

Wenn Sie in diesen CloudFormation Vorlagenparameter nichts eingeben, werden alle Eigenschaften indexiert, deren Werte sicher in Äquivalente konvertiert werden können. OpenSearch Aufgelistete Typen, die von der Abfragesprache nicht unterstützt werden, werden ignoriert.

Zuordnung von SPARQL- und Gremlin-Datentypen zu OpenSearch

Neue Datentypzuordnungen in OpenSearch werden auf der Grundlage des Datentyps erstellt, der in der Eigenschaft oder dem Objekt verwendet wird. Da einige Felder Werte unterschiedlichen Typs enthalten, kann es sein, dass bei der ersten Zuordnung einige Werte des Felds ausgeschlossen werden.

Neptun-Datentypen werden Datentypen wie folgt zugeordnet OpenSearch :

SPARQL-Typen	Gremlin-Typen	OpenSearch Typen
XSD:int XSD:unsignedInt XSD:integer XSD:byte XSD:unsignedByte XSD:short XSD:unsignedShort XSD:long XSD:unsignedLong	byte short int long	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
Benutzerdefinierter Datentyp	N/A	text
Jeder andere Datentyp	N/A	text

Die folgende Gremlin-Aktualisierungsabfrage bewirkt beispielsweise, dass ein neues Mapping für „NewField“ hinzugefügt wird, und zwar: OpenSearch `{ "type" : "double" }`

```
g.V("1").property("newField" 10.5)
```

In ähnlicher Weise bewirkt die folgende SPARQL-Aktualisierungsabfrage, dass ein neues Mapping für „ex:byte“ hinzugefügt wird, und zwar: OpenSearch `{ "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**
>
> Wie Sie sehen können, OpenSearch kann ein Objekt, das von Neptune auf zugeordnet wurde, am Ende einen anderen Datentyp haben OpenSearch als in Neptune. Es gibt jedoch ein explizites Textfeld in „datatype“ OpenSearch, das den Datentyp aufzeichnet, den das Element in Neptune hat.

Validierung von Datenzuordnungen

Daten werden mit diesem Verfahren OpenSearch von Neptune aus repliziert:

- Wenn eine Zuordnung für das fragliche Feld bereits vorhanden ist in: OpenSearch
  - Wenn die Daten mithilfe von Datenvalidierungsregeln sicher in die vorhandene Zuordnung konvertiert werden können, speichern Sie das Feld in OpenSearch.
  - Andernfalls löschen Sie den entsprechenden Stream-Update-Datensatz.
- Wenn für das fragliche Feld keine Zuordnung vorhanden ist, suchen Sie einen OpenSearch Datentyp, der dem Datentyp des Felds in Neptune entspricht.
  - Wenn die Felddaten mithilfe von Datenvalidierungsregeln sicher in den OpenSearch Datentyp konvertiert werden können, speichern Sie die neue Zuordnung und die Felddaten in. OpenSearch
  - Andernfalls löschen Sie den entsprechenden Stream-Update-Datensatz.

Die Werte werden anhand äquivalenter OpenSearch Typen oder vorhandener OpenSearch Zuordnungen und nicht anhand der Neptuntypen validiert. Beispielsweise erfolgt die Validierung des Werts `"123"` in `"123"^^xsd:int` anhand des `long`-Typs und nicht anhand des `int`-Typs.

Obwohl Neptune versucht, alle Daten zu replizieren OpenSearch, gibt es Fälle, in denen sich die Datentypen in völlig von denen in Neptune unterscheiden, und in solchen Fällen OpenSearch werden Datensätze übersprungen, anstatt indexiert zu werden. OpenSearch

In Neptune kann eine Eigenschaft beispielsweise mehrere Werte unterschiedlichen Typs haben, wohingegen in OpenSearch einem Feld im gesamten Index denselben Typ haben muss.

Wenn Sie Debug-Logs aktivieren, können Sie sehen, welche Datensätze beim Export von Neptune nach gelöscht wurden. OpenSearch Ein Beispiel für einen Debug-Protokolleintrag ist:

```
Dropping Record : Data type not a valid Gremlin type
<Record>
```

Datentypen werden wie folgt validiert:

- **`text`**— Alle Werte in Neptune können sicher dem Text in zugeordnet werden. OpenSearch
- **`long`**— Die folgenden Regeln für Neptun-Datentypen gelten, wenn der OpenSearch Mapping-Typ lang ist (in den folgenden Beispielen wird davon ausgegangen, dass er einen `long` Mapping-Typ `"testLong"` hat):
  - `boolean` – Ungültig, kann nicht konvertiert werden und der entsprechende Datensatz zum Stream-Update wird gelöscht.

    Ungültige Gremlin-Beispiele sind:

    ```
    "testLong" : true.
    "testLong" : false.
    ```

    Ungültige SPARQL-Beispiele sind:

    ```
    ":testLong" : "true"^^xsd:boolean
    ":testLong" : "false"^^xsd:boolean
    ```
  - `datetime` – Ungültig, kann nicht konvertiert werden und der entsprechende Datensatz zum Stream-Update wird gelöscht.

    Ein ungültiges Gremlin-Beispiel ist:

```
":testLong" :  datetime('2018-11-04T00:00:00').
```

Ein ungültiges SPARQL-Beispiel ist:

```
":testLong" : "2016-01-01"^^xsd:date
```

- `float`,`double`, oder `decimal` — Wenn der Wert in Neptune eine Ganzzahl ist, die in 64 Bit passen kann, ist er gültig und wird als Long gespeichert, aber wenn er einen Bruchteil hat oder a oder an ist oder größer OpenSearch als 9.223.372.036.854.775.807 oder kleiner als -9.223.372.036.854.775.808 ist`INF`, dann ist er nicht gültig und der entsprechende Stream Der Aktualisierungsdatensatz wurde gelöscht. `NaN`

  Gültige Gremlin-Beispiele sind:

  ```
  "testLong" :  145.0.
  ":testLong" :  123
  ":testLong" :  -9223372036854775807
  ```

  Gültige SPARQL-Beispiele sind:

  ```
  ":testLong" : "145.0"^^xsd:float
  ":testLong" :  145.0
  ":testLong" : "145.0"^^xsd:double
  ":testLong" : "145.0"^^xsd:decimal
  ":testLong" : "-9223372036854775807"
  ```

  Ungültige Gremlin-Beispiele sind:

  ```
  "testLong" :  123.45
  ":testLong" :  9223372036854775900
  ```

  Ungültige SPARQL-Beispiele sind:

  ```
  ":testLong" :  123.45
  ":testLong" :  9223372036854775900
  ":testLong" : "123.45"^^xsd:float
  ":testLong" : "123.45"^^xsd:double
  ":testLong" : "123.45"^^xsd:decimal
  ```

- `string`— Wenn der Wert in Neptune eine Zeichenkettendarstellung einer Ganzzahl ist, die in einer 64-Bit-Ganzzahl enthalten sein kann, dann ist er gültig und wird in ein `long` In umgewandelt. OpenSearch Jeder andere Zeichenfolgenwert ist für eine Elasticsearch-`long`-Zuordnung ungültig und der entsprechende Stream-Update-Datensatz wird gelöscht.

  Gültige Gremlin-Beispiele sind:

  ```
  "testLong" :  "123".
  ":testLong" :  "145.0"
  ":testLong" :  "-9223372036854775807"
  ```

  Gültige SPARQL-Beispiele sind:

  ```
  ":testLong" : "145.0"^^xsd:string
  ":testLong" : "-9223372036854775807"^^xsd:string
  ```

  Ungültige Gremlin-Beispiele sind:

  ```
  "testLong" :  "123.45"
  ":testLong" :  "9223372036854775900"
  ":testLong" :  "abc"
  ```

  Ungültige SPARQL-Beispiele sind:

  ```
  ":testLong" : "123.45"^^xsd:string
  ":testLong" : "abc"
  ":testLong" : "9223372036854775900"^^xsd:string
  ```

- **`double`**— Wenn der OpenSearch Mapping-Typ ist`double`, gelten die folgenden Regeln (hier wird davon ausgegangen, dass das Feld „TestDouble“ ein `double` Mapping enthält): OpenSearch

  - `boolean` – Ungültig, kann nicht konvertiert werden und der entsprechende Datensatz zum Stream-Update wird gelöscht.

    Ungültige Gremlin-Beispiele sind:

    ```
    "testDouble" : true.
    "testDouble" : false.
    ```

    Ungültige SPARQL-Beispiele sind:

```
  ":testDouble" : "true"^^xsd:boolean
  ":testDouble" : "false"^^xsd:boolean
```

- `datetime` – Ungültig, kann nicht konvertiert werden und der entsprechende Datensatz zum Stream-Update wird gelöscht.

  Ein ungültiges Gremlin-Beispiel ist:

  ```
    ":testDouble" :  datetime('2018-11-04T00:00:00').
  ```

  Ein ungültiges SPARQL-Beispiel ist:

  ```
    ":testDouble" : "2016-01-01"^^xsd:date
  ```

- Gleitkomma-`NaN` oder `-INF` – Wenn der Wert in SPARQL ein Gleitkomma-`NaN` oder `-INF` ist, dann ist er nicht gültig und der entsprechende Stream-Update-Datensatz wird gelöscht.

  Ungültige SPARQL-Beispiele sind:

  ```
  "  :testDouble" : "NaN"^^xsd:float
    ":testDouble" : "NaN"^^double
    ":testDouble" : "INF"^^double
    ":testDouble" : "-INF"^^double
  ```

- Zahl oder numerische Zeichenfolge — Wenn der Wert in Neptune eine andere Zahl oder numerische Zeichenkettendarstellung einer Zahl ist, die sicher als a ausgedrückt werden kann`double`, dann ist er gültig und wird in ein In umgewandelt. `double` OpenSearch Jeder andere Zeichenkettenwert ist für eine OpenSearch `double` Zuordnung ungültig und der entsprechende Datensatz zur Stream-Aktualisierung wird gelöscht.

  Gültige Gremlin-Beispiele sind:

  ```
    "testDouble" :  123
    ":testDouble" :  "123"
    ":testDouble" :  145.67
    ":testDouble" :  "145.67"
  ```

  Gültige SPARQL-Beispiele sind:

```
":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
```

Ein ungültiges Gremlin-Beispiel ist:

```
":testDouble" :  "abc"
```

Ein ungültiges SPARQL-Beispiel ist:

```
":testDouble" : "abc"
```

- **`date`**— Wenn der OpenSearch Mapping-Typ ist`date`, sind Neptune `date` und der `dateTime` Wert gültig, ebenso wie jeder Zeichenkettenwert, der erfolgreich in ein Format geparst werden kann. `dateTime`

  Gültige Beispiele in Gremlin oder SPARQL sind:

```
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"
```

Ungültige Beispiele sind:

```
  123.45
  True
  "abc"
```

OpenSearch Beispielabfragen ohne Zeichenfolge in Neptune

Neptune unterstützt derzeit keine direkten OpenSearch Bereichsabfragen. Sie können jedoch den gleichen Effekt mit der Lucene-Syntax und query-type="query_string“ erzielen, wie Sie in den folgenden Beispielabfragen sehen können.

Abrufen alle Scheitelpunkte, deren Alter über 30 liegt und deren Name mit „Si“ beginnt

In 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*');
```

In 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 .
  }
}
```

Hier wird der Kürze halber "*foaf*age anstelle des vollständigen URI verwendet. Dieser reguläre Ausdruck ruft alle Felder ab, die foaf und age im URI enthalten.

Abrufen aller Knoten mit einem Alter zwischen 10 und 50 und einem Namen, der unscharf mit „Ronka“ übereinstimmt

In 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~');
```

In 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 .
  }
}
```

Abrufen aller Knoten mit einem Zeitstempel, der in die letzten 25 Tage fällt

In 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');
```

In 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 .
  }
}
```

Abrufen aller Knoten mit einem Zeitstempel, der in ein bestimmtes Jahr und einen bestimmten Monat fällt

In Gremlin unter Verwendung von mathematischen Datumsausdrücken in der Lucene-Syntax für Dezember 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');
```

Eine Gremlin-Alternative:

```
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 Ausführung von Abfragen in Amazon Neptune

In einer Abfrage, die Folgendes beinhaltet full-text-search, versucht Neptune, die full-text-search Aufrufe an die erste Stelle zu setzen, bevor andere Teile der Abfrage ausgeführt werden. Dadurch wird die Anzahl der Aufrufe reduziert OpenSearch und in den meisten Fällen die Leistung erheblich verbessert. Dies ist jedoch keineswegs eine hard-and-fast Regel. Es gibt Situationen, in denen beispielsweise ein `PatternNode` oder `UnionNode` einem Volltextsuchaufruf vorausgehen kann.

Sehen Sie sich beispielsweise die folgende Gremlin-Datenbankabfrage an eine Datenbank an, die 100 000 Instances von `Person` enthält:

```
g.withSideEffect('Neptune#fts.endpoint', 'your-es-endpoint-URL')
 .hasLabel('Person')
 .has('name', 'Neptune#fts marcello~');
```

Wenn diese Abfrage in der Reihenfolge ausgeführt würde, in der die Schritte erscheinen, würden 100.000 Lösungen zusammenfließen OpenSearch, was zu Hunderten von OpenSearch Aufrufen

führen würde. Tatsächlich ruft Neptune OpenSearch zuerst an und verbindet dann die Ergebnisse mit den Neptun-Ergebnissen. In den meisten Fällen erfolgt dies viel schneller als die Ausführung der Abfrage in der ursprünglichen Reihenfolge.

Sie können diese Neuordnung der Abfrageschrittausführung mit dem Abfragehinweis „noReordering“ verhindern:

```
g.withSideEffect('Neptune#fts.endpoint', 'your-es-endpoint-URL')
 .withSideEffect('Neptune#noReordering', true)
 .hasLabel('Person')
 .has('name', 'Neptune#fts marcello~');
```

In diesem zweiten Fall wird zuerst der Schritt `.hasLabel` und anschließend der Schritt `.has('name', 'Neptune#fts marcello~')` ausgeführt.

Ein weiteres Beispiel ist eine SPARQL-Abfrage für gleiche Art von Daten:

```
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 .
  }
}
```

Neptune führt hier erneut zuerst den `SERVICE`-Teil der Abfrage aus und führt dann die Ergebnisse mit den `Person`-Daten zusammen. Sie können dieses Verhalten mit dem Abfragehinweis „joinOrder“ unterdrücken:

```
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 .
  }
}
```

Auch in der zweiten Abfrage werden die Teile in der Reihenfolge ausgeführt, in der sie in der Abfrage erscheinen.

> **Note**
>
> Wenn Sie einen Opensearch-Alias über einen Index abfragen, anstatt direkt einen Opensearch-Index abzufragen, kann dies zu falschen Ergebnissen führen. Sie sollten den Opensearch-Index direkt abfragen und nicht den Alias.

Beispiele für SPARQL-Volltextsuchabfragen in Neptune

Im Folgenden finden Sie einige Beispiele für SPARQL-Volltextsuchabfragen in Amazon Neptune.

Beispiel für eine SPARQL-Übereinstimmungsabfrage

```
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 .
  }
}
```

Beispiel für eine SPARQL-Präfix-Abfrage

```
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 .
  }
}
```

Beispiel für eine SPARQL-Fuzzy-Abfrage

```
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 .
  }
}
```

Beispiel für eine SPARQL-Begriffsabfrage

```
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 .
  }
}
```

Beispiel für eine SPARQL-query_string-Abfrage

Diese Abfrage gibt mehrere Felder an.

```
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 .
  }
}
```

Beispiel für eine SPARQL-simple_query_string-Abfrage

Die folgende Abfrage gibt Felder mit dem Platzhalterzeichen („*“) an.

```
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 .
  }
}
```

Beispiel für eine SPARQL-Abfrage mit Sortierung nach Zeichenfolgenfeld

```
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 .
  }
}
```

Beispiel für eine SPARQL-Abfrage mit Sortierung nach Nicht-Zeichenfolgenfeld

```
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 .
  }
}
```

Beispiel für eine SPARQL-Abfrage mit Sortierung nach ID

```
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 .
  }
}
```

Beispiel für eine SPARQL-Abfrage mit Sortierung nach Etikett

```
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 .
  }
}
```

Beispiel für eine SPARQL-Abfrage mit Sortierung nach doc_type

```
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 .
  }
}
```

Beispiel für die Verwendung der Lucene-Syntax in SPARQL

Die Lucene-Syntax wird nur für `query_string` Abfragen in OpenSearch unterstützt.

```
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 .
  }
}
```

Verwenden der Neptune-Volltextsuche in Gremlin-Abfragen

`NeptuneSearchStep` aktiviert Volltextsuchabfragen für den Teil eines Gremlin-Traversals, der nicht in Neptune-Schritte konvertiert wird. Sehen Sie sich zum Beispiel folgende Abfrage an:

```
g.withSideEffect("Neptune#fts.endpoint", "your-es-endpoint-URL")
  .V()
      .tail(100)
      .has("name", "Neptune#fts mark*")            <== # Limit the search on name
```

Diese Abfrage wird in die folgende optimierte Traversale in Neptune konvertiert:

```
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}
    }
}]
```

Die folgenden Beispiele sind Gremlin-Abfragen anhand von Flugstreckendaten:

Grundlegende Gremlin-**`match`**-Übereinstimmungsabfrage mit Groß- und Kleinschreibung

```
g.withSideEffect("Neptune#fts.endpoint",
```

```
                  "your-OpenSearch-endpoint-URL")
  .withSideEffect('Neptune#fts.queryType', 'match')
  .V().has("city","Neptune#fts dallas")

==>v[186]
==>v[8]
```

Gremlin-**match**-Abfrage

```
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]
```

Gremlin-**fuzzy**-Abfrage

```
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
```

Gremlin-**query_string**-Fuzzy-Abfrage

```
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
```

Gremlin-**query_string**-Abfrage mit regulären Ausdrücken

```
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
```

Gremlin-Hybridabfrage

Diese Abfrage verwendet einen internen Neptun-Index und den OpenSearch Index in derselben Abfrage.

```
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
```

Einfaches Beispiel für eine Gremlin-Volltextsuche

```
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]
```

Gremlin-Abfrage mit **`query_string`** mit den Operatoren „+“ und „-“

Obwohl der `query_string`-Abfragetyp viel weniger strenger ist als der standardmäßige `simple_query_string`-Typ, ermöglicht er genauere Abfragen. Bei der ersten der folgenden Abfragen wird `query_string` verwendet. Bei der zweiten der standardmäßige `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]
```

Beachten Sie, wie `simple_query_string` in den folgenden Beispielen die Operatoren „+“ und „-“ unbemerkt ignoriert:

```
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]
```

Gremlin-**`query_string`**-Abfrage mit den Operatoren **`AND`** und **`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]
```

Gremlin-**`term`**-Abfrage

```
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]
```

Gremlin-**`prefix`**-Abfrage

```
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]
```

Verwenden der Lucene-Syntax in Neptune Gremlin

In Neptune Gremlin können Sie auch sehr leistungsstarke Abfragen mit der Lucene-Abfragesyntax schreiben. Beachten Sie, dass die Lucene-Syntax nur für `query_string` Abfragen in unterstützt wird. OpenSearch

Setzen Sie die folgenden Daten voraus:

```
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")
```

Mit der Lucene-Syntax, die aufgerufen wird, wenn der `queryType` auf `query_string` eingestellt ist, können Sie diese Daten wie folgt nach Vor- und Nachnamen durchsuchen:

```
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]
```

Beachten Sie, dass im obigen Schritt `has()` das Feld durch `"*"` ersetzt wird). Tatsächlich wird jeder dort platzierte Wert von den Feldern außer Kraft gesetzt, auf die Sie innerhalb der Abfrage zugreifen. Sie greifen mit `predicates.name.value,` auf das Namensfeld zu, da das Datenmodell so strukturiert ist.

Sie können wie folgt nach Name, Nachname und Bezeichnung suchen:

```
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]
```

Auf die Bezeichnung wird mit `entity_type` zugegriffen, da das Datenmodell so strukturiert ist.

Sie können auch Verschachtelungsbedingungen einschließen:

```
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]
```

Einfügen eines modernen Graphen 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')
```

Beispiel für eine Sortierung nach einem Zeichenfolgen-Feldwert

```
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')
```

Beispiel für eine Sortierung nach einem Nicht-Zeichenfolgen-Feldwert

```
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')
```

Beispiel für eine Sortierung nach einem ID-Feldwert

```
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')
```

Beispiel für eine Sortierung nach einem Etikett-Feldwert

```
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')
```

Beispiel für eine Sortierung nach einem **`document_type`**-Feldwert

```
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')
```

Fehlerbehebung bei der Neptune-Volltextsuche

> **Note**
>
> Wenn Sie eine differenzierte Zugriffskontrolle auf Ihrem OpenSearch Cluster aktiviert haben, müssen Sie die IAM-Authentifizierung auch in Ihrer Neptune-Datenbank aktivieren.

Informationen zur Diagnose von Problemen mit der Replikation von Neptune nach OpenSearch finden Sie in den CloudWatch Protokollen für Ihre Poller-Lambda-Funktion. Diese Protokolle enthalten Details zur Anzahl der aus dem Stream gelesenen Datensätze und zur Anzahl der Datensätze, in die erfolgreich repliziert wurden. OpenSearch

Sie können auch die LOGGING-Ebene für Ihre Lambda-Funktion ändern, indem Sie die Umgebungsvariable `LoggingLevel` ändern.

> **Note**
>
> Wenn diese `LoggingLevel` Option auf gesetzt ist`DEBUG`, können Sie zusätzliche Details anzeigen, z. B. gelöschte Stream-Datensätze und den Grund, warum jeder gelöscht wurde, während Sie Daten StreamPoller von Neptune nach replizieren. OpenSearch Dies kann nützlich sein, wenn Sie feststellen, dass Ihnen Datensätze fehlen.

Die Neptune Streams-Verbraucheranwendung veröffentlicht zwei Metriken CloudWatch , die Ihnen auch bei der Diagnose von Problemen helfen können:

- `StreamRecordsProcessed` – Die Anzahl der Datensätze, die von der Anwendung pro Zeiteinheit verarbeitet werden. Dies ist bei der Verfolgung der Anwendungsausführungsrate hilfreich.
- `StreamLagTime` – Die Zeitdifferenz in Millisekunden zwischen der aktuellen Zeit und der Commit-Zeit eines Streamdatensatzes, der verarbeitet wird. Diese Metrik zeigt an, wie stark die Consumer-Anwendung verzögert wird.

Darüber hinaus werden alle Metriken im Zusammenhang mit dem Replikationsprozess in einem Dashboard CloudWatch unter demselben Namen angezeigt, den Sie bei der Instanziierung der Anwendung mithilfe der Vorlage `ApplicationName` angegeben haben. CloudWatch

Sie können sich auch dafür entscheiden, einen CloudWatch Alarm zu erstellen, der ausgelöst wird, wenn die Abfrage mehr als zweimal hintereinander fehlschlägt. Dies erreichen Sie, indem Sie das `CreateCloudWatchAlarm`-Feld bei Anwendungsinstanziierung auf `true` festlegen. Geben Sie dann die E-Mail-Adressen an, die beim Auslösen des Alarms benachrichtigt werden sollen.

Problembehandlung eines Prozesses, der beim Lesen von Datensätzen aus dem Stream fehlschlägt

Wenn ein Prozess beim Lesen von Datensätzen aus dem Stream fehlschlägt, stellen Sie sicher, dass Folgendes vorhanden ist:

- Der Stream ist auf dem Cluster aktiviert.
- Der Neptune-Stream-Endpunkt hat das korrekteFormat:
  - Für Gremlin oder OpenCypher: `https://`*`your cluster endpoint`*`:`*`your cluster port`*`/propertygraph/stream` oder dessen Alias `https://`*`your cluster endpoint`*`:`*`your cluster port`*`/pg/stream`
  - Für SPARQL: `https://`*`your cluster endpoint`*`:`*`your cluster port`*`/sparql/stream`
- Der DynamoDB-Endpunkt ist für Ihre VPC konfiguriert.
- Der Überwachungsendpunkt ist für Ihre VPC-Subnetze konfiguriert.

Problembehandlung bei einem Prozess, der beim Schreiben von Daten fehlschlägt OpenSearch

Wenn ein Prozess beim Schreiben von Datensätzen fehlschlägt OpenSearch, stellen Sie sicher, dass Sie über Folgendes verfügen:

- Ihre Elasticsearch-Version ist 7.1 oder höher oder Opensearch 2.3 und höher.
- OpenSearch kann über die Poller Lambda-Funktion in Ihrer VPC aufgerufen werden.
- Die beigefügte Sicherheitsrichtlinie OpenSearch erlaubt eingehende Anfragen. HTTP/HTTPS

Behebung von out-of-sync Problemen zwischen Neptune und einem bestehenden OpenSearch Replikations-Setup

Sie können die folgenden Schritte ausführen, um eine Neptune-Datenbank und OpenSearch -Domain wieder mit den neuesten Daten zu synchronisieren, falls out-of-sync Probleme zwischen ihnen aufgrund einer `ExpiredStreamException` oder Datenbeschädigung auftreten.

Beachten Sie, dass dieser Ansatz alle Daten in der OpenSearch Domain löscht und sie mit dem aktuellen Status der Neptune-Datenbank erneut synchronisiert, sodass keine Daten in die Neptune-Datenbank neu geladen werden müssen.

1. Deaktivieren Sie den Replikationsprozess, wie unter Deaktivieren (Unterbrechen) des Stream-Poller-Prozesses beschrieben.
2. Löschen Sie den Neptun-Index auf der OpenSearch Domain mit dem folgenden Befehl:

   ```
   curl -X DELETE "(your OpenSearch endpoint)/amazon_neptune"
   ```

3. Erstellen Sie einen Klon der Datenbank (siehe Klonen von Datenbanken in Neptune).
4. Holen Sie sich die neueste `eventID` zu den Streams in der geklonten Datenbank, indem Sie einen Befehl dieser Art für den Streams-API-Endpunkt ausführen (weitere Informationen finden Sie unter Aufrufen der Neptune-Streams-REST-API):

   ```
   curl "https://(your neptune endpoint):(port)/(propertygraph or sparql)/stream?
   iteratorType=LATEST"
   ```

   Notieren Sie sich die Werte in den Feldern `commitNum` und `opNum` im `lastEventId`-Objekt in der Antwort.
5. Verwenden Sie das export-neptune-to-elasticsearchTool auf Github, um eine einmalige Synchronisation von der geklonten Datenbank zur Domain durchzuführen. OpenSearch
6. Gehen Sie zur DynamoDB-Tabelle für den Replikationsstack. Der Name der Tabelle ist der Anwendungsname, den Sie in der CloudFormation Vorlage angegeben haben (die Standardeinstellung ist`NeptuneStream`) mit einem `-LeaseTable` Suffix. Mit anderen Worten: Der Standardtabellenname ist `NeptuneStream-LeaseTable`.

   Sie können Tabellenzeilen durch Scannen untersuchen, da die Tabelle nur eine Zeile enthalten sollte. Nehmen Sie mit den Werten `commitNum` und `opNum`, die Sie oben aufgezeichnet haben, die folgenden Änderungen vor:

- Ändern Sie den Wert für das `checkpoint`-Feld in der Tabelle auf den Wert, den Sie für `commitNum` notiert haben.
- Ändern Sie den Wert für das `checkpointSubSequenceNumber`-Feld in der Tabelle auf den Wert, den Sie für `opNum` notiert haben.

7. Aktivieren Sie den Replikationsprozess erneut, wie unter Erneutes Aktivieren des Stream-Poller-Prozesses beschrieben.
8. Löschen Sie die geklonte Datenbank und den für das CloudFormation `export-neptune-to-elasticsearch` Tool erstellten Stack.

AWS Lambda Funktionen in Amazon Neptune verwenden

AWS Lambda Funktionen haben viele Verwendungsmöglichkeiten in Amazon Neptune Neptune-Anwendungen. Hier finden Sie allgemeine Anleitungen zur Verwendung von Lambda-Funktionen mit allen gängigen Gremlin-Treibern und Sprachvarianten sowie spezifische Beispiele für Lambda-Funktionen, die in Java, JavaScript und Python geschrieben wurden.

> **Note**
>
> Die beste Art, Lambda-Funktionen mit Neptune zu verwenden, hat sich mit den letzten Engine-Versionen geändert. Neptune ließ inaktive Verbindungen lange nach der Wiederverwendung eines Lambda-Ausführungskontextes offen, was zu Ressourcenlecks auf dem Server führen konnte. Um dies zu vermeiden, haben wir früher empfohlen, bei jedem Lambda-Aufruf eine Verbindung zu öffnen und zu schließen. Ab Engine-Version 1.0.3.0 wurde der Zeitrahmen (Timeout) für inaktive Verbindungen jedoch reduziert, so dass Verbindungen nicht mehr lecken, nachdem ein inaktiver Lambda-Ausführungskontext wiederverwendet wurde. Daher empfehlen wir jetzt, für die Dauer des Ausführungskontextes eine einzige Verbindung zu verwenden. Dies sollte etwas Fehlerbehandlung und back-off-and-retry Standardcode für den Umgang mit Verbindungen beinhalten, die unerwartet geschlossen werden.

Verwaltung von Gremlin-Verbindungen WebSocket in Funktionen AWS Lambda

Wenn Sie eine Gremlin-Sprachvariante verwenden, um Neptune abzufragen, stellt der Treiber über eine Verbindung eine Verbindung zur Datenbank her. WebSocket WebSockets sind so konzipiert, dass sie langlebige Client-Server-Verbindungsszenarien unterstützen. AWS Lambda ist dagegen darauf ausgelegt, relativ kurzlebige und zustandslose Hinrichtungen zu unterstützen. Diese Diskrepanz bei der Designphilosophie kann zu unerwarteten Problemen führen, wenn Lambda zur Abfrage von Neptune verwendet wird.

Eine AWS Lambda Funktion wird in einem Ausführungskontext ausgeführt, der die Funktion von anderen Funktionen isoliert. Der Ausführungskontext wird beim ersten Aufruf der Funktion erstellt und kann für nachfolgende Aufrufe derselben Funktion wiederverwendet werden.

Ein einziger Ausführungskontext wird jedoch niemals verwendet, um mehrere gleichzeitige Aufrufe der Funktion zu verarbeiten. Wenn Ihre Funktion gleichzeitig von mehreren Clients aufgerufen wird, erstellt Lambda für jede Instance der Funktion einen zusätzlichen Ausführungskontext. Alle diese neuen Ausführungskontexte können wiederum für nachfolgende Aufrufe der Funktion wiederverwendet werden.

Irgendwann recycelt Lambda Ausführungskontexte, insbesondere wenn sie einige Zeit inaktiv waren. AWS Lambda macht den Lebenszyklus des Ausführungskontextes, einschließlich der `Shutdown` Phasen `Invoke` und`Init`, über Lambda-Erweiterungen verfügbar. Mithilfe dieser Erweiterungen können Sie Code schreiben, der externe Ressourcen wie Datenbankverbindungen bereinigt, wenn der Ausführungskontext wiederverwendet wird.

Eine gängige bewährte Methode besteht darin, die Datenbankverbindung außerhalb der Lambda-Handler-Funktion zu öffnen, so dass sie bei jedem Handler-Aufruf wiederverwendet werden kann. Wenn die Datenbankverbindung irgendwann unterbrochen wird, können Sie die Verbindung innerhalb des Handlers erneut herstellen. Bei dieser Vorgehensweise besteht jedoch die Gefahr von Verbindungslecks. Wenn eine Verbindung im Leerlauf noch lange geöffnet bleibt, nachdem ein Ausführungskontext nicht mehr besteht, können Lambda-Aufrufszenarien mit intermittierenden oder kurzlebigen Lambda-Aufrufen nach und nach Verbindungen verlieren und Datenbankressourcen erschöpfen.

Die Verbindungslimits und Timeout-Werte von Neptune haben sich mit neueren Engine-Versionen geändert. Bisher unterstützte jede Instanz bis zu 60.000 WebSocket Verbindungen. Jetzt variiert die maximale Anzahl gleichzeitiger WebSocket Verbindungen pro Neptune-Instanz je nach Instanztyp.

Außerdem reduzierte Neptune ab der Engine-Version 1.0.3.0 das Leerlauf-Timeout für Verbindungen von einer Stunde auf etwa 20 Minuten. Wenn ein Client eine Verbindung nicht schließt, wird die Verbindung nach einem Leerlauf-Timeout von 20 bis 25 Minuten automatisch geschlossen. AWS Lambda dokumentiert nicht die Lebensdauer des Ausführungskontextes, aber Experimente zeigen, dass das neue Neptune-Verbindungstimeout gut zu inaktiven Lambda-Ausführungskontext-Timeouts passt. Wenn ein inaktiver Ausführungskontext recycelt wird, besteht eine gute Möglichkeit, dass seine Verbindung bereits von Neptune geschlossen wurde oder bald darauf geschlossen werden wird.

Empfehlungen für die Verwendung AWS Lambda mit Amazon Neptune Gremlin

Wir empfehlen jetzt, für die gesamte Lebensdauer eines Lambda-Ausführungskontextes eine einzige Verbindungs- und Graph-Traversal-Quelle zu verwenden, anstatt einer für jeden Funktionsaufruf

(jeder Funktionsaufruf verarbeitet nur eine Client-Anfrage). Da gleichzeitige Client-Anfragen von verschiedenen Funktions-Instances verarbeitet werden, die in separaten Ausführungskontexten ausgeführt werden, ist es nicht erforderlich, einen Pool von Verbindungen aufrechtzuerhalten, um gleichzeitige Anfragen innerhalb einer Funktions-Instance zu verarbeiten. Wenn der Gremlin-Treiber, den Sie verwenden, über einen Verbindungspool verfügt, konfigurieren Sie ihn so, dass er nur eine Verbindung verwendet.

Verwenden Sie bei jeder Abfrage eine Wiederholungslogik, um mit Verbindungsfehlern umzugehen. Obwohl das Ziel darin besteht, eine einzige Verbindung für die gesamte Lebensdauer eines Ausführungskontextes aufrechtzuerhalten, können unerwartete Netzwerkereignisse dazu führen, dass diese Verbindung abrupt beendet wird. Solche Verbindungsfehler äußern sich je nach verwendetem Treiber als unterschiedliche Fehler. Sie sollten Ihre Lambda-Funktion so programmieren, dass sie diese Verbindungsprobleme behebt und bei Bedarf versucht, die Verbindung wiederherzustellen.

Einige Gremlin-Treiber kümmern sich automatisch um erneute Verbindungen. Der Java-Treiber versucht beispielsweise automatisch, die Konnektivität zu Neptune für Ihren Client-Code wiederherzustellen. Mit diesem Treiber muss Ihr Funktionscode nur zurückgezogen werden und die Abfrage erneut versuchen. Die Python-Treiber JavaScript und die Python-Treiber implementieren dagegen keine automatische Verbindungslogik. Daher muss Ihr Funktionscode bei diesen Treibern versuchen, die Verbindung nach dem Zurückziehen erneut herzustellen, und die Abfrage erst wiederholen, wenn die Verbindung wiederhergestellt wurde.

Die Codebeispiele hier beinhalten Logik zur Wiederherstellung einer Verbindung, anstatt davon auszugehen, dass der Client dies übernimmt.

Empfehlungen für die Verwendung von Gremlin-Schreibanforderungen in Lambda

Wenn Ihre Lambda-Funktion Grafikdaten ändert, sollten Sie eine back-off-and-retry Strategie zur Behandlung der folgenden Ausnahmen in Betracht ziehen. Eine ausführliche Anleitung zur Entwicklung einer praktischen Wiederholungsstrategie finden Sie unter. the section called “Ausnahmen und Wiederholungen”

- **`ConcurrentModificationException`** – Die Neptune-Transaktionssemantik bedeutet, dass Schreibanforderungen manchmal mit einer `ConcurrentModificationException` fehlschlagen. Versuchen Sie in diesen Situationen einen exponentiellen back-off-based Wiederholungsmechanismus.

- **`ReadOnlyViolationException`** – Da sich die Cluster-Topologie aufgrund von geplanten oder ungeplanten Ereignissen jederzeit ändern kann, können Schreibzuständigkeiten von einer Instance im Cluster auf eine andere übertragen werden. Wenn Ihr Funktionscode versucht, eine Schreibanforderung an eine Instance zu senden, die nicht mehr die primäre (Writer-)Instance ist, schlägt die Anforderung mit einer `ReadOnlyViolationException` fehl. Schließen Sie in diesem Fall die bestehende Verbindung, stellen Sie erneut eine Verbindung zum Cluster-Endpunkt her und wiederholen Sie dann die Anforderung.

Wenn Sie eine back-off-and-retry Strategie zur Behandlung von Problemen mit Schreibanforderungen verwenden, sollten Sie außerdem erwägen, idempotente Abfragen für Erstellungs- und Aktualisierungsanforderungen mithilfe von zu implementieren. the section called “Effiziente Upserts ab 3.6.x”

Empfehlungen für die Verwendung von Gremlin-Leseanforderungen in Lambda

Wenn Sie eine oder mehrere Lesereplikate in Ihrem Cluster haben, ist es sinnvoll, die Leseanforderungen auf diese zu verteilen. Eine Möglichkeit besteht darin, den Reader-Endpunkt zu verwenden. Der Reader-Endpunkt gleicht Verbindungen zwischen den Replikaten aus, auch wenn sich die Cluster-Topologie ändert, wenn Sie Replikate hinzufügen oder entfernen oder ein Replikat zur neuen primären Instance heraufstufen.

Die Verwendung des Reader-Endpunkts kann jedoch unter bestimmten Umständen zu einer ungleichmäßigen Nutzung der Cluster-Ressourcen führen. Der Leser-Endpunkt funktioniert durch periodisches Ändern des Hosts, auf den der DNS-Eintrag verweist. Wenn ein Client viele Verbindungen öffnet, bevor sich der DNS-Eintrag ändert, werden alle Verbindungsanfragen an eine einzelne Neptune-Instance gesendet. Dies kann bei einem Lambda-Szenario mit hohem Durchsatz der Fall sein, in dem eine große Anzahl gleichzeitiger Anfragen an Ihre Lambda-Funktion dazu führt, dass mehrere Ausführungskontexte mit jeweils eigener Verbindung erstellt werden. Wenn diese Verbindungen fast gleichzeitig erstellt werden, verweisen sie wahrscheinlich alle auf dasselbe Replikat im Cluster und tun dies so lange, bis die Ausführungskontexte wiederverwendet werden.

Eine Möglichkeit, Anfragen auf Instances zu verteilen, besteht darin, Ihre Lambda-Funktion so zu konfigurieren, dass sie eine Verbindung mit einem Instance-Endpunkt herstellt, der nach dem Zufallsprinzip aus einer Liste von Replikat-Instance-Endpunkten ausgewählt wird, und nicht mit dem Reader-Endpunkt. Der Nachteil dieser Vorgehensweise besteht darin, dass der Lambda-Code

Änderungen in der Cluster-Topologie verarbeiten muss, indem er den Cluster überwacht und die Endpunktliste aktualisiert, wenn sich die Mitgliedschaft des Clusters ändert.

Wenn Sie eine Java-Lambda-Funktion schreiben, die Leseanforderungen zwischen Instances in Ihrem Cluster ausgleichen muss, können Sie den Gremlin-Client für Amazon Neptune verwenden, einen Java-Gremlin-Client, der Ihre Cluster-Topologie kennt und Verbindungen und Anfragen in fairer Weise auf eine Reihe von Instances in einem Neptune-Cluster verteilt. Dieser Blog-Beitrag enthält ein Beispiel für eine Java-Lambda-Funktion, die den Gremlin-Client für Amazon Neptune verwendet.

Faktoren, die den Kaltstart der Neptune-Gremlin-Lambda-Funktionen verlangsamen können

Das erste Mal, dass eine AWS Lambda Funktion aufgerufen wird, wird als Kaltstart bezeichnet. Es gibt mehrere Faktoren, die die Latenz eines Kaltstarts erhöhen können:

- Stellen Sie sicher, dass Sie Ihrer Lambda-Funktion ausreichend Speicherplatz zuweisen. — Die Kompilierung während eines Kaltstarts kann für eine Lambda-Funktion erheblich langsamer sein als bei EC2, da die CPU-Zyklen linear proportional zum Speicher zugewiesen werden, den Sie der Funktion AWS Lambda zuweisen. Bei 1.769 MB Arbeitsspeicher erhält eine Funktion das Äquivalent einer vollen vCPU (eine vCPU-Sekunde an Credits pro Sekunde). Die Auswirkung einer unzureichenden Zuweisung von Arbeitsspeicher, um adäquate CPU-Zyklen zu empfangen, ist bei großen Lambda-Funktionen, die in Java geschrieben wurden, besonders ausgeprägt.
- Beachten Sie, dass die Aktivierung der IAM-Datenbankauthentifizierung einen Kaltstart verlangsamen kann – die AWS Identity and Access Management (IAM)-Datenbankauthentifizierung kann ebenfalls Kaltstarts verlangsamen, insbesondere wenn die Lambda-Funktion einen neuen Signaturschlüssel generieren muss. Diese Latenz wirkt sich nur auf den Kaltstart und nicht auf nachfolgende Anfragen aus, denn sobald die IAM-DB-Authentifizierung die Verbindungsdaten eingerichtet hat, überprüft Neptune nur noch periodisch, ob diese noch gültig sind.

Verwenden CloudFormation um eine Lambda-Funktion zur Verwendung in Neptune zu erstellen

Sie können eine CloudFormation Vorlage verwenden, um eine AWS Lambda Funktion zu erstellen, die auf Neptune zugreifen kann.

1. Um den Lambda-Funktionsstapel auf der CloudFormation Konsole zu starten, wählen Sie eine der Schaltflächen Stack starten in der folgenden Tabelle.

Region	Anzeigen	In Designer anzeigen	Starten
USA Ost (Nord-Virginia)	Anzeigen	In Designer anzeigen	
USA Ost (Ohio)	Anzeigen	In Designer anzeigen	
USA West (Nordkalifornien)	Anzeigen	In Designer anzeigen	
USA West (Oregon)	Anzeigen	In Designer anzeigen	
Kanada (Zentral)	Anzeigen	In Designer anzeigen	
Südamerika (São Paulo)	Anzeigen	In Designer anzeigen	
Europa (Stockholm)	Anzeigen	In Designer anzeigen	
Europa (Irland)	Anzeigen	In Designer anzeigen	
Europa (London)	Anzeigen	In Designer anzeigen	
Europa (Paris)	Anzeigen	In Designer anzeigen	
Europa (Frankfurt)	Anzeigen	In Designer anzeigen	
Middle East (Bahrain)	Anzeigen	In Designer anzeigen	
Naher Osten (VAE)	Anzeigen	In Designer anzeigen	

Region	Anzeigen	In Designer anzeigen	Starten
Israel (Tel Aviv)	Anzeigen	In Designer anzeigen	
Afrika (Kapstadt)	Anzeigen	In Designer anzeigen	
Asien-Pazifik (Hongkong)	Anzeigen	In Designer anzeigen	
Asien-Pazifik (Tokio)	Anzeigen	In Designer anzeigen	
Asien-Pazifik (Seoul)	Anzeigen	In Designer anzeigen	
Asien-Pazifik (Singapur)	Anzeigen	In Designer anzeigen	
Asien-Pazifik (Sydney)	Anzeigen	In Designer anzeigen	
Asien-Pazifik (Mumbai)	Anzeigen	In Designer anzeigen	
China (Peking)	Anzeigen	In Designer anzeigen	
China (Ningxia)	Anzeigen	In Designer anzeigen	
AWS GovCloud (US-West)	Anzeigen	In Designer anzeigen	
AWS GovCloud (US-East)	Anzeigen	In Designer anzeigen	

2. Wählen Sie auf der Seite Vorlage auswählen die Option Weiter aus.
3. Legen Sie auf der Seite Specify DB Details (DB-Details angeben) die folgenden Optionen fest:

a. Wählen Sie die Lambda-Laufzeit aus, je nachdem, welche Sprache Sie in Ihrer Lambda-Funktion verwenden möchten. Diese CloudFormation Vorlagen unterstützen derzeit die folgenden Sprachen:

   - Python 3.9 (entspricht `python39` in der Amazon-S3-URL)
   - NodeJS 18 (entspricht `nodejs18x` in der Amazon-S3-URL)
   - Ruby 2.5 (entspricht `ruby25` in der Amazon-S3-URL)

b. Geben Sie den entsprechenden Neptune-Cluster-Endpunkt und die Portnummer an.

c. Geben Sie die entsprechende Neptune-Sicherheitsgruppe an.

d. Geben Sie die entsprechenden Neptune-Subnetzparameter an.

4. Wählen Sie Weiter aus.
5. Wählen Sie auf der Seite Optionen Weiter aus.
6. Aktivieren Sie auf der Seite Überprüfen das erste Kontrollkästchen, um zu bestätigen, dass CloudFormation IAM-Ressourcen erstellt.

   Wählen Sie die Option Erstellen aus.

Wenn Sie eigene Änderungen an der Lambda-Laufzeit vornehmen möchten, können Sie eine generische Laufzeit aus einem Amazon-S3-Speicherort in Ihrer Region herunterladen:

```
https://s3.Amazon region.amazonaws.com/aws-neptune-customer-samples-Amazon region/
lambda/runtime-language/lambda_function.zip.
```

Beispiel:

```
https://s3.us-west-2.amazonaws.com/aws-neptune-customer-samples-us-west-2/lambda/
python36/lambda_function.zip
```

AWS Lambda Funktionsbeispiele für Amazon Neptune

Die folgenden in Java JavaScript und Python geschriebenen AWS Lambda Beispielfunktionen veranschaulichen, wie ein einzelner Scheitelpunkt mit einer zufällig generierten ID mithilfe des `mergeV()` Schritts verschoben wird (siehe). the section called “Effiziente Upserts ab 3.6.x”

Bei einem Großteil des Codes in den einzelnen Funktionen handelt es sich um Boilerplate-Code, der für die Verwaltung von Verbindungen und die Wiederholung von Verbindungen und Abfragen im Falle

eines Fehlers zuständig ist. Die eigentliche Anwendungslogik und die Gremlin-Abfrage werden in den Methoden `doQuery()` bzw. `query()` implementiert. Wenn Sie diese Beispiele als Grundlage für Ihre eigenen Lambda-Funktionen verwenden, können Sie sich auf die Änderung von `doQuery()` und `query()` konzentrieren.

Die Funktionen sind so konfiguriert, dass fehlgeschlagene Abfragen 5-mal wiederholt werden und zwischen den Wiederholungen jeweils 1 Sekunde gewartet wird. Erwägen Sie in einer Produktionsanwendung die Verwendung eines exponentiellen Backoffs mit Jitter anstelle eines festen Intervalls. Eine ausführliche Anleitung finden Sie unter. the section called “Ausnahmen und Wiederholungen”

Die Funktionen erfordern Werte in den folgenden Lambda-Umgebungsvariablen:

- **`NEPTUNE_ENDPOINT`** – Ihr Neptune-DB-Cluster-Endpunkt. Für Python sollte dies `neptuneEndpoint` sein.
- **`NEPTUNE_PORT`** – Der Neptune-Port. Für Python sollte dies `neptunePort` sein.
- **`USE_IAM`** — (`true`oder`false`) Wenn in Ihrer Datenbank die AWS Identity and Access Management (IAM-) Datenbankauthentifizierung aktiviert ist, setzen Sie die `USE_IAM` Umgebungsvariable auf`true`. Dies veranlasst die Lambda-Funktion, Verbindungsanforderungen an Neptune mit Sigv4 zu signieren. Stellen Sie für solche IAM-DB-Authentifizierungsanforderungen sicher, dass der Ausführungsrolle der Lambda-Funktion eine entsprechende IAM-Richtlinie angefügt ist, die es der Funktion ermöglicht, eine Verbindung zu Ihrem Neptune-DB-Cluster herzustellen (siehe Arten von IAM-Richtlinien).

Beispiel für eine Java-Lambda-Funktion für Amazon Neptune

Hier sind einige Dinge, die Sie bei AWS Lambda Java-Funktionen beachten sollten:

- Der Java-Treiber unterhält seinen eigenen Verbindungspool, den Sie nicht benötigen. Konfigurieren Sie Ihr `Cluster`-Objekt daher mit `minConnectionPoolSize(1)` und `maxConnectionPoolSize(1)`.
- Die Erstellung des `Cluster`-Objekts kann langsam sein, da es einen oder mehrere Serialisierer erstellt (standardmäßig Gyro, plus einen weiteren, wenn Sie es für zusätzliche Ausgabeformate wie z. B. `binary` konfiguriert haben). Deren Instanziierung kann eine Weile dauern.
- Der Verbindungspool wird mit der ersten Anforderung initialisiert. Zu diesem Zeitpunkt richtet der Treiber den `Netty`-Stack ein, weist Byte-Puffer zu und erstellt einen Signaturschlüssel, wenn Sie die IAM-DB-Authentifizierung verwenden. All dies kann die Kaltstart-Latenz erhöhen.

- Der Verbindungspool des Java-Treibers überwacht die Verfügbarkeit der Server-Hosts und versucht automatisch eine erneute Verbindung, wenn eine Verbindung fehlschlägt. Er startet eine Hintergrundaufgabe, um zu versuchen, die Verbindung wiederherzustellen. Verwenden Sie `reconnectInterval( )`, um das Intervall zwischen erneuten Verbindungsversuchen zu konfigurieren. Während der Treiber eine erneute Verbindung versucht, kann Ihre Lambda-Funktion die Abfrage einfach wiederholen.

  Wenn das Intervall zwischen den Wiederholungen kleiner als das Intervall zwischen den erneuten Verbindungsversuchen ist, schlagen Wiederholungsversuche bei einer fehlgeschlagenen Verbindung erneut fehl, da der Host als nicht verfügbar angesehen wird. Dies gilt nicht für Wiederholungen für eine `ConcurrentModificationException`.
- Verwenden Sie Java 8 anstelle von Java 11. `Netty`-Optimierungen sind in Java 11 standardmäßig nicht aktiviert.
- In diesem Beispiel wird Retry4J für Wiederholungen verwendet.
- Informationen zur Verwendung des `Sigv4`-Signaturtreibers in Ihrer Java-Lambda-Funktion finden Sie in den Abhängigkeitsanforderungen in Herstellen einer Verbindung zu Amazon Neptune Neptune-Datenbanken mithilfe von IAM mit Gremlin Java.

> **Warning**
>
> Der `CallExecutor` von Retry4j ist möglicherweise nicht threadsicher. Überlegen Sie sich, jeden Thread eine eigene `CallExecutor`-Instance verwenden zu lassen.

> **Note**
>
> Das folgende Beispiel wurde aktualisiert und beinhaltet nun die Verwendung von requestInterceptor (). Dies wurde in 3.6.6 hinzugefügt. TinkerPop Vor TinkerPop Version 3.6.6 verwendete das Codebeispiel HandshakeInterceptor (), das in dieser Version veraltet war.

```
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;
     }
   }
}
```

Wenn Sie die Wiederverbindungslogik in Ihre Funktion aufnehmen möchten, finden Sie weitere Informationen unter Beispiel für eine erneute Verbindung mit Java.

JavaScript Beispiel für eine Lambda-Funktion für Amazon Neptune

Anmerkungen zu diesem Beispiel

- Der JavaScript Treiber unterhält keinen Verbindungspool. Es öffnet immer eine einzige Verbindung.

- Die Beispielfunktion verwendet die Sigv4-Signaturdienstprogramme von gremlin-aws-sigv4 zum Signieren von Anfragen an eine Datenbank mit IAM-Authentifizierung.
- Sie verwendet die Funktion retry () aus dem Open-Source-Async-Utility-Modul, um Versuche zu verarbeiten. backoff-and-retry
- Gremlin-Terminalschritte geben a zurück JavaScript `promise` (siehe Dokumentation). TinkerPop Für `next()` ist dies ein Tupel `{value, done}`
- Verbindungsfehler werden innerhalb des Handlers ausgelöst und mit einer gewissen backoff-and-retry Logik gemäß den hier skizzierten Empfehlungen behandelt, mit einer Ausnahme. Es gibt eine Art von Verbindungsproblem, das der Treiber nicht als Ausnahme behandelt und das daher mit dieser backoff-and-retry Logik nicht behoben werden kann.

  Das Problem besteht darin, dass die Abfrage zwar abgeschlossen zu werden scheint, jedoch einen Nullwert zurückgibt, wenn eine Verbindung geschlossen wird, nachdem ein Treiber eine Anforderung gesendet hat, jedoch bevor der Treiber eine Antwort erhält. Was den Client der Lambda-Funktion betrifft, so scheint die Funktion erfolgreich abgeschlossen zu werden, allerdings mit einer leeren Antwort.

  Wie sich dieses Problem auswirkt, hängt davon ab, wie Ihre Anwendung mit einer leeren Antwort umgeht. Einige Anwendungen behandeln eine leere Antwort auf eine Leseanforderung möglicherweise als Fehler, während andere sie fälschlicherweise als leeres Ergebnis behandeln.

  Schreibanforderungen, bei denen dieses Verbindungsproblem auftritt, geben ebenfalls eine leere Antwort zurück. Signalisiert ein erfolgreicher Aufruf mit einer leeren Antwort eine erfolgreiche Ausführung oder einen Fehler? Wenn der Client, der eine Schreibfunktion aufruft, den erfolgreichen Aufruf der Funktion einfach so interpretiert, dass der Schreibvorgang in die Datenbank festgeschrieben wurde, anstatt den Hauptteil der Antwort zu überprüfen, kann es so aussehen, als ob das System Daten verliert.

  Dieses Problem ist darauf zurückzuführen, wie der Treiber Ereignisse behandelt, die vom zugrunde liegenden Socket ausgegeben werden. Wenn der zugrundeliegende Netzwerk-Socket mit einem `ECONNRESET` Fehler geschlossen wird, WebSocket wird der vom Treiber verwendete Socket geschlossen und es wird ein `'ws close'` Ereignis ausgelöst. Der Treiber enthält jedoch nichts, um dieses Ereignis so zu behandeln, dass eine Ausnahme ausgelöst werden könnte. Infolgedessen verschwindet die Abfrage einfach.

  Um dieses Problem zu umgehen, fügt die Lambda-Beispielfunktion hier einen `'ws close'`-Event-Handler hinzu, der beim Herstellen einer Remote-Verbindung eine Ausnahme für den Treiber

auslöst. Diese Ausnahme wird jedoch nicht entlang des Anfrage-Antwort-Pfads der Gremlin-Abfrage ausgelöst und kann daher nicht verwendet werden, um backoff-and-retry Logik innerhalb der Lambda-Funktion selbst auszulösen. Die vom `'ws close'`-Event-Handler ausgelöste Ausnahme fühlt vielmehr zu einer unbehandelten Ausnahme, die zur Folge hat, dass der Lambda-Aufruf fehlschlägt. So kann der Client, der die Funktion aufruft, den Fehler behandeln und den Lambda-Aufruf gegebenenfalls erneut versuchen.

Wir empfehlen Ihnen, backoff-and-retry Logik in der Lambda-Funktion selbst zu implementieren, um Ihre Clients vor zeitweiligen Verbindungsproblemen zu schützen. Um das oben genannte Problem zu umgehen, muss jedoch auch der Client eine Wiederholungslogik implementieren, um Fehler zu behandeln, die sich aus diesem speziellen Verbindungsproblem ergeben.

Javascript-Code

```
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);
};
```

Beispiel für eine Python-Lambda-Funktion für Amazon Neptune

Hier einige Dinge, die Sie in Bezug auf die folgende Python- AWS Lambda -Beispielfunktion beachten sollten:

- Sie verwendet das Backoff-Modul.
- Sie legt `pool_size=1` fest, damit kein unnötiger Verbindungspool erstellt wird.

```
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)
```

Hier sind Beispielergebnisse, die Zeiten hoher und geringer Auslastung im Wechsel zeigen:


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 für Machine Learning anhand von Diagrammen

Große verbundene Datensätze enthalten häufig wertvolle Informationen, die nur schwer mit Abfragen extrahiert werden können, die allein auf menschlicher Intuition basieren. Machine-Learning (ML)-Techniken können helfen, in Diagrammen mit Milliarden von Beziehungen verborgene Korrelationen zu finden. Diese Korrelationen können nützlich sein, um Produkte zu empfehlen, die Kreditwürdigkeit vorherzusagen, Betrugsfälle zu erkennen und mehr.

Das Neptune-ML-Feature ermöglicht das Erstellen und Trainieren nützlicher Machine-Learning-Modelle anhand großer Diagramme in wenigen Stunden statt in Wochen. Um dies zu erreichen, verwendet Neptune ML die Graph Neural Network (GNN) -Technologie, die von Amazon SageMaker AI und der Deep Graph Library (DGL) (Open Source) unterstützt wird. Graph Neural Networks sind ein neues Feld in der künstlichen Intelligenz (siehe Umfassende Übersicht über Graph Neural Networks). Ein praktisches Tutorial zur Verwendung GNNs mit DGL finden Sie unter Lernen neuronaler Graphennetzwerke mit der Deep Graph Library.

> **Note**
>
> Diagrammeckpunkte werden in Neptune-ML-Modellen als „Knoten“ bezeichnet. Beispielsweise verwendet die Eckpunktklassifizierung ein Machine-Learning-Modell für die Knotenklassifizierung. Die Eckpunktregression verwendet ein Knotenregressionsmodell.

Fähigkeiten von Neptune ML

Neptune unterstützt sowohl die transduktive Inferenz, die Vorhersagen zurückgibt, die während des Trainings anhand der Diagrammdaten vorab berechnet wurden, und die induktive Inferenz, die Datenverarbeitung und Modellauswertung in Echtzeit basierend auf aktuellen Daten ausführt und zurückgibt. Siehe Der Unterschied zwischen induktiver und transduktiver Inferenz.

Neptune ML kann Machine-Learning-Modelle zur Unterstützung von fünf verschiedenen Inferenzkategorien trainieren:

Arten der zurzeit von Neptune ML unterstützten Inferenzaufgaben

- Knotenklassifizierung – Vorhersage des kategorischen Features einer Eckpunkteigenschaft.

Beispielsweise kann Neptune ML für den Film Die Verurteilten dessen `genre`-Eigenschaft als `story` vorhersagen, basierend auf einem Satz mit den Kandidaten `[story, crime, action, fantasy, drama, family, ...]`.

Es gibt zwei Arten von Aufgaben zur Knotenklassifizierung:

- Einzelklassenklassifizierung: Bei dieser Art von Aufgabe besitzt jeder Knoten nur ein Ziel-Feature. Beispielsweise hat die Eigenschaft `Place_of_birth` von `Alan Turing` den Wert UK.
- Mehrklassenklassifizierung: Bei dieser Art von Aufgabe besitzt jeder Knoten mehr als ein Ziel-Feature. Beispielsweise hat die Eigenschaft `genre` des Films Der Pate die Werte `crime` und `story`.

- Knotenregression – Vorhersage einer numerischen Eigenschaft eines Eckpunkts.

  Beispielsweise kann Neptune ML für den Film Avengers: Endgame vorhersagen, dass dessen Eigenschaft `popularity` den Wert `5.0` hat.

- Kantenklassifizierung – Vorhersage des kategorischen Features einer Kanteneigenschaft.

  Es gibt zwei Arten von Aufgaben zur Kantenklassifizierung:

  - Einzelklassenklassifizierung: Bei dieser Art von Aufgabe besitzt jede Kante nur ein Ziel-Feature. Eine Bewertungskante zwischen einem Benutzer und einem Film könnte beispielsweise die Eigenschaft `liked` mit dem Wert „Ja“ oder „Nein“ haben.
  - Mehrklassenklassifizierung: Bei dieser Art von Aufgabe besitzt jede Kante mehr als ein Ziel-Feature. Beispielsweise kann eine Bewertungskante zwischen einem Benutzer und einem Film mehrere Werte für das Eigenschafts-Tag haben, z. B. „Lustig“, „Herzerwärmend“, „Entspannend“ usw.

- Kantenregression – Vorhersage einer numerischen Eigenschaft einer Kante.

  Beispielsweise könnte eine Bewertungskante zwischen einem Benutzer und einem Film die numerische Eigenschaft `score` haben, für die Neptune ML einen Wert anhand eines Benutzers und eines Films vorhersagen könnte.

- Linkvorhersage – Vorhersage der wahrscheinlichsten Zielknoten für einen bestimmten Quellknotens und eine bestimmte ausgehende Kante oder der wahrscheinlichsten Quellknoten für einen bestimmten Zielknoten und eine bestimmte eingehende Kante.

  Beispielsweise kann Neptune ML anhand eines Arzneimittel-Erkrankung-Wissensdiagramms anhand von `Aspirin` als Quellknoten und `treats` als ausgehende Kante die relevantesten Zielknoten als `heart disease`, `fever` usw. vorhersagen.

Neptune ML kann auch anhand des Wikimedia-Wissensdiagramms mit `President-of` als Kante oder Beziehung und `United-States` als Zielknoten die relevantesten Köpfe als `George Washington`, `Abraham Lincoln`, `Franklin D. Roosevelt` usw. vorhersagen.

> **Note**
>
> Knotenklassifizierung und Kantenklassifizierung unterstützen nur Zeichenfolgenwerte. Das bedeutet, dass numerische Eigenschaftswerte wie `0` oder `1` nicht unterstützt werden, auch wenn die Zeichenfolgenäquivalente `"0"` und `"1"` unterstützt werden. Ähnlich funktionieren die booleschen Eigenschaftswerte `true` und `false` nicht, während `"true"` und `"false"` funktionieren.

Mit Neptune ML können Sie Machine-Learning-Modelle verwenden, die zu zwei allgemeinen Kategorien gehören:

Arten von Machine-Learning-Modellen, die Neptune ML zurzeit unterstützt

- Graph Neural Network (GNN) -Modelle — Dazu gehören Relational Graph Convolutional Networks (R-). GCNs GNN-Modelle funktionieren für alle drei oben genannten Aufgabentypen.
- Knowledge-Graph-Embedding (KGE)-Modelle – Dies umfasst `TransE`-, `DistMult`- und `RotatE`-Modelle. Sie funktionieren nur für Linkvorhersagen.

Benutzerdefinierte Modelle – Mit Neptune ML können Sie auch für alle oben genannten Aufgabentypen eine eigene benutzerdefinierte Modellimplementierung bereitstellen. Sie können Ihre Python-basierte benutzerdefinierte Modellimplementierung mit dem Neptune-ML-Toolkit entwickeln und testen, bevor Sie die Neptune-ML-Trainings-API mit Ihrem Modell verwenden. Einzelheiten dazu, wie Sie Ihre Implementierung strukturieren und organisieren können, damit sie mit der Neptune-ML-Trainingsinfrastruktur kompatibel ist, finden Sie unter Benutzerdefinierte Modelle in Neptune ML.

Einrichten von Neptune ML

Der einfachste Weg, mit Neptune ML zu beginnen, ist die Verwendung der CloudFormation Schnellstart-Vorlage. Diese Vorlage installiert alle erforderlichen Komponenten, einschließlich eines neuen Neptune-DB-Clusters, aller erforderlichen IAM-Rollen und eines neuen Neptune-Graph-Notebooks, um die Arbeit mit Neptune ML zu vereinfachen.

Sie können Neptune ML auch manuell installieren, wie in Neptune ML einrichten, ohne die Schnellstartvorlage zu verwenden CloudFormation beschrieben.

Verwendung des Neptune ML AWS CloudFormation Vorlage für den schnellen Einstieg in einem neuen DB-Cluster

Der einfachste Weg, mit Neptune ML zu beginnen, ist die Verwendung der CloudFormation Schnellstart-Vorlage. Diese Vorlage installiert alle erforderlichen Komponenten, einschließlich eines neuen Neptune-DB-Clusters, aller erforderlichen IAM-Rollen und eines neuen Neptune-Graph-Notebooks, um die Arbeit mit Neptune ML zu vereinfachen.

So erstellen Sie den Schnellstart-Stack für Neptune ML

1. Um den CloudFormation Stack auf der CloudFormation Konsole zu starten, wählen Sie eine der Schaltflächen Stack starten in der folgenden Tabelle:

Region	Anzeigen	Starten
USA Ost (Nord-Virginia)	Anzeigen	Launch Stack
USA Ost (Ohio)	Anzeigen	Launch Stack
USA West (Nordkalifornien)	Anzeigen	Launch Stack
USA West (Oregon)	Anzeigen	Launch Stack
Kanada (Zentral)	Anzeigen	Launch Stack
Südamerika (São Paulo)	Anzeigen	Launch Stack
Europa (Stockholm)	Anzeigen	Launch Stack
Europa (Irland)	Anzeigen	Launch Stack
Europa (London)	Anzeigen	Launch Stack

Region	Anzeigen	Starten
Europa (Paris)	Anzeigen	
Europa (Frankfurt)	Anzeigen	
Middle East (Bahrain)	Anzeigen	
Naher Osten (VAE)	Anzeigen	
Israel (Tel Aviv)	Anzeigen	
Afrika (Kapstadt)	Anzeigen	
Asien-Pazifik (Hongkong)	Anzeigen	
Asien-Pazifik (Tokio)	Anzeigen	
Asien-Pazifik (Seoul)	Anzeigen	
Asien-Pazifik (Singapur)	Anzeigen	
Asien-Pazifik (Sydney)	Anzeigen	
Asien-Pazifik (Mumbai)	Anzeigen	
China (Peking)	Anzeigen	
China (Ningxia)	Anzeigen	
AWS GovCloud (US-West)	Anzeigen	

2. Wählen Sie auf der Seite Vorlage auswählen die Option Weiter aus.
3. Wählen Sie auf der Seite Specify Details (Details angeben) die Option Next (Weiter) aus.
4. Wählen Sie auf der Seite Optionen Weiter aus.
5. Auf der Seite Überprüfen gibt es zwei Kontrollkästchen, die Sie aktivieren müssen:

   - Die erste bestätigt, dass dadurch AWS CloudFormation möglicherweise IAM-Ressourcen mit benutzerdefinierten Namen erstellt werden.
   - Die zweite räumt ein, dass AWS CloudFormation möglicherweise die `CAPABILITY_AUTO_EXPAND` Fähigkeit für den neuen Stack erforderlich ist. `CAPABILITY_AUTO_EXPAND`Derlaubt ausdrücklich CloudFormation , Makros bei der Erstellung des Stacks automatisch und ohne vorherige Überprüfung zu erweitern.

     Kunden erstellen häufig einen Änderungssatz aus einer verarbeiteten Vorlage, sodass die von Makros vorgenommenen Änderungen überprüft werden können, bevor der Stack tatsächlich erstellt wird. Weitere Informationen finden Sie in der CloudFormation CreateStack-API.

   Wählen Sie die Option Erstellen aus.

Mit der Schnellstartvorlage wird Folgendes erstellt und eingerichtet:

- Ein Neptune-DB-Cluster
- Die erforderlichen IAM-Rollen (Diese werden auch angefügt.)
- Die erforderliche Amazon-EC2-Sicherheitsgruppe
- Die erforderlichen SageMaker KI-VPC-Endpunkte.
- Eine DB-Cluster-Parametergruppe für Neptune ML
- Die erforderlichen Parameter in dieser Parametergruppe
- Ein SageMaker KI-Notizbuch mit vorausgefüllten Notizbuchbeispielen für Neptune ML. Beachten Sie, dass nicht alle Instance-Größen in jeder Region verfügbar sind. Sie müssen also sicherstellen, dass die ausgewählte Notebook-Instance-Größe von Ihrer Region unterstützt wird.
- Der Service. Neptune-Export

Wenn der Schnellstart-Stack fertig ist, rufen Sie das SageMaker KI-Notizbuch auf, das mit der Vorlage erstellt wurde, und sehen Sie sich die vorab ausgefüllten Beispiele an. Sie helfen Ihnen

dabei, Beispiel-Datensätze herunterzuladen, die Sie zum Experimentieren mit Neptune-ML-Funktionen verwenden können.

Außerdem können Sie damit viel Zeit sparen, wenn Sie Neptune ML verwenden. Sehen Sie sich zum Beispiel die Zeilen-Magic %neptune_ml und die Zellen-Magic %%neptune_ml an, die diese Notebooks unterstützen.

Sie können auch den folgenden AWS CLI Befehl verwenden, um die Schnellstart-Vorlage auszuführen: 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)
```

Neptune ML einrichten, ohne die Schnellstartvorlage zu verwenden CloudFormation

Dieses Handbuch enthält step-by-step Anweisungen zur Einrichtung von Amazon Neptune ML ohne Verwendung der AWS CloudFormation Schnellstartvorlage. Es wird davon ausgegangen, dass Sie bereits über einen funktionierenden Neptune-DB-Cluster verfügen, und behandelt die notwendige Einrichtung, einschließlich der Installation des Neptune-Export-Dienstes, der Erstellung benutzerdefinierter IAM-Rollen und der Konfiguration Ihres DB-Clusters zur Aktivierung von Neptune ML. In der Anleitung wird auch erklärt, wie Sie zwei SageMaker KI-Endpunkte in Ihrer Neptune-VPC erstellen, um der Neptune-Engine Zugriff auf das erforderliche KI-Management zu gewähren. SageMaker APIs Wenn Sie diese Anweisungen befolgen, können Sie Neptune ML auf Ihrer vorhandenen Neptune-Infrastruktur einrichten, ohne sich auf die Vorlage verlassen zu müssen. CloudFormation

Beginnen mit einem funktionierenden Neptune-DB-Cluster

Wenn Sie die CloudFormation Schnellstartvorlage nicht für die Einrichtung von Neptune ML verwenden, benötigen Sie einen vorhandenen Neptune-DB-Cluster, mit dem Sie arbeiten können. Sie können entweder einen bereits vorhandenen Neptune-DB-Cluster verwenden, einen Neptune-DB-Cluster, den Sie bereits verwenden, klonen oder einen neuen Neptune-DB-Cluste erstellen (siehe Neptun-Cluster erstellen).

Installieren des Service Neptune-Export

Installieren Sie den Service Neptune-Export gemäß den Anweisungen unter Nutzung des Neptune-Export Dienstes zum Exportieren von Neptun-Daten, falls Sie dies noch nicht getan haben.

Fügen Sie der Sicherheitsgruppe `NeptuneExportSecurityGroup`, die bei der Installation erstellt wird, eine Regel für eingehenden Datenverkehr mit den folgenden Einstellungen hinzu:

- Typ: `Custom TCP`
- Protocol (Protokoll): `TCP`
- Port-Bereich: `80 - 443`
- Quelle: *`(Neptune DB cluster security group ID)`*

Erstellen Sie eine benutzerdefinierte S3-IAM-Rolle NeptuneLoadFrom

Erstellen Sie die benutzerdefinierte IAM-Rolle`NeptuneLoadFromS3` gemäß dem Anweisungen unter Erstellen einer IAM-Rolle für den Zugriff auf Amazon S3, falls Sie dies noch nicht getan haben.

Erstellen Sie eine benutzerdefinierte Rolle NeptuneSageMaker IAMRole

Verwenden Sie die IAM-Konsole, um eine benutzerdefinierte `NeptuneSageMakerIAMRole` mit der folgenden Richtlinie zu erstellen:

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"
    }
  ]
}
```

Bearbeiten Sie während der Erstellung dieser Rolle die Vertrauensstellung, sodass sie wie folgt aussieht:

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "ec2.amazonaws.com",
          "rds.amazonaws.com",
          "sagemaker.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Kopieren Sie abschließend den ARN, der dieser neuen Rolle `NeptuneSageMakerIAMRole` zugewiesen ist.

> **Important**
>
> - Stellen Sie sicher, dass die Amazon-S3-Berechtigungen in `NeptuneSageMakerIAMRole` mit den oben aufgeführten übereinstimmen.
> - Der universelle ARN `arn:aws:s3:::*` wird in der obigen Richtlinie für die Amazon-S3-Ressource verwendet. Wenn der universelle ARN aus irgendeinem Grund nicht verwendet werden kann, müssen `arn:aws:s3:::graphlytics*` und der ARN für jede andere Amazon-S3-Kundenressource, die von NeptuneML-Befehlen verwendet wird, dem Ressourcenbereich hinzugefügt werden.

Konfigurieren Ihres DB-Clusters zur Aktivierung von Neptune ML

So richten Sie Ihren DB-Cluster für Neptune ML ein

1. Navigieren Sie in der Neptune-Konsole zu Parametergruppen und dann zu der DB-Cluster-Parametergruppe, die dem DB-Cluster zugeordnet ist, den Sie verwenden werden. Setzen Sie den Parameter `neptune_ml_iam_role` auf den ARN, der der gerade erstellten Rolle zugewiesen ist.
2. Navigieren Sie zu Datenbanken und wählen Sie dann den DB-Cluster aus, den Sie für Neptune ML verwenden werden. Wählen Sie Aktionen und dann IAM-Rollen verwalten aus.
3. Wählen Sie auf der Seite IAM-Rollen verwalten die Option Rolle hinzufügen aus und fügen Sie `NeptuneSageMakerIAMRole` hinzu. Fügen Sie dann die Rolle `NeptuneLoadFromS3` hinzu.
4. Starten Sie die Writer-Instance Ihres DB-Clusters neu.

Erstellen Sie zwei SageMaker KI-Endpunkte in Ihrer Neptune VPC

Um der Neptune-Engine Zugriff auf das erforderliche SageMaker KI-Management zu gewähren APIs, müssen Sie schließlich zwei SageMaker KI-Endpunkte in Ihrer Neptune-VPC erstellen, wie unter erklärt. Erstellen Sie zwei Endpunkte für SageMaker KI in Ihrer Neptune VPC

Manuelles Konfigurieren eines Neptune-Notebooks für Neptune ML

Neptune SageMaker AI-Notebooks sind mit einer Vielzahl von Musternotizbüchern für Neptune ML vorinstalliert. Sie können eine Vorschau dieser Beispiele im Open-Source-Graph-Notebook-Repository anzeigen. GitHub

Sie können eins der vorhandenen Neptune-Notebooks verwenden oder, wenn Sie möchten, ein eigenes erstellen, indem Sie die Anweisungen unter Verwenden der Neptune-Workbench zum Hosten von Neptune-Notebooks befolgen.

Sie können auch ein standardmäßiges Neptune-Notebook zum Verwenden mit Neptune ML konfigurieren, indem Sie die folgenden Schritte ausführen:

Ändern eines Notebooks für Neptune ML

1. Öffnen Sie die Amazon SageMaker AI-Konsole unter https://console.aws.amazon.com/sagemaker/.
2. Wählen Sie im Navigationsbereich auf der linken Seite Notebook und dann Notebook-Instances aus. Suchen Sie den Namen des Neptune-Notebooks, das Sie für Neptune ML verwenden möchten, und wählen Sie es aus, um die zugehörige Detailseite aufzurufen.
3. Wenn die Notebook-Instance ausgeführt wird, klicken Sie oben rechts auf der Notebook-Detailseite auf die Schaltfläche Stopp.
4. Wählen Sie in den Notebook-Instance-Einstellungen unter Lebenszykluskonfiguration den Link aus, um die Seite zum Lebenszyklus des Notebooks zu öffnen.
5. Wählen Sie oben rechts Bearbeiten und dann Weiter aus.
6. Ändern Sie auf der Registerkarte Notebook starten das Skript, um zusätzliche Exportbefehle aufzunehmen und die Felder für Ihre IAM-Rolle in Neptune ML und den Export-Service-URI auszufüllen – je nach Ihrer Shell in etwa wie folgt:

   ```
   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. Wählen Sie Aktualisieren.
8. Kehren Sie zur Notebook-Instance-Seite zurück. Unter Berechtigungen und Verschlüsselung gibt es ein Feld für den IAM-Rollen-ARN. Wählen Sie den Link in diesem Feld aus, um zu der IAM-Rolle zu gelangen, mit der diese Notebook-Instance ausgeführt wird.
9. Erstellen Sie eine neue Inline-Richtlinie wie diese:

   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. Speichern Sie diese neue Richtlinie und fügen Sie sie an die IAM-Rolle aus Schritt 8 an.
11. Wählen Sie oben rechts auf der Detailseite der SageMaker KI-Notebook-Instance Start aus, um die Notebook-Instance zu starten.

Verwenden von AWS CLI , um Neptune ML auf einem DB-Cluster einzurichten

Neben der CloudFormation Schnellstart-Vorlage und dem AWS-Managementkonsole können Sie Neptune ML auch mit dem einrichten. AWS CLI

Erstellen einer DB-Cluster-Parametergruppe für den neuen Neptune-ML-Cluster

Die folgenden AWS CLI Befehle erstellen eine neue DB-Cluster-Parametergruppe und richten sie so ein, dass sie mit Neptune ML funktioniert:

So erstellen und konfigurieren Sie eine DB-Cluster-Parametergruppe für Neptune ML

1. Erstellen Sie eine neue DB-Cluster-Parametergruppe:

```
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. Erstellen Sie einen `neptune_ml_iam_role` DB-Cluster-Parametersatz, der auf den ARN des `SageMakerExcecutionIAMRole` gesetzt ist, damit Ihr DB-Cluster ihn beim Aufrufen von SageMaker KI zum Erstellen von Jobs und zum Abrufen von Vorhersagen aus gehosteten ML-Modellen verwenden kann:

```
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)
```

   Wenn Sie diesen Parameter festlegen, kann Neptune auf SageMaker KI zugreifen, ohne dass Sie die Rolle bei jedem Anruf weitergeben müssen.

   Informationen zum Erstellen von `SageMakerExcecutionIAMRole` finden Sie unter Erstellen Sie eine benutzerdefinierte Rolle NeptuneSageMaker IAMRole.

3. Verwenden Sie abschließend `describe-db-cluster-parameters`, um zu überprüfen, ob alle Parameter in der neuen DB-Cluster-Parametergruppe wie gewünscht festgelegt sind:

```
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)
```

Anfügen der neuen DB-Cluster-Parametergruppe an den DB-Cluster, der mit Neptune ML verwendet werden soll

Jetzt können Sie die neue DB-Cluster-Parametergruppe, die Sie gerade erstellt haben, mit dem folgenden Befehl an einen vorhandenen DB-Cluster anfügen:

```
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)
```

Anschließend können Sie den DB-Cluster neu starten, damit alle Parameter wirksam werden:

```
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)
```

Wenn Sie einen neuen DB-Cluster zum Verwenden mit Neptune ML erstellen, können Sie alternativ den folgenden Befehl verwenden, um den Cluster mit der angefügten neuen Parametergruppe zu erstellen und dann eine neue primäre (Writer-)Instance zu erstellen:

```
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)
```

Hängen Sie das **NeptuneSageMakerIAMRole** an Ihren DB-Cluster an, damit es auf SageMaker KI- und Amazon S3 S3-Ressourcen zugreifen kann

Folgen Sie abschließend den Anweisungen unter, Erstellen Sie eine benutzerdefinierte Rolle NeptuneSageMaker IAMRole um eine IAM-Rolle zu erstellen, die es Ihrem DB-Cluster ermöglicht, mit SageMaker KI und Amazon S3 zu kommunizieren. Verwenden Sie dann den folgenden Befehl, um die von Ihnen erstellte Rolle `NeptuneSageMakerIAMRole` an Ihren DB-Cluster anzufügen:

```
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)
```

Erstellen Sie zwei Endpunkte für SageMaker KI in Ihrer Neptune VPC

Neptune ML benötigt zwei SageMaker KI-Endpunkte in der VPC Ihres Neptune-DB-Clusters:

- `com.amazonaws.(AWS region, like us-east-1).sagemaker.runtime`
- `com.amazonaws.(AWS region, like us-east-1).sagemaker.api`

Wenn Sie die CloudFormation Schnellstart-Vorlage, die diese automatisch für Sie erstellt, nicht verwendet haben, können Sie sie mit den folgenden Befehlen erstellen: AWS CLI

Mit diesem wird der Endpunkt `sagemaker.runtime` erstellt:

```
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
```

Und mit diesem wird der Endpunkt `sagemaker.api` erstellt:

```
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
```

Sie können diese Endpunkte auch mit der VPC-Konsole erstellen. Weitere Informationen finden Sie unter Sichere Prognoseaufrufe in Amazon SageMaker mit AWS PrivateLink und Sichern aller SageMaker Amazon-API-Aufrufe mit AWS PrivateLink.

Erstellen Sie einen SageMaker AI-Inferenz-Endpunktparameter in Ihrer DB-Cluster-Parametergruppe

Um zu vermeiden, dass Sie den SageMaker AI-Inferenzendpunkt des Modells, das Sie verwenden, bei jeder Abfrage angeben müssen, erstellen Sie einen DB-Cluster-Parameter mit dem Namen `neptune_ml_endpoint` in der DB-Cluster-Parametergruppe für Neptune ML. Legen Sie den Parameter auf die `id` des betreffenden Instance-Endpunkts fest.

Dazu können Sie den folgenden AWS CLI Befehl verwenden:

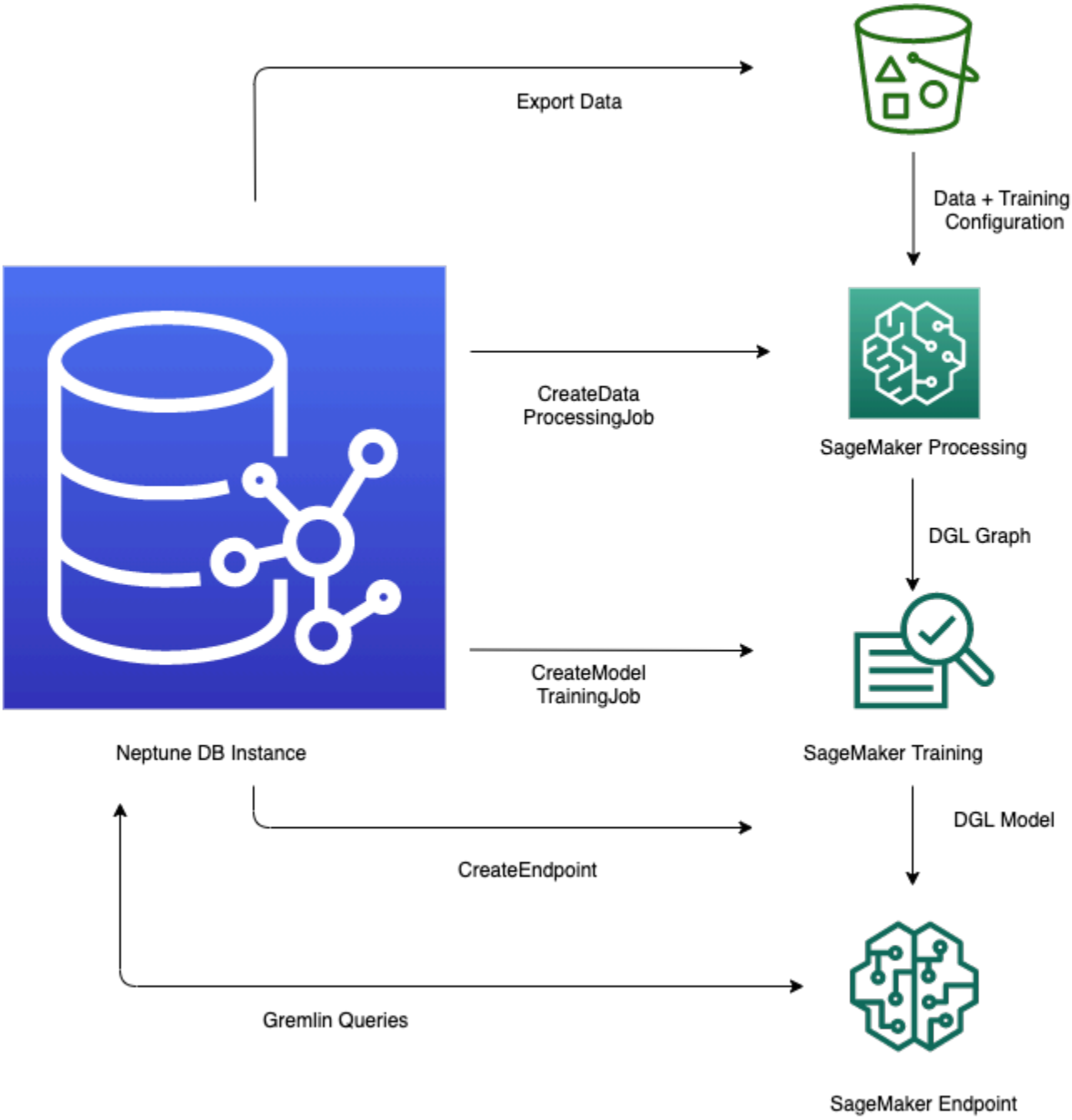
```
aws neptune modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name neptune-ml-demo \
  --parameters "ParameterName=neptune_ml_endpoint, \
                ParameterValue=(the name of the SageMaker AI inference endpoint you
 want to query), \
                Description=NeptuneMLEndpoint, \
                ApplyMethod=pending-reboot" \
  --region (AWS region, such as us-east-1)
```

Überblick über die Verwendung des Features Neptune ML

Die Neptune ML-Funktion in Amazon Neptune bietet einen optimierten Arbeitsablauf für die Nutzung von Modellen für maschinelles Lernen in einer Graphdatenbank. Der Prozess umfasst mehrere wichtige Schritte: das Exportieren von Daten aus Neptune in das CSV-Format, die Vorverarbeitung der Daten, um sie für das Modelltraining vorzubereiten, das maschinelle Lernmodell mit Amazon SageMaker AI zu trainieren, einen Inferenzendpunkt für Vorhersagen zu erstellen und dann das Modell direkt aus Gremlin-Abfragen abzufragen. Die Neptune Workbench bietet praktische Befehle für Linien- und Zellenmagie, mit denen Sie diese Schritte verwalten und automatisieren können. Durch die direkte Integration von Funktionen für maschinelles Lernen in die Graphdatenbank ermöglicht Neptune ML Benutzern, anhand der umfangreichen relationalen Daten, die im Neptun-Diagramm gespeichert sind, wertvolle Erkenntnisse abzuleiten und Vorhersagen zu treffen.

Starten des Workflows für die Verwendung von Neptune ML

Die Verwendung des Features Neptune ML in Amazon Neptune umfasst in der Regel zunächst die folgenden fünf Schritte:



1. Datenexport und Konfiguration – Beim Datenexport werden mithilfe des Service Neptune-Export oder des `neptune-export`-Befehlszeilen-Tools Daten aus Neptune im CSV-Format in Amazon Simple Storage Service (Amazon S3) exportiert. Gleichzeitig wird automatisch eine Konfigurationsdatei mit dem Namen `training-data-configuration.json` generiert, in der angegeben ist, wie die exportierten Daten in ein trainierbares Diagramm geladen werden können.
2. Datenvorverarbeitung – In diesem Schritt wird der exportierte Datensatz mithilfe von Standardverfahren vorverarbeitet, um ihn für das Modelltraining vorzubereiten. Für numerische

Daten kann eine Merkmalsnormalisierung durchgeführt und Textmerkmale können mit `word2vec` codiert werden. Am Ende dieses Schritts wird aus dem exportierten Datensatz ein DGL-Diagramm (Deep Graph Library) generiert, das während des Modelltrainings verwendet wird.

Dieser Schritt wird mithilfe eines SageMaker KI-Verarbeitungsauftrags in Ihrem Konto implementiert, und die resultierenden Daten werden an einem von Ihnen angegebenen Amazon S3 S3-Standort gespeichert.

3. Modelltraining – Beim Modelltraining wird das Machine-Learning-Modell trainiert, das für Prognosen verwendet wird.

   Das Modelltraining erfolgt in zwei Phasen:

   - In der ersten Phase wird mithilfe eines SageMaker KI-Verarbeitungsauftrags ein Konfigurationssatz für die Modelltrainingsstrategie generiert, der festlegt, welcher Modelltyp und welche Modell-Hyperparameterbereiche für das Modelltraining verwendet werden.
   - In der zweiten Phase werden dann mithilfe eines SageMaker KI-Modeltuning-Jobs verschiedene Hyperparameter-Konfigurationen ausprobiert und der Trainingsjob ausgewählt, der das Modell mit der besten Leistung hervorgebracht hat. Der Optimierungsauftrag führt eine vordefinierte Anzahl von Modelloptimierungsaufträgen mit den verarbeiteten Daten durch. Am Ende dieser Phase werden die trainierten Modellparameter des besten Trainingsauftrags verwendet, um Modellartefakte für Inferenzen zu generieren.

4. Erstellen Sie einen Inferenzendpunkt in Amazon SageMaker AI — Der Inferenzendpunkt ist eine SageMaker KI-Endpunkt-Instance, die mit den Modellartefakten gestartet wird, die durch den besten Trainingsjob erzeugt wurden. Jedes Modell ist an einen einzelnen Endpunkt gebunden. Der Endpunkt ist in der Lage, eingehende Anfragen von der Graphdatenbank anzunehmen und die Modellvorhersagen für die Eingaben in den Anfragen zurückzugeben. Nach der Erstellung des Endpunkts bleibt dieser aktiv, bis Sie ihn löschen.
5. Abfragen des Machine-Learning-Modells mithilfe von Gremlin – Sie können Erweiterungen der Gremlin-Abfragesprache verwenden, um Prognosen vom Inferenzendpunkt abzufragen.

> **Note**
>
> Die Neptune-Workbench enthält ein Line-Magic und ein Zell-Magic, mit denen Sie bei der Verwaltung dieser Schritte viel Zeit sparen können, nämlich:
>
> - %neptune_ml

- %%neptune_ml

Erstellen von Prognosen auf der Grundlage sich entwickelnder Graphdaten

Bei einem sich ständig ändernden Graphen möchten Sie möglicherweise regelmäßig neue Stapelprognosen auf der Grundlage aktueller Daten erstellen. Das Abfragen vorab berechneter Prognosen (transduktive Inferenz) kann erheblich schneller sein als die spontane Generierung neuer Prognosen auf der Grundlage der allerneuesten Daten (induktive Inferenz). Beide Ansätze haben ihre Berechtigung, je nachdem, wie schnell sich Ihre Daten ändern und welche Leistungsanforderungen Sie haben.

Der Unterschied zwischen induktiver und transduktiver Inferenz

Bei der Durchführung transduktiver Inferenz sucht Neptune nach Prognosen, die zum Zeitpunkt des Trainings vorberechnet wurden, und gibt sie zurück.

Bei der Durchführung induktiver Inferenz konstruiert Neptune den entsprechenden Untergraphen und ruft seine Eigenschaften ab. Das DGL-GNN-Modell wendet dann Datenverarbeitung und Modellauswertung in Echtzeit an.

Durch induktive Inferenz können daher Prognosen zu Knoten und Edges generiert werden, die zum Zeitpunkt des Trainings nicht vorhanden waren und die den aktuellen Zustand des Graphen widerspiegeln. Damit geht jedoch eine höhere Latenz einher.

Wenn Ihr Graph dynamisch ist, sollten Sie die induktive Inferenz verwenden, um sicherzugehen, dass die neuesten Daten berücksichtigt werden. Wenn Ihr Graph jedoch statisch ist, ist die transduktive Inferenz schneller und effizienter.

Die induktive Inferenz ist standardmäßig deaktiviert. Sie können sie für eine Abfrage aktivieren, indem Sie das Gremlin-Prädikat Neptune#ml.inductiveInference in der Abfrage wie folgt verwenden:

```
.with( "Neptune#ml.inductiveInference")
```

Inkrementelle transduktive Workflows

Während Sie Modellartefakte einfach aktualisieren, indem Sie die Schritte eins bis drei (vom Datenexport und der Konfiguration bis zur Modelltransformation) erneut ausführen, unterstützt

Neptune ML einfachere Methoden, um Ihre ML-Stapelprognosen mithilfe neuer Daten zu aktualisieren. Eine Möglichkeit besteht darin, einen inkrementellen Modell-Workflow zu verwenden, und eine andere darin, das Modell mit einem Warmstart neu zu trainieren.

Workflow mit inkrementellem Modell

Bei diesem Workflow aktualisieren Sie die ML-Prognosen, ohne das ML-Modell erneut zu trainieren.

> **Note**
>
> Dies ist nur möglich, wenn die Grafikdaten mit neuen Knoten und/oder Edges aktualisiert wurden. Es funktioniert derzeit nicht, wenn Knoten entfernt werden.

1. Datenexport und Konfiguration – Dieser Schritt ist identisch mit dem im Haupt-Workflow.
2. Inkrementelle Datenvorverarbeitung – Dieser Schritt ähnelt dem Schritt der Datenvorverarbeitung im Haupt-Workflow, verwendet jedoch dieselbe Verarbeitungskonfiguration wie zuvor, die einem bestimmten trainierten Modell entspricht.
3. Modelltransformation – Anstatt eines Modelltrainingsschritts nimmt dieser Schritt der Modelltransformation das trainierte Modell aus dem Haupt-Workflow und den Ergebnissen des Schritts der inkrementellen Datenvorverarbeitung und generiert neue Modellartefakte, die für Inferenzen verwendet werden können. Im Schritt zur Modelltransformation wird ein SageMaker KI-Verarbeitungsjob gestartet, um die Berechnung durchzuführen, die die aktualisierten Modellartefakte generiert.
4. Amazon SageMaker AI-Inferenzendpunkt aktualisieren — Wenn Sie über einen vorhandenen Inferenzendpunkt verfügen, aktualisiert dieser Schritt optional den Endpunkt mit den neuen Modellartefakten, die durch den Modelltransformationsschritt generiert wurden. Alternativ können Sie zudem einen neuen Inferenzendpunkt mit den neuen Modellartefakten erstellen.

Erneutes Training des Modells mit Warmstart

Mit diesem Workflow können Sie ein neues ML-Modell für die Erstellung von Prognosen anhand der inkrementellen Grafikdaten trainieren und bereitstellen. Sie können jedoch mit einem vorhandenen Modell beginnen, das mit dem Haupt-Workflow generiert wurde:

1. Datenexport und Konfiguration – Dieser Schritt ist identisch mit dem im Haupt-Workflow.

2. Inkrementelle Datenvorverarbeitung – Dieser Schritt ist identisch mit dem beim inkrementellen Modellinferenz-Workflow. Die neuen Graphdaten sollten mit derselben Verarbeitungsmethode verarbeitet werden, die zuvor für das Modelltraining verwendet wurde.
3. Modelltraining mit Warmstart – Das Modelltraining ähnelt dem Training im Haupt-Workflow, Sie können jedoch die Suche nach Modell-Hyperparametern beschleunigen, indem Sie die Informationen aus der vorherigen Modelltrainingsaufgabe nutzen.
4. Den Amazon SageMaker AI-Inferenzendpunkt aktualisieren — Dieser Schritt ist derselbe wie beim inkrementellen Model-Inferenz-Workflow.

Workflows für benutzerdefinierte Modelle in Neptune ML

Mit Neptune ML können Sie eigene benutzerdefinierte Modelle für alle Aufgaben implementieren, trainieren und einsetzen, die Neptune ML unterstützt. Der Workflow für die Entwicklung und Bereitstellung eines benutzerdefinierten Modells ist im Wesentlichen derselbe wie für die integrierten Modelle, mit einigen Unterschieden, wie unter Workflow für benutzerdefinierte Modelle erläutert.

Auswahl der Instance für die Neptune-ML-Phasen

Die verschiedenen Phasen der Neptune ML-Verarbeitung verwenden unterschiedliche SageMaker KI-Instanzen. Im Folgenden wird beschrieben, wie Sie den richtigen Instance-Typ für die einzelnen Phasen auswählen. Informationen zu SageMaker KI-Instance-Typen und Preisen finden Sie unter Amazon SageMaker Pricing.

Auswahl einer Instance für die Datenverarbeitung

Für den SageMaker KI-Datenverarbeitungsschritt ist eine Verarbeitungsinstanz erforderlich, die über ausreichend Arbeitsspeicher und Festplattenspeicher für die Eingabe-, Zwischen- und Ausgabedaten verfügt. Die spezifische Menge an Arbeits- und Festplattenspeicher, die benötigt wird, ist von den Eigenschaften des Neptune-ML-Diagramms und dessen exportierten Features abhängig.

Standardmäßig wählt Neptune ML die kleinste `ml.r5`-Instance aus, deren Arbeitsspeicher zehnmal größer als die Größe der exportierten Diagrammdaten auf der Festplatte ist.

Auswahl einer Instance für Modelltraining und Modelltransformation

Die Auswahl des richtigen Instance-Typs für Modelltraining oder Modelltransformation ist vom Typ der Aufgabe, der Größe des Diagramms und den Anforderungen an die Bearbeitungszeit abhängig. GPU-Instances stellen die beste Leistung bereit. Wir empfehlen grundsätzlich serielle `p3`- und `g4dn`-Instances. Sie können auch `p2`- oder `p4d`-Instances verwenden.

Standardmäßig wählt Neptune ML die kleinste GPU-Instance aus, deren Arbeitsspeicher größer als von Modelltraining und Modelltransformation benötigt ist. Informationen zu dieser Auswahl finden Sie in der Datei `train_instance_recommendation.json` im Amazon-S3-Datenverarbeitungsausgabeort. Dies ist ein Beispiel für den Inhalt einer `train_instance_recommendation.json`-Datei:

```
{
  "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)"
}
```

Auswahl einer Instance für einen Inferenzendpunkt

Die Auswahl des richtigen Instance-Typs für einen Inferenzendpunkt ist vom Typ der Aufgabe, der Größe des Diagramms und dem Budget abhängig. Standardmäßig wählt Neptune ML die kleinste `ml.m5d`-Instance aus, deren Arbeitsspeicher größer als vom Inferenzendpunkt benötigt ist.

> **Note**
>
> Wenn mehr als 384 GB Arbeitsspeicher benötigt werden, verwendet Neptune ML eine `ml.r5d.24xlarge`-Instance.

Informationen zum von Neptune ML empfohlenen Instance-Typ finden Sie in der Datei `infer_instance_recommendation.json` am Amazon-S3-Ort, den Sie zum Modelltraining verwenden. Dies ist ein Beispiel für den Inhalt dieser Datei:

```
{
  "instance" :   "(the recommended instance type for an inference endpoint)",
  "disk_size" :  "(the estimated disk space required)",
  "mem_size" :   "(the estimated memory required)"
}
```

Verwenden des Neptune-Export-Tools oder des Neptune-Export-Service zum Exportieren von Daten aus Neptune für Neptune ML

Neptune ML erfordert, dass Sie Trainingsdaten für die Deep Graph Library (DGL) bereitstellen, um Modelle zu erstellen und auszuwerten.

Sie können mit dem Neptune-Export-Service oder mit dem `neptune-export`-Hilfsprogramm Daten aus Neptune exportieren. Sowohl der Service als auch das Befehlszeilentool veröffentlichen Daten zu Amazon Simple Storage Service (Amazon S3) im CSV-Format, die mit serverseitiger Amazon-S3-Verschlüsselung (`SSE-S3`) verschlüsselt sind. Siehe Von Neptune-Export exportierte Dateien und `neptune-export`.

Wenn Sie einen Export von Trainingsdaten für Neptune ML konfigurieren, erstellt und veröffentlicht der Exportauftrag eine verschlüsselte Modelltrainings-Konfigurationsdatei zusammen mit den exportierten Daten. Diese Datei hat standardmäßig den Namen `training-data-configuration.json`.

Beispiele für die Verwendung des Neptune-Export-Service zum Exportieren von Trainingsdaten für Neptune ML

Diese Anforderung exportiert Eigenschaftsdiagramm-Trainingsdaten für eine Aufgabe zur Knotenklassifizierung:

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**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. AWS Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

Diese Anforderung exportiert RDF-Trainingsdaten für eine Aufgabe zur Knotenklassifizierung:

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**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

Felder, die beim Exportieren von Trainingsdaten im Params-Objekt festgelegt werden sollen

Das `params`-Objekt in einer Exportanforderung kann verschiedene Felder enthalten, wie in der `params`-Dokumentation beschrieben. Die folgenden Felder sind für den Export von Machine-Learning-Trainingsdaten am relevantesten:

- **`endpoint`** – Verwenden Sie `endpoint`, um den Endpunkt einer Neptune-Instance in Ihrem DB-Cluster anzugeben, den der Exportvorgang abfragen kann, um Daten zu extrahieren.

- **`profile`** – Das Feld `profile` im `params`-Objekt muss auf **`neptune-ml`** festgelegt sein.

  Dadurch formatiert der Exportvorgang die exportierten Daten für das Neptune-ML-Modelltraining im CSV-Format für Eigenschaftsdiagrammdaten oder als N-Tripel für RDF-Daten. Außerdem wird die Datei `training-data-configuration.json` erstellt und zum selben Amazon-S3-Speicherort geschrieben wie die exportierten Trainingsdaten.
- **`cloneCluster`** – Wenn diese Option auf `true` festgelegt ist, klont der Exportvorgang den DB-Cluster, exportiert Daten aus dem Klon und löscht den Klon, wenn der Vorgang abgeschlossen ist.
- **`useIamAuth`** – Wenn für den DB-Cluster die IAM-Authentifizierung aktiviert ist, müssen Sie dieses Feld auf `true` festlegen.

Der Exportvorgang bietet auch mehrere Möglichkeiten für das Filtern der exportierten Daten (siehe diese Beispiele).

Verwenden Sie das AdditionalParams-Objekt, um den Export von Modelltrainingsinformationen zu optimieren

Das `additionalParams`-Objekt enthält Felder, mit denen Sie Machine-Learning-Klassenbezeichnungen und -Features zu Trainingszwecken angeben und die Erstellung einer Trainingsdaten-Konfigurationsdatei steuern können.

Der Exportvorgang kann nicht automatisch ableiten, welche Knoten- und Kanteneigenschaften als Beispiele für Machine-Learning-Klassenbezeichnungen zu Trainingszwecken dienen sollen. Außerdem kann er nicht automatisch die beste Feature-Kodierung für numerische, kategorische und Texteigenschaften ableiten. Daher müssen Sie in den Feldern des `additionalParams`-Objekts Hinweise bereitstellen, um dies anzugeben oder die Standardkodierung zu überschreiben.

Für Eigenschaftsdiagrammdaten könnte die Struktur der obersten Ebene von `additionalParams` in einer Exportanforderung wie folgt aussehen:

```
{
  "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) ]
    }
  }
}
```

Für RDF-Daten könnte die Struktur der obersten Ebene wie folgt aussehen:

```
{
  "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) ]
    }
  }
}
```

Sie können im Feld `jobs` auch mehrere Exportkonfigurationen angeben:

```
{
  "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) ]
        }
      ]
    }
  }
}
```

Elemente der obersten Ebene im Feld neptune_ml in AdditionalParams

Das Versionselement in neptune_ml

Gibt die Version der Trainingsdatenkonfiguration an, die generiert werden soll.

(Optional), Typ: string, Standard: v2.0.

Wenn Sie `version` einfügen, legen Sie dies auf `v2.0` fest.

Das Jobfeld in neptune_ml

Enthält eine Reihe von Trainingsdaten-Konfigurationsobjekten, von denen jedes einen Datenverarbeitungsauftrag definiert, und enthält:

- **`name`** – Der Name der Trainingsdatenkonfiguration, die erstellt werden soll.

  Beispielsweise ergibt eine Trainingsdatenkonfiguration mit dem Namen „"job-number-1“ eine Trainingsdaten-Konfigurationsdatei mit dem Namen `job-number-1.json`.
- **`targets`** – Ein JSON-Array mit Knoten- und Kanten-Klassenbezeichnungszielen, die die Machine-Learning-Klassenbezeichnungen für Trainingszwecke darstellen. Siehe Das Zielfeld in einem neptune_ml-Objekt.
- **`features`** – Ein JSON-Array mit Knoteneigenschaft-Features. Siehe Das Feature-Feld in neptune_ml.

Das Zielfeld in einem neptune_ml-Objekt

Das Feld `targets` in einer Exportkonfiguration für JSON-Trainingsdaten enthält mehrere Zielobjekte, die eine Trainingsaufgabe spezifizieren, sowie die Machine-Learning-Klassenbezeichnungen zum Trainieren dieser Aufgabe. Der Inhalt der Zielobjekte ist davon abhängig, ob Sie mit Eigenschaftsdiagrammdaten oder RDF-Daten trainieren.

Für Aufgaben zur Klassifizierung und Regression von Eigenschaftsdiagramm-Knoten können die Zielobjekte im Array wie folgt aussehen:

```
{
  "node": "(node property-graph label)",
  "property": "(property name)",
  "type" : "(used to specify classification or regression)",
  "split_rate": [0.8,0.2,0.0],
  "separator": ","
}
```

Für Aufgaben zur Klassifizierung, Regression oder Linkvorhersage von Eigenschaftsdiagramm-Kanten können sie wie folgt aussehen:

```
{
  "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": ","
}
```

Für Aufgaben zur RDF-Klassifizierung und -Regression können die Zielobjekte im Array wie folgt aussehen:

```
{
  "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]
}
```

Für Aufgaben zur RDF-Linkvorhersage können Zielobjekte im Array wie folgt aussehen:

```
{
  "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]
}
```

Zielobjekte können die folgenden Felder enthalten:

Inhalt

- Felder in einem Eigenschaftsdiagramm-Zielobjekt
  - Das Knotenfeld (Vertex) in einem Zielobjekt
  - Das Kantenfeld in einem Eigenschaftsgraph-Zielobjekt
  - Das Eigenschaftsfeld in einem Eigenschaftsgraph-Zielobjekt
  - Das Typfeld in einem Eigenschaftsdiagramm-Zielobjekt
  - Das Split_Rate-Feld in einem Property-Graph-Zielobjekt
  - Das Trennfeld in einem Zielobjekt mit einem Eigenschaftsdiagramm
- Felder in einem RDF-Zielobjekt
  - Das Knotenfeld in einem RDF-Zielobjekt
  - Das Betreff-Feld in einem RDF-Zielobjekt
  - Das Prädikatfeld in einem RDF-Zielobjekt
  - Das Objektfeld in einem RDF-Zielobjekt
  - Das Typfeld in einem RDF-Zielobjekt
  - Das Feld split_rate in einem Eigenschaftsdiagramm-Zielobjekt

Felder in einem Eigenschaftsdiagramm-Zielobjekt

Das Knotenfeld (Vertex) in einem Zielobjekt

Die Eigenschaftsdiagramm-Bezeichnung eines Zielknotens (Eckpunkt). Ein Zielobjekt muss ein `node`-Element oder ein `edge`-Element enthalten, darf aber nicht beide Elemente enthalten.

A `node` kann einen einzelnen Wert annehmen, z. B.:

```
"node": "Movie"
```

Im Fall eines Eckpunkts mit mehreren Bezeichnungen kann er mehrere Werte annehmen, z. B.:

```
"node": ["Content", "Movie"]
```

Das Kantenfeld in einem Eigenschaftsgraph-Zielobjekt

Gibt eine Zielkante anhand der Startknotenbezeichnung(en), der eigenen Bezeichnung und der Endknotenbezeichnung(en) an. Ein Zielobjekt muss ein `edge`-Element oder ein `node`-Element enthalten, darf aber nicht beide Elemente enthalten.

Der Wert eines `edge`-Felds ist ein JSON-Array mit drei Zeichenfolgen, die die Eigenschaftsgraphenbezeichnungen des Startknotens, die Eigenschaftsgraphenbezeichnung der Kante selbst und die Eigenschaftsgraphenbeschriftung(en) des Endknotens darstellen, z. B.:

```
"edge": ["Person_A", "knows", "Person_B"]
```

Wenn der Startknoten und der and/or Endknoten mehrere Beschriftungen hat, schließen Sie sie in ein Array ein, etwa so:

```
"edge": [ ["Admin", Person_A"], "knows", ["Admin", "Person_B"] ]
```

Das Eigenschaftsfeld in einem Eigenschaftsgraph-Zielobjekt

Gibt eine Eigenschaft des Zieleckpunkts oder der Zielkante wie folgt an:

```
"property" : "rating"
```

Dieses Feld ist erforderlich, es sei denn, es handelt sich bei der Zielaufgabe um eine Linkvorhersage.

Das Typfeld in einem Eigenschaftsdiagramm-Zielobjekt

Gibt den Typ der Zielaufgabe an, die für `node` oder `edge` ausgeführt werden soll, z. B.:

```
"type" : "regression"
```

Die unterstützten Aufgabentypen für Knoten sind:

- `classification`
- `regression`

Die unterstützten Aufgabentypen für Kanten sind:

- `classification`
- `regression`
- `link_prediction`

Dies ist ein Pflichtfeld.

Das Split_Rate-Feld in einem Property-Graph-Zielobjekt

(Optional) Eine Schätzung des Anteils der Knoten oder Kanten, die in der Trainings-, Validierungs- und Testphase jeweils verwendet werden. Diese Anteile werden durch ein JSON-Array mit drei Zahlen zwischen null und eins dargestellt, die zusammen eins ergeben:

```
"split_rate": [0.7, 0.1, 0.2]
```

Wenn Sie das optionale `split_rate` Feld nicht angeben, gilt der geschätzte Standardwert `[0.9, 0.1, 0.0]` für Klassifizierungs- und Regressionsaufgaben sowie `[0.9,0.05, 0.05]` für Aufgaben zur Linkvorhersage.

Das Trennfeld in einem Zielobjekt mit einem Eigenschaftsdiagramm

(Optional) Wird zusammen mit einer Klassifizierungsaufgabe verwendet.

Das Feld `separator` gibt ein Zeichen an, das zum Aufteilen eines Zieleigenschaftswerts in mehrere kategoriale Werte verwendet wird, wenn es zum Speichern mehrerer Kategoriewerte in einer Zeichenfolge verwendet wird. Beispiel:

```
"separator": "|"
```

Das Vorhandensein des Felds `separator` weist darauf hin, dass es sich bei der Aufgabe um eine Klassifizierungsaufgabe mit mehreren Zielen handelt.

Felder in einem RDF-Zielobjekt

Das Knotenfeld in einem RDF-Zielobjekt

Definiert den Knotentyp der Zielknoten. Wird bei Aufgaben zur Klassifizierung oder Regression von Knoten verwendet. Der Knotentyp eines Knotens in RDF wird wie folgt definiert:

```
node_id, <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>, node_type
```

Ein `node`-Knoten kann nur einen Wert annehmen, z. B.:

```
"node": "http://aws.amazon.com/neptune/csv2rdf/class/Movie"
```

Das Betreff-Feld in einem RDF-Zielobjekt

Bei Linkvorhersageaufgaben definiert `subject` den Quellknotentyp von Zielkanten.

```
"subject": "http://aws.amazon.com/neptune/csv2rdf/class/Director"
```

> **Note**
>
> Bei Linkvorhersageaufgaben sollte `subject` zusammen mit `predicate` und `object` verwendet werden. Wenn keiner dieser drei Werte angegeben ist, werden alle Kanten als Trainingsziel behandelt.

Das Prädikatfeld in einem RDF-Zielobjekt

Bei Aufgaben zur Klassifizierung und Regression von Knoten definiert `predicate` die Literaldaten, die als Zielknoten-Feature eines Zielknotens verwendet werden.

```
"predicate": "http://aws.amazon.com/neptune/csv2rdf/datatypeProperty/genre"
```

> **Note**
>
> Wenn die Zielknoten nur ein Prädikat besitzen, das das Zielknoten-Feature definiert, kann das Feld `predicate` übersprungen werden.

Bei Linkvorhersageaufgaben definiert `predicate` den Beziehungstyp von Zielkanten.

```
"predicate": "http://aws.amazon.com/neptune/csv2rdf/datatypeProperty/direct"
```

> **Note**
>
> Bei Linkvorhersageaufgaben sollte `predicate` zusammen mit `subject` und `object` verwendet werden. Wenn keiner dieser drei Werte angegeben ist, werden alle Kanten als Trainingsziel behandelt.

Das Objektfeld in einem RDF-Zielobjekt

Bei Linkvorhersageaufgaben definiert `object` den Zielknotentyp von Zielkanten:

```
  "object": "http://aws.amazon.com/neptune/csv2rdf/class/Movie"
```

> **Note**
>
> Bei Linkvorhersageaufgaben sollte `object` zusammen mit `subject` und `predicate` verwendet werden. Wenn keiner dieser drei Werte angegeben ist, werden alle Kanten als Trainingsziel behandelt.

Das Typfeld in einem RDF-Zielobjekt

Gibt den Typ der Zielaufgabe an, die ausgeführt werden soll, z. B.:

```
  "type" : "regression"
```

Die unterstützten Aufgabentypen für RDF-Daten sind:

- `link_prediction`
- `classification`
- `regression`

Dies ist ein Pflichtfeld.

Das Feld **`split_rate`** in einem Eigenschaftsdiagramm-Zielobjekt

(Optional) Eine Schätzung des Anteils der Knoten oder Kanten, die in der Trainings-, Validierungs- und Testphase jeweils verwendet werden. Diese Anteile werden durch ein JSON-Array mit drei Zahlen zwischen null und eins dargestellt, die zusammen eins ergeben:

```
"split_rate": [0.7, 0.1, 0.2]
```

Wenn Sie keinen Wert für das optionale `split_rate`-Feld angeben, ist der geschätzte Standardwert `[0.9, 0.1, 0.0]`.

Das Feature-Feld in neptune_ml

Eigenschaftswerte und RDF-Literale haben verschiedene Formate und Datentypen. Um eine gute Machine-Learning-Leistung zu erzielen, müssen diese Werte in numerische Kodierungen konvertiert werden, so genannte features.

Neptune ML extrahiert und kodiert Features als Teil der Schritte für Datenexport und Datenverarbeitung durch, wie in Feature-Kodierung in Neptune ML beschrieben.

Für Eigenschaftsdiagramm-Datensätze inferiert der Exportvorgang automatisch `auto`-Features für Zeichenfolgeneigenschaften und numerische Eigenschaften, die mehrere Werte enthalten. Für numerische Eigenschaften, die Einzelwerte enthalten, werden `numerical`-Features inferiert. Für Datumseigenschaften werden `datetime`-Features inferiert.

Wenn Sie eine automatisch abgeleitete Feature-Spezifikation überschreiben oder eine numerische Bucket-, TF-IDF- oder SBERT-Spezifikation für eine Eigenschaft hinzufügen möchten FastText, können Sie die Feature-Codierung mithilfe des Features-Felds steuern.

> **Note**
>
> Sie können das Feld `features` nur verwenden, um die Feature-Spezifikationen für Eigenschaftsdiagrammdaten zu steuern, nicht für RDF-Daten.

Für Freiformtext kann Neptune ML mehrere verschiedene Modelle verwenden, um die Sequenz von Token in einem Zeichenfolgen-Eigenschaftswert in einen Realwertvektor fester Größe zu konvertieren:

- `text_fasttext` – Verwendet fastText-Kodierung. Dies ist die empfohlene Kodierung für Features, die eine und nur eine der fünf Sprachen verwenden, die fastText unterstützt.
- `text_sbert` – Verwendet die Kodierungsmodelle Sentence BERT (SBERT). Dies ist die empfohlene Kodierung für Text, den `text_fasttext` nicht unterstützt.
- `text_word2vec` – Verwendet Word2Vec-Algorithmen (ursprünglich von Google veröffentlicht), um Text zu kodieren. Word2Vec unterstützt nur Englisch.
- `text_tfidf` – Verwendet den Vektorisierer term frequency–inverse document frequency (TF-IDF) für die Textkodierung. Die TF-IDF-Kodierung unterstützt statistische Features, die von den anderen Kodierungen nicht unterstützt werden.

Das Feld `features` enthält ein JSON-Array mit Knoteneigenschaft-Features. Objekte im Array können die folgenden Felder enthalten:

Inhalt

- Das Knotenfeld in Features
- Das Kantenfeld in Features
- Das Eigenschaftsfeld in Features
- Mögliche Werte des Typfeldes für Features
- Das Normfeld
- Das Sprachfeld
- Das Feld max_length
- Das Trennfeld
- Das Bereichsfeld
- Das Feld bucket_cnt
- Das Feld slide_window_size
- Das Imputer-Feld
- Das Feld max_features
- Das Feld min_df
- Das Feld ngram_range
- Das Feld datetime_parts

Das Knotenfeld in Features

Das Feld node gibt die Eigenschaftsdiagramm-Bezeichnung eines Feature-Eckpunkts an. Beispiel:

```
  "node": "Person"
```

Wenn ein Eckpunkt mehrere Bezeichnungen hat, verwenden Sie ein Array, um sie aufzunehmen. Beispiel:

```
  "node": ["Admin", "Person"]
```

Das Kantenfeld in Features

Das Feld `edge` gibt den Kantentyp einer Feature-Kante an. Ein Kantentyp besteht aus einem Array, das die Eigenschaftsdiagramm-Bezeichnung(en) des Quelleckpunkts, der Kante und des Zieleckpunkts enthält. Sie müssen alle drei Werte angeben, wenn Sie ein Kanten-Feature angeben. Beispiel:

```
"edge": ["User", "reviewed", "Movie"]
```

Wenn ein Quell- oder Zieleckpunkt eines Kantentyps mehrere Bezeichnungen hat, verwenden Sie ein weiteres Array. Beispiel:

```
"edge": [["Admin", "Person"]. "edited", "Post"]
```

Das Eigenschaftsfeld in Features

Mit dem Eigenschaftsparameter können Sie eine Eigenschaft des Eckpunkts angeben, der vom Parameter `node` identifiziert wird. Beispiel:

```
"property" : "age"
```

Mögliche Werte des Typfeldes für Features

Der Parameter `type` gibt den Typ des Features an, das definiert wird. Beispiel:

```
"type": "bucket_numerical"
```

Mögliche Werte des Parameters **`type`**

- **`"auto"`** – Gibt an, dass Neptune ML den Eigenschaftstyp automatisch erkennen und die korrekte Feature-Kodierung anwenden soll. Ein `auto`-Feature kann auch das optionale Feld `separator` besitzen.

  Siehe Auto-Feature-Kodierung in Neptune ML.
- **`"category"`** – Diese Feature-Kodierung stellt einen Eigenschaftswert als eine von mehreren Kategorien dar. Mit anderen Worten, das Feature kann einen oder mehrere diskrete Werte annehmen. Ein `category`-Feature kann auch das optionale Feld `separator` besitzen.

Siehe Kategorische Features in Neptune ML.

- **`"numerical"`** – Diese Feature-Kodierung stellt numerische Eigenschaftswerte als Zahlen in einem kontinuierlichen Intervall dar, wobei „größer als“ und „kleiner als“ eine Bedeutung haben.

  Ein `numerical`-Feature kann auch die optionalen Felder `norm`, `imputer` und `separator` besitzen.

  Siehe Numerische Features in Neptune ML.

- **`"bucket_numerical"`** – Diese Feature-Kodierung unterteilt numerische Eigenschaftswerte in eine Reihe von Buckets oder Kategorien.

  Sie könnten beispielsweise das Alter von Personen in 4 Buckets unterteilen: Kinder (0–20), junge Erwachsene (20–40), Personen mittleren Alters (40–60) und ältere Menschen (60 und älter).

  Ein `bucket_numerical` Feature erfordert ein `range` `bucket_cnt` Feld und kann optional auch ein `imputer` and/or `slide_window_size` Feld enthalten.

  Siehe Bucket-numerische Features in Neptune ML.

- **`"datetime"`** – Diese Feature-Kodierung stellt den datetime-Eigenschaftswert als Array dieser Kategorie-Features dar: Jahr, Monat, Wochentag und Stunde.

  Eine oder mehrere dieser vier Kategorien können mithilfe des Parameters `datetime_parts` eliminiert werden.

  Siehe Datetime-Features in Neptune ML.

- **`"text_fasttext"`** – Diese Feature-Kodierung konvertiert Eigenschaftswerte, die aus Sätzen oder Freiformtext bestehen, mithilfe von fastText-Modellen in numerische Vektoren. Sie unterstützt fünf Sprachen, nämlich Englisch (`en`), Chinesisch (`zh`), Hindi (`hi`), Spanisch (`es`) und Französisch (`fr`). Für Texteigenschaftswerte in einer dieser fünf Sprachen wird die Kodierung `text_fasttext` empfohlen. Sie kann jedoch nicht für Fälle verwendet werden, in denen derselbe Satz Wörter in mehr als einer Sprache enthält.

  Für Sprachen, die nicht von fastText unterstützt werden, verwenden Sie die `text_sbert`-Kodierung.

  Wenn es zahlreiche Eigenschaftswert-Textzeichenfolgen mit mehr als beispielsweise 120 Token gibt, verwenden Sie das Feld `max_length`, um die Anzahl der Token in jeder von `"text_fasttext"` kodierten Zeichenfolge zu begrenzen.

Siehe fastText-Kodierung von Texteigenschaftswerten in Neptune ML.

- **`"text_sbert"`** – Diese Kodierung konvertiert Texteigenschaftswerte mithilfe von Sentence BERT (SBERT)-Modellen in numerische Vektoren. Neptune unterstützt zwei SBERT-Methoden: `text_sbert128` (die Standardmethode, wenn Sie nur `text_sbert` angeben) und `text_sbert512`. Der Unterschied zwischen ihnen besteht in der maximalen Anzahl von Token in einer Texteigenschaft, die kodiert wird. Die `text_sbert128`-Kodierung kodiert nur die ersten 128 Token, während `text_sbert512` bis zu 512 Token kodiert. Daher kann die Verwendung von `text_sbert512` mehr Verarbeitungszeit als `text_sbert128` in Anspruch nehmen. Beide Methoden sind langsamer als `text_fasttext`.

  Die `text_sbert*`-Methoden unterstützen zahlreiche Sprachen und können einen Satz kodieren, der mehr als eine Sprache enthält.

  Siehe Sentence BERT (SBERT)-Satzkodierung von Text-Features in Neptune ML.

- **`"text_word2vec"`** – Diese Kodierung konvertiert Texteigenschaftswerte mithilfe von Word2Vec-Algorithmen in numerische Vektoren. Sie unterstützt nur Englisch.

  Siehe Word2Vec-Kodierung von Text-Features in Neptune ML.

- **`"text_tfidf"`** – Bei dieser Kodierung werden Texteigenschaftswerte mithilfe des Vektorisierers term frequency–inverse document frequency (TF-IDF) in numerische Vektoren konvertiert.

  Sie definieren die Parameter einer `text_tfidf`-Feature-Kodierung mithilfe der Felder `ngram_range`, `min_df` und `max_features`.

  Siehe TF-IDF-Kodierung von Text-Features in Neptune ML.

- **`"none"`** – Die Verwendung des Typs `none` führt dazu, dass keine Feature-Kodierung erfolgt. Stattdessen werden die rohen Eigenschaftswerte geparst und gespeichert.

  Verwenden Sie `none` nur, wenn Sie Ihre eigene benutzerdefinierte Feature-Kodierung als Teil eines benutzerdefinierten Modelltrainings ausführen möchten.

Das Normfeld

Dies ist ein Pflichtfeld für numerische Features. Es gibt eine Normalisierungsmethode für numerische Werte an:

```
"norm": "min-max"
```

Die folgenden Normalisierungsmethoden werden unterstützt:

- "min-max" – Normalisiert jeden Wert, indem der Mindestwert subtrahiert wird und der Wert dann durch die Differenz zwischen Maximalwert und Mindestwert dividiert wird.
- "standard" – Normalisiert jeden Wert, indem er durch die Summe aller Werte dividiert wird.
- "none" – Normalisiert die numerischen Werte während der Kodierung nicht.

Siehe Numerische Features in Neptune ML.

Das Sprachfeld

Das Feld für die Sprache gibt die Sprache an, die in Texteigenschaftswerten verwendet wird. Die Nutzung ist von der Textkodierungsmethode abhängig:

- Dieses Feld ist für die `text_fasttext`-Kodierung erforderlich und es muss eine der folgenden Sprachen angegeben werden:
  - `en` (Englisch)
  - `zh` (Chinesisch)
  - `hi` (Hindi)
  - `es` (Spanisch)
  - `fr` (Französisch)
- Dieses Feld wird für die `text_sbert`-Kodierung nicht verwendet, da die SBERT-Kodierung mehrsprachig ist.
- Dieses Feld ist für die `text_word2vec`-Kodierung optional, da `text_word2vec` nur Englisch unterstützt. Wenn vorhanden, muss der Name des englischsprachigen Modells angegeben werden:

  ```
  "language" : "en_core_web_lg"
  ```

- Dieses Feld wird für die `text_tfidf`-Kodierung nicht verwendet.

Das Feld max_length

Das Feld `max_length` ist optional für `text_fasttext`-Features, wo es die maximale Anzahl von Token in einem Eingabe-Text-Feature angibt, die kodiert werden. Eingabetext, der länger als `max_length` ist, wird abgeschnitten. Wenn Sie beispielsweise max_length auf 128 festlegen, werden alle Token in einer Textsequenz nach dem 128. Zeichen ignoriert:

```
"max_length": 128
```

Das Trennfeld

Dieses Feld wird optional mit den Features `category`, `numerical` und `auto` verwendet. Es gibt ein Zeichen an, das für die Unterteilung eines Eigenschaftswerts in mehrere kategorische oder numerische Werte verwendet werden kann:

```
"separator": ";"
```

Verwenden Sie das Feld `separator` nur, wenn die Eigenschaft mehrere durch Trennzeichen getrennte Werte in einer einzigen Zeichenfolge speichert, z. B. `"Actor;Director"` oder `"0.1;0.2"`.

Siehe Kategorische Features, Numerische Features und Auto-Verschlüsselung.

Das Bereichsfeld

Dies ist ein Pflichtfeld für `bucket_numerical`-Features. Es gibt den Bereich der numerischen Werte an, die in Buckets unterteilt werden sollen, und zwar im folgenden Format `[`*`lower-bound, upper-bound`*`]`:

```
"range" : [20, 100]
```

Wenn ein Eigenschaftswert kleiner als der untere Grenzwert ist, wird er dem ersten Bucket zugewiesen. Wenn er größer als der obere Grenzwert ist, wird er dem letzten Bucket zugewiesen.

Siehe Bucket-numerische Features in Neptune ML.

Das Feld bucket_cnt

Dies ist ein Pflichtfeld für `bucket_numerical`-Features. Es gibt die Anzahl der Buckets an, in die der numerische Bereich unterteilt werden soll, der durch den Parameter `range` definiert wird:

```
"bucket_cnt": 10
```

Siehe Bucket-numerische Features in Neptune ML.

Das Feld slide_window_size

Dieses Feld wird optional mit `bucket_numerical`-Features verwendet, um mehr als einem Bucket Werte zuzuweisen:

```
"slide_window_size": 5
```

Ein Gleitfenster funktioniert, indem Neptune ML die Fenstergröße **s** übernimmt und jeden numerischen Wert **v** einer Eigenschaft in einen Bereich von `v - s/2` bis `v + s/2` transformiert. Der Wert wird dann jedem Bucket zugewiesen, in dem sich der Bereich überschneidet.

Siehe Bucket-numerische Features in Neptune ML.

Das Imputer-Feld

Dieses Feld wird optional mit `numerical`- und `bucket_numerical`-Features verwendet, um eine Imputationstechnik zum Ausfüllen fehlender Werte bereitzustellen:

```
"imputer": "mean"
```

Die folgenden Imputationstechniken werden unterstützt:

- `"mean"`
- `"median"`
- `"most-frequent"`

Wenn Sie diesen Parameter nicht angeben, wird die Datenvorverarbeitung angehalten und beendet, wenn ein fehlender Wert gefunden wird.

Siehe Numerische Features in Neptune ML und Bucket-numerische Features in Neptune ML.

Das Feld max_features

Dieses Feld wird optional von `text_tfidf`-Features verwendet, um die maximale Anzahl der Begriffe anzugeben, die kodiert werden sollen:

```
"max_features": 100
```

Die Einstellung 100 bewirkt, dass der TF-IDF-Vektorizer nur die 100 häufigsten Begriffe kodiert. Wenn Sie `max_features` nicht angeben, ist der Standardwert 5 000.

Siehe TF-IDF-Kodierung von Text-Features in Neptune ML.

Das Feld min_df

Dieses Feld wird optional von `text_tfidf`-Features verwendet, um die Mindesthäufigkeit von Begriffen anzugeben, die kodiert werden sollen:

```
"min_df": 5
```

Die Einstellung 5 bewirkt, dass ein Begriff in mindestens 5 verschiedenen Eigenschaftswerten vorkommen muss, um kodiert zu werden.

Wenn Sie den Parameter `min_df` nicht angeben, ist der Standardwert 2.

Siehe TF-IDF-Kodierung von Text-Features in Neptune ML.

Das Feld ngram_range

Dieses Feld wird optional von `text_tfidf`-Features verwendet, um die Größe anzugeben, die Wortsequenzen oder Token haben sollten, um als potenzielle einzelne Begriffe kodiert zu werden.

```
"ngram_range": [2, 4]
```

Der Wert `[2, 4]` gibt an, dass Sequenzen von 2, 3 und 4 Wörtern als potenzielle einzelne Begriffe betrachtet werden sollen.

Wenn Sie `ngram_range` nicht explizit angeben, ist der Standardwert `[1, 1]`. Das bedeutet, dass nur einzelne Wörter oder Token kodiert werden.

Siehe TF-IDF-Kodierung von Text-Features in Neptune ML.

Das Feld datetime_parts

Dieses Feld wird optional von `datetime`-Features verwendet, um anzugeben, welche Teile des datetime-Werts kategorisch kodiert werden sollen:

```
"datetime_parts": ["weekday", "hour"]
```

Wenn Sie `datetime_parts` nicht angeben, kodiert Neptune ML standardmäßig die Teile für Jahr, Monat, Wochentag und Stunden des datetime-Werts. Der Wert `["weekday", "hour"]` gibt an, dass nur die datetime-Werte für Wochentag und Stunde kategorisch im Feature kodiert werden sollen.

Wenn ein Teil nicht mehr als einen eindeutigen Wert im Trainingssatz hat, wird er nicht kodiert.

Siehe Datetime-Features in Neptune ML.

Beispiele für die Verwendung von Parametern in additionalParams zur Optimierung der Modelltrainingskonfiguration

Die folgenden Beispiele zeigen, wie die Funktion „AdditionalParams“ in Property-Graph- und RDF-Datenmodellen verwendet werden kann, um verschiedene Aspekte des Modelltrainingsprozesses für eine Neptune ML-Anwendung zu konfigurieren. Die Beispiele decken ein breites Funktionsspektrum ab, darunter die Angabe von Standard-Split-Raten für training/validation/test Daten, die Definition von Aufgaben zur Knotenklassifizierung, Regression und Verbindungsvorhersage sowie die Konfiguration verschiedener Feature-Typen wie numerische Buckets, Texteinbettungen, Datetime und kategoriale Daten. Diese detaillierten Konfigurationen ermöglichen es Ihnen, die Machine-Learning-Pipeline an Ihre spezifischen Daten- und Modellierungsanforderungen anzupassen und so das volle Potenzial der Neptune ML-Funktionen auszuschöpfen.

Inhalt

- Beispiele für Eigenschaftsdiagramme mit AdditionalParams
  - Angabe einer Standard-Aufteilungsrate für die Modelltrainingskonfiguration
  - Angabe einer Knotenklassifizierungsaufgabe für die Modelltrainingskonfiguration
  - Angabe einer Klassifizierungsaufgabe für Knoten mit mehreren Klassen für die Modelltrainingskonfiguration
  - Angabe einer Knotenregressionsaufgabe für die Modelltrainingskonfiguration
  - Angabe einer Kantenklassifizierungsaufgabe für die Modelltrainingskonfiguration
  - Angabe einer Klassifizierungsaufgabe für Kanten mit mehreren Klassen für die Modelltrainingskonfiguration
  - Angabe einer Kantenregression für die Modelltrainingskonfiguration
  - Angabe einer Linkvorhersageaufgabe für die Modelltrainingskonfiguration
  - Angabe eines numerischen Bucket-Features
  - Angabe eines Word2Vec-Features
  - Angabe eines FastText-Features
  - Angabe eines Sentence BERT-Features
  - Angabe eines TF-IDF-Features
  - Angabe eines datetime-Features
  - Angabe eines category-Features
  - Angabe eines numerical-Features

  - Angabe eines auto-Features
- RDF-Beispiele mit additionalParams
  - Angabe einer Standard-Aufteilungsrate für die Modelltrainingskonfiguration
  - Angabe einer Knotenklassifizierungsaufgabe für die Modelltrainingskonfiguration
  - Angabe einer Knotenregressionsaufgabe für die Modelltrainingskonfiguration
  - Angabe einer Linkvorhersageaufgabe für bestimmte Kanten
  - Angabe einer Linkvorhersageaufgabe für alle Kanten

Beispiele für Eigenschaftsdiagramme mit AdditionalParams

Angabe einer Standard-Aufteilungsrate für die Modelltrainingskonfiguration

Im folgenden Beispiel legt der Parameter `split_rate` die Standard-Aufteilungsrate für das Modelltraining fest. Wenn keine Standard-Aufteilungsrate angegeben ist, verwendet das Training den Wert [0,9, 0,1, 0,0]. Sie können den Standardwert pro Ziel überschreiben, indem Sie für jedes Ziel eine `split_rate` angeben.

Im folgenden Beispiel gibt das Feld `default split_rate` an, dass die Aufteilungsrate `[0.7,0.1,0.2]` verwendet werden sollte, wenn sie nicht pro Ziel überschrieben wird:

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "split_rate": [0.7,0.1,0.2],
    "targets": [
      (...)
    ],
    "features": [
      (...)
    ]
  }
}
```

Angabe einer Knotenklassifizierungsaufgabe für die Modelltrainingskonfiguration

Zur Angabe der Knoteneigenschaft, die bezeichnete Beispiele für Trainingszwecke enthält, fügen Sie dem Array `targets` ein Element für die Knotenklassifizierung mit `"type" : "classification"`

hinzu. Fügen Sie das Feld `split_rate` hinzu, wenn Sie die Standard-Aufteilungsrate überschreiben möchten.

Im folgenden Beispiel zeigt das Ziel `node` an, dass die Eigenschaft `genre` jedes `Movie`-Knotens als Klassenbezeichnung eines Knotens behandelt werden soll. Der Wert `split_rate` überschreibt die Standard-Aufteilungsrate:

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      {
        "node": "Movie",
        "property": "genre",
        "type": "classification",
        "split_rate": [0.7,0.1,0.2]
      }
    ],
    "features": [
      (...)
    ]
  }
}
```

Angabe einer Klassifizierungsaufgabe für Knoten mit mehreren Klassen für die Modelltrainingskonfiguration

Zur Angabe der Knoteneigenschaft, die mehrere bezeichnete Beispiele für Trainingszwecke enthält, fügen Sie mit `"type" : "classification"` dem Ziel-Array ein Knotenklassifikationselement und einen `separator` hinzu, um das Zeichen anzugeben, das zur Aufteilung eines Zieleigenschaftswerts in mehrere kategoriale Werte verwendet werden kann. Fügen Sie das Feld `split_rate` hinzu, wenn Sie die Standard-Aufteilungsrate überschreiben möchten.

Im folgenden Beispiel zeigt das Ziel `node` an, dass die Eigenschaft `genre` jedes `Movie`-Knotens als Klassenbezeichnung eines Knotens behandelt werden soll. Das Feld `separator` zeigt an, dass jede Genre-Eigenschaft mehrere durch Semikolon getrennte Werte enthält:

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
```

```
      {
        "node": "Movie",
        "property": "genre",
        "type": "classification",
        "separator": ";"
      }
    ],
    "features": [
      (...)
    ]
  }
}
```

Angabe einer Knotenregressionsaufgabe für die Modelltrainingskonfiguration

Zur Angabe der Knoteneigenschaft, die bezeichnete Regressionen für Trainingszwecke enthält, fügen Sie mit `"type" : "regression"` dem Ziel-Array ein Element für die Knotenregression hinzu. Fügen Sie das Feld split_rate hinzu, wenn Sie die Standard-Aufteilungsrate überschreiben möchten.

Im folgenden Beispiel zeigt das Ziel `node` an, dass die Eigenschaft `rating` jedes `Movie`-Knotens als Regressionsbezeichnung eines Knotens behandelt werden soll.

```
    "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      {
        "node": "Movie",
        "property": "rating",
        "type" : "regression",
        "split_rate": [0.7,0.1,0.2]
      }
    ],
    "features": [
      ...
    ]
  }
}
```

Angabe einer Kantenklassifizierungsaufgabe für die Modelltrainingskonfiguration

Zur Angabe der Kanteneigenschaft, die bezeichnete Beispiele für Trainingszwecke enthält, fügen Sie mit `"type" : "regression"` dem Array `targets` ein Kantenelement hinzu. Fügen Sie das Feld split_rate hinzu, wenn Sie die Standard-Aufteilungsrate überschreiben möchten.

Das folgende Ziel `edge` zeigt an, dass die Eigenschaft `metAtLocation` jeder `knows`-Kante als Kantenklassenbezeichnung behandelt werden soll:

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      {
        "edge": ["Person", "knows", "Person"],
        "property": "metAtLocation",
        "type": "classification"
      }
    ],
    "features": [
      (...)
    ]
  }
}
```

Angabe einer Klassifizierungsaufgabe für Kanten mit mehreren Klassen für die Modelltrainingskonfiguration

Zur Angabe der Kanteneigenschaft, die mehrere bezeichnete Beispiele für Trainingszwecke enthält, fügen Sie mit `"type" : "classification"` dem `targets`-Array ein Kantenelement und das Feld `separator` hinzu, um das Zeichen anzugeben, das zur Aufteilung eines Zieleigenschaftswerts in mehrere kategoriale Werte verwendet werden kann. Fügen Sie das Feld `split_rate` hinzu, wenn Sie die Standard-Aufteilungsrate überschreiben möchten.

Das folgende Ziel `edge` zeigt an, dass die Eigenschaft `sentiment` jeder `repliedTo`-Kante als Kantenklassenbezeichnung behandelt werden soll: Das Separatorfeld zeigt an, dass jede Stimmungseigenschaft mehrere durch Komma getrennte Werte enthält:

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
```

```
    "targets": [
      {
        "edge": ["Person", "repliedTo", "Message"],
        "property": "sentiment",
        "type": "classification",
        "separator": ","
      }
    ],
    "features": [
      (...)
    ]
  }
}
```

Angabe einer Kantenregression für die Modelltrainingskonfiguration

Zur Angabe der Kanteneigenschaft, die bezeichnete Regressionsbeispiele für Trainingszwecke enthält, fügen Sie mit `"type" : "regression"` dem Array `targets` das Element `edge` hinzu. Fügen Sie das Feld `split_rate` hinzu, wenn Sie die Standard-Aufteilungsrate überschreiben möchten.

Das folgende Ziel `edge` zeigt an, dass die Eigenschaft `rating` jeder `reviewed`-Kante als Kantenregression behandelt werden soll:

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      {
        "edge": ["Person", "reviewed", "Movie"],
        "property": "rating",
        "type" : "regression"
      }
    ],
    "features": [
      (...)
    ]
  }
}
```

Angabe einer Linkvorhersageaufgabe für die Modelltrainingskonfiguration

Zur Angabe der Kanten, die für Linkvorhersagetrainings verwendet werden sollen, fügen Sie mit `"type" : "link_prediction"` dem Ziel-Array ein Kantenelement hinzu. Fügen Sie das Feld `split_rate` hinzu, wenn Sie die Standard-Aufteilungsrate überschreiben möchten.

Das folgende Ziel `edge` zeigt an, dass `cites`-Kanten für die Linkvorhersage verwendet werden sollen:

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      {
        "edge": ["Article", "cites", "Article"],
        "type" : "link_prediction"
      }
    ],
    "features": [
      (...)
    ]
  }
}
```

Angabe eines numerischen Bucket-Features

Sie können ein numerisches Daten-Feature für eine Knoteneigenschaft angeben, indem Sie `"type": "bucket_numerical"` zum `features`-Array hinzufügen.

Das folgende Feature `node` zeigt an, dass die Eigenschaft `age` jedes `Person`-Knotens als numerisches Bucket-Feature behandelt werden soll:

```
  "additionalParams": {
  "neptune_ml": {
    "targets": [
      ...
    ],
    "features": [
      {
        "node": "Person",
        "property": "age",
        "type": "bucket_numerical",
```

```
        "range": [1, 100],
        "bucket_cnt": 5,
        "slide_window_size": 3,
        "imputer": "median"
      }
    ]
  }
}
```

Angabe eines **`Word2Vec`**-Features

Sie können ein `Word2Vec`-Feature für eine Knoteneigenschaft angeben, indem Sie `"type": "text_word2vec"` zum `features`-Array hinzufügen.

Das folgende Feature node zeigt an, dass die Eigenschaft `description` jedes `Movie`-Knotens als `Word2Vec`-Feature behandelt werden soll:

```
"additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      ...
    ],
    "features": [
      {
        "node": "Movie",
        "property": "description",
        "type": "text_word2vec",
        "language": "en_core_web_lg"
      }
    ]
  }
}
```

Angabe eines **`FastText`**-Features

Sie können ein `FastText`-Feature für eine Knoteneigenschaft angeben, indem Sie `"type": "text_fasttext"` zum `features`-Array hinzufügen. Das Feld `language` ist ein Pflichtfeld und muss einen der folgenden Sprachcodes angeben:

- `en` (Englisch)
- `zh` (Chinesisch)

- `hi` (Hindi)
- `es` (Spanisch)
- `fr` (Französisch)

Beachten Sie, dass die `text_fasttext`-Kodierung nicht mehr als eine Sprache gleichzeitig in einem Feature verarbeiten kann.

Das folgende Feature `node` zeigt an, dass die Eigenschaft `description` (Französisch) jedes `Movie`-Knotens als `FastText`-Feature behandelt werden soll:

```
"additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      ...
    ],
    "features": [
      {
        "node": "Movie",
        "property": "description",
        "type": "text_fasttext",
        "language": "fr",
        "max_length": 1024
      }
    ]
  }
}
```

Angabe eines **`Sentence BERT`**-Features

Sie können ein `Sentence BERT`-Feature für eine Knoteneigenschaft angeben, indem Sie `"type": "text_sbert"` zum `features`-Array hinzufügen. Sie müssen die Sprache nicht angeben, da die Methode Text-Features automatisch mithilfe eines mehrsprachigen Sprachmodells kodiert.

Das folgende Feature `node` zeigt an, dass die Eigenschaft `description` jedes `Movie`-Knotens als `Sentence BERT`-Feature behandelt werden soll:

```
"additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
```

```
      ...
    ],
    "features": [
      {
        "node": "Movie",
        "property": "description",
        "type": "text_sbert128",
      }
    ]
  }
}
```

Angabe eines **`TF-IDF`**-Features

Sie können ein `TF-IDF`-Feature für eine Knoteneigenschaft angeben, indem Sie `"type": "text_tfidf"` zum `features`-Array hinzufügen.

Das folgende Feature `node` zeigt an, dass die Eigenschaft `bio` jedes `Person`-Knotens als `TF-IDF`-Feature behandelt werden soll:

```
  "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
      }
    ]
  }
}
```

Angabe eines **`datetime`**-Features

Beim Exportvorgang werden automatisch `datetime`-Features für Datumseigenschaften inferiert. Wenn Sie jedoch die `datetime_parts` einschränken möchten, die für ein `datetime`-Feature

verwendet werden, oder eine Feature-Spezifikation überschreiben möchten, sodass eine Eigenschaft, die normalerweise als `auto`-Feature behandelt würde, explizit als `datetime`-Feature behandelt wird, fügen Sie dem Feature-Array `"type": "datetime"` hinzu.

Das folgende Feature `node` zeigt an, dass die Eigenschaft `createdAt` jedes `Post`-Knotens als `datetime`-Feature behandelt werden soll:

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      ...
    ],
    "features": [
      {
        "node": "Post",
        "property": "createdAt",
        "type": "datetime",
        "datetime_parts": ["month", "weekday", "hour"]
      }
    ]
  }
}
```

Angabe eines **`category`**-Features

Der Exportvorgang inferiert automatisch `auto`-Features für Zeichenfolgeneigenschaften und numerische Eigenschaften, die mehrere Werte enthalten. Für numerische Eigenschaften, die Einzelwerte enthalten, werden `numerical`-Features inferiert. Für Datumseigenschaften werden `datetime`-Features inferiert.

Wenn Sie eine Feature-Spezifikation überschreiben möchten, damit eine Eigenschaft als kategoriales Feature behandelt wird, fügen Sie dem Feature-Array `"type": "category"` hinzu. Wenn die Eigenschaft mehrere Werte enthält, fügen Sie das Feld `separator` hinzu. Beispiel:

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      ...
    ],
    "features": [
```

```
      {
        "node": "Post",
        "property": "tag",
        "type": "category",
        "separator": "|"
      }
    ]
  }
}
```

Angabe eines **`numerical`**-Features

Der Exportvorgang inferiert automatisch `auto`-Features für Zeichenfolgeneigenschaften und numerische Eigenschaften, die mehrere Werte enthalten. Für numerische Eigenschaften, die Einzelwerte enthalten, werden `numerical`-Features inferiert. Für Datumseigenschaften werden `datetime`-Features inferiert.

Wenn Sie eine Feature-Spezifikation überschreiben möchten, damit eine Eigenschaft als `numerical`-Feature behandelt wird, fügen Sie dem Feature-Array `"type": "numerical"` hinzu. Wenn die Eigenschaft mehrere Werte enthält, fügen Sie das Feld `separator` hinzu. Beispiel:

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      ...
    ],
    "features": [
      {
        "node": "Recording",
        "property": "duration",
        "type": "numerical",
        "separator": ","
      }
    ]
  }
}
```

Angabe eines **`auto`**-Features

Der Exportvorgang inferiert automatisch `auto`-Features für Zeichenfolgeneigenschaften und numerische Eigenschaften, die mehrere Werte enthalten. Für numerische Eigenschaften, die

Einzelwerte enthalten, werden `numerical`-Features inferiert. Für Datumseigenschaften werden `datetime`-Features inferiert.

Wenn Sie eine Feature-Spezifikation überschreiben möchten, damit eine Eigenschaft als `auto`-Feature behandelt wird, fügen Sie dem Feature-Array `"type": "auto"` hinzu. Wenn die Eigenschaft mehrere Werte enthält, fügen Sie das Feld `separator` hinzu. Beispiel:

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      ...
    ],
    "features": [
      {
        "node": "User",
        "property": "role",
        "type": "auto",
        "separator": ","
      }
    ]
  }
}
```

RDF-Beispiele mit **`additionalParams`**

Angabe einer Standard-Aufteilungsrate für die Modelltrainingskonfiguration

Im folgenden Beispiel legt der Parameter `split_rate` die Standard-Aufteilungsrate für das Modelltraining fest. Wenn keine Standard-Aufteilungsrate angegeben ist, verwendet das Training den Wert [0,9, 0,1, 0,0]. Sie können den Standardwert pro Ziel überschreiben, indem Sie für jedes Ziel eine `split_rate` angeben.

Im folgenden Beispiel gibt das Feld `default split_rate` an, dass die Aufteilungsrate `[0.7,0.1,0.2]` verwendet werden sollte, wenn sie nicht pro Ziel überschrieben wird:

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "split_rate": [0.7,0.1,0.2],
    "targets": [
```

```
      (...)
    ]
  }
}
```

Angabe einer Knotenklassifizierungsaufgabe für die Modelltrainingskonfiguration

Zur Angabe der Knoteneigenschaft, die bezeichnete Beispiele für Trainingszwecke enthält, fügen Sie dem Array `targets` ein Element für die Knotenklassifizierung mit `"type" : "classification"` hinzu. Fügen Sie ein Knotenfeld hinzu, um den Knotentyp der Zielknoten anzugeben. Fügen Sie ein `predicate`-Feld hinzu, um die Literaldaten zu definieren, die als Zielknoten-Feature des Zielknotens verwendet werden. Fügen Sie das Feld `split_rate` hinzu, wenn Sie die Standard-Aufteilungsrate überschreiben möchten.

Im folgenden Beispiel zeigt das Ziel `node` an, dass die Eigenschaft `genre` jedes `Movie`-Knotens als Klassenbezeichnung eines Knotens behandelt werden soll. Der Wert `split_rate` überschreibt die Standard-Aufteilungsrate:

```
  "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]
      }
    ]
  }
}
```

Angabe einer Knotenregressionsaufgabe für die Modelltrainingskonfiguration

Zur Angabe der Knoteneigenschaft, die bezeichnete Regressionen für Trainingszwecke enthält, fügen Sie mit `"type" : "regression"` dem Ziel-Array ein Element für die Knotenregression hinzu. Fügen Sie ein `node`-Feld hinzu, um den Knotentyp der Zielknoten anzugeben. Fügen Sie ein `predicate`-Feld hinzu, um die Literaldaten zu definieren, die als Zielknoten-Feature des Zielknotens verwendet werden. Fügen Sie das Feld `split_rate` hinzu, wenn Sie die Standard-Aufteilungsrate überschreiben möchten.

Im folgenden Beispiel zeigt das Ziel `node` an, dass die Eigenschaft `rating` jedes `Movie`-Knotens als Regressionsbezeichnung eines Knotens behandelt werden soll.

```
    "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]
      }
    ]
  }
}
```

Angabe einer Linkvorhersageaufgabe für bestimmte Kanten

Zur Angabe der Kanten, die für Linkvorhersagetrainings verwendet werden sollen, fügen Sie mit `"type" : "link_prediction"` dem Ziel-Array ein Kantenelement hinzu. Fügen Sie die Felder `subject`, `predicate` und `object` hinzu, um den Kantentyp anzugeben. Fügen Sie das Feld `split_rate` hinzu, wenn Sie die Standard-Aufteilungsrate überschreiben möchten.

Das folgende `edge`-Ziel zeigt an, dass `directed`-Kanten, die `Directors` mit `Movies` verbinden, für die Linkvorhersage verwendet werden sollen:

```
  "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"
      }
    ]
  }
}
```

Angabe einer Linkvorhersageaufgabe für alle Kanten

Um anzugeben, dass alle Kanten für Linkvorhersagetrainings verwendet werden sollen, fügen Sie mit `"type" : "link_prediction"` dem Ziel-Array ein `edge`-Element hinzu. Fügen Sie keine Felder `subject`, `predicate` oder `object` hinzu. Fügen Sie das Feld `split_rate` hinzu, wenn Sie die Standard-Aufteilungsrate überschreiben möchten.

```
  "additionalParams": {
  "neptune_ml": {
    "version": "v2.0",
    "targets": [
      {
        "type" : "link_prediction"
      }
    ]
  }
}
```

Verarbeitung der aus Neptune zu Trainingszwecken exportierten Diagrammdaten

Im Datenverarbeitungsschritt werden anhand der durch den Exportvorgang erstellten Neptune-Diagrammdaten die Informationen erstellt, die von der Deep Graph Library (DGL) beim Training verwendet werden. Dies umfasst die verschiedene Datenzuordnungen und -transformationen:

- Analyse von Knoten und Kanten, um die von der DGL benötigten Diagramm- und ID-Zuordnungsdateien zu erstellen.
- Konvertierung von Knoten- und Kanteneigenschaften in die von der DGL benötigten Knoten- und Kanten-Features.
- Aufteilung der Daten in Trainings-, Validierungs- und Testsätze.

Verwalten des Datenverarbeitungsschritts für Neptune ML

Nachdem Sie die Daten aus Neptune exportiert haben, die Sie für das Modelltraining verwenden möchten, können Sie einen Datenverarbeitungsjob mit einem Befehl wie dem folgenden starten:

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"
```

Weitere Informationen finden Sie unter start-ml-data-processing-job in der Befehlsreferenz. 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.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**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

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"
      }'
```

Details zur Verwendung dieses Befehls werden in Der Datenverarbeitungsbefehl beschrieben. Dort finden Sie auch Informationen dazu, wie Sie ausgeführte Aufträge abrufen und beenden und alle ausgeführten Aufträge auflisten.

Verarbeiten aktualisierter Diagrammdaten für Neptune ML

Sie können der API auch eine `previousDataProcessingJobId` bereitstellen, damit der neue Datenverarbeitungsauftrag dieselbe Verarbeitungsmethode wie ein vorheriger Auftrag verwendet. Dies ist erforderlich, wenn Sie Vorhersagen für aktualisierte Diagrammdaten in Neptune erhalten möchten, indem Sie entweder das alte Modell mit den neuen Daten trainieren oder die Modellartefakte anhand der neuen Daten neu berechnen.

Sie tun dies, indem Sie einen Befehl wie diesen verwenden:

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)"
```

Weitere Informationen finden Sie unter start-ml-data-processing-job in der AWS CLI Befehlsreferenz.

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**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

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)"
     }'
```

Legen Sie den Wert des Parameters `previousDataProcessingJobId` auf die Auftrags-ID des vorherigen Datenverarbeitungsauftrags für das trainierte Modell fest.

> **Note**
>
> Das Löschen von Knoten im aktualisierten Diagramm wird zurzeit nicht unterstützt. Wenn in einem aktualisierten Diagramm Knoten entfernt wurden, müssen Sie einen völlig neuen Datenverarbeitungsauftrag starten, statt `previousDataProcessingJobId` zu verwenden.

Feature-Kodierung in Neptune ML

Eigenschaftswerte haben verschiedene Formate und Datentypen. Um eine gute Machine-Learning-Leistung zu erzielen, müssen diese Werte in numerische Kodierungen konvertiert werden, so genannte features.

Neptune ML extrahiert und kodiert Features im Rahmen der Schritte data-export und data-processing mit den hier beschrieben Feature-Kodierungs-Techniken.

> **Note**
>
> Wenn Sie eine eigene Feature-Kodierung in einer benutzerdefinierten Modellimplementierung implementieren möchten, können Sie die automatische Feature-Kodierung in der Datenvorverarbeitungsphase deaktivieren, indem Sie als Feature-Kodierungstyp none auswählen. In diesem Fall findet für diese Knoten- oder Kanteneigenschaft keine Feature-Kodierung statt. Stattdessen werden die rohen Eigenschaftswerte analysiert und in einem Verzeichnis gespeichert. Während der Datenvorverarbeitung wird weiter das DGL-Diagramm aus dem exportierten Datensatz erstellt, aber das konstruierte DGL-Diagramm verfügt nicht über keine vorverarbeiteten Features für das Training.
> Sie sollten diese Option nur verwenden, wenn Sie eine benutzerdefinierte Feature-Kodierung als Teil eines benutzerdefinierten Modelltrainings ausführen möchten. Details dazu finden Sie unter Benutzerdefinierte Modelle in Neptune ML.

Kategorische Features in Neptune ML

Eine Eigenschaft, die einen oder mehrere unterschiedliche Werte aus einer festen Liste möglicher Werte annehmen kann, ist ein kategorisches Feature. In Neptune ML werden kategorische Features mit One-Hot-Kodierung kodiert. Das folgende Beispiel zeigt, wie die Eigenschaftsnamen verschiedener Lebensmittel entsprechend ihrer Kategorie mit One-Hot-Kodierung kodiert werden:

```
 Food        Veg.   Meat   Fruit    Encoding
---------    ----   ----   -----    --------
 Apple         0      0      1         001
 Chicken       0      1      0         010
 Broccoli      1      0      0         100
```

> **Note**
>
> Die maximale Anzahl von Kategorien in einem kategorischen Feature ist 100. Wenn eine Eigenschaft mehr als 100 Wertkategorien besitzt, werden nur die 99 häufigsten Kategorien in verschiedenen Kategorien platziert. Der Rest wird in einer speziellen Kategorie mit dem Namen `OTHER` platziert.

Numerische Features in Neptune ML

Eine Eigenschaft, deren Werte reale Zahlen sind, kann in Neptune ML als numerisches Feature kodiert werden. Numerische Features werden mit Gleitkommazahlen kodiert.

Sie können eine Methode für die Datennormalisierung angeben, die beim Kodieren numerischer Features verwendet werden soll, z. B.: `"norm": "normalization technique"`. Die folgenden Normalisierungstechniken werden unterstützt:

- "none" – Normalisiert die numerischen Werte während der Kodierung nicht.
- "min-max" – Normalisiert jeden Wert, indem der Mindestwert subtrahiert wird und der Wert dann durch die Differenz zwischen Maximalwert und Mindestwert dividiert wird.
- "standard" – Normalisiert jeden Wert, indem er durch die Summe aller Werte dividiert wird.

Bucket-numerische Features in Neptune ML

Anstatt eine numerische Eigenschaft mit rohen Zahlen darzustellen, können Sie numerische Werte zu Kategorien kondensieren. Sie könnten beispielsweise das Alter von Personen in Kategorien wie Kinder (0–20), junge Erwachsene (20–40), Personen mittleren Alters (40–60) und ältere Menschen (ab 60) unterteilen. Mithilfe dieser numerischen Buckets würden Sie eine numerische Eigenschaft in eine Art kategorisches Feature transformieren.

In Neptune ML können Sie die Kodierung einer numerischen Eigenschaft als Bucket-numerisches Feature veranlassen. Sie müssen zwei Dinge angeben:

- Ein numerischer Bereich im Format `"range": [a, b]`, wobei a und b Ganzzahlen sind.
- Eine Bucket-Zahl im Format `"bucket_cnt": c`, wobei c die Anzahl der Buckets ist, ebenfalls eine Ganzzahl.

Neptune ML berechnet dann die Größe jedes Buckets als `( b - a ) / c` und kodiert jeden numerischen Wert als Nummer des Buckets, zu dem er gehört. Jeder Wert kleiner als a wird als zum ersten Bucket gehörend betrachtet. Jeder Wert größer als b wird als zum letzten Bucket gehörend betrachtet.

Sie können optional auch festlegen, dass numerische Werte zu mehr als einem Bucket gehören, indem Sie eine Größe für das Gleitfenster angeben, z. B.: `"slide_window_size": s` . Dabei ist s eine Zahl. Neptune ML transformiert anschließend jeden numerischen Wert v der Eigenschaft in einen Bereich von `v - s/2` bis `v + s/2` und weist den Wert v jedem Bucket zu, den der Bereich abdeckt.

Schließlich können Sie optional auch eine Möglichkeit zum Ausfüllen fehlender Werte für numerische Features und Bucket-numerische Features bereitstellen. Hierzu verwenden Sie `"imputer": "imputation technique "`, wobei die Imputationstechnik entweder `"mean"`, `"median"` oder `"most-frequent"` ist. Wenn Sie keinen Imputer angeben, kann ein fehlender Wert dazu führen, dass die Verarbeitung angehalten wird.

Text-Feature-Kodierung in Neptune ML

Für Freiformtext kann Neptune ML mehrere verschiedene Modelle verwenden, um die Sequenz von Token in einer Eigenschaftswert-Zeichenfolge in einen Realwertvektor fester Größe zu konvertieren:

- `text_fasttext` – Verwendet fastText-Kodierung. Dies ist die empfohlene Kodierung für Features, die eine und nur eine der fünf Sprachen verwenden, die fastText unterstützt.
- `text_sbert` – Verwendet die Kodierungsmodelle Sentence BERT (SBERT). Dies ist die empfohlene Kodierung für Text, den `text_fasttext` nicht unterstützt.
- `text_word2vec` – Verwendet die Word2Vec-Algorithmen (ursprünglich von Google veröffentlicht), um Text zu kodieren. Word2Vec unterstützt nur Englisch.
- `text_tfidf` – Verwendet den Vektorisierer term frequency–inverse document frequency (TF-IDF) für die Textkodierung. Die TF-IDF-Kodierung unterstützt statistische Features, die von den anderen Kodierungen nicht unterstützt werden.

fastText-Kodierung von Texteigenschaftswerten in Neptune ML

Neptune ML kann die fastText-Modelle verwenden, um Texteigenschaftswerte in Realwertvektoren fester Größe zu konvertieren. Dies ist die empfohlene Kodierungsmethode für Texteigenschaftswerte in einer der fünf Sprachen, die fastText unterstützt:

- `en` (Englisch)
- `zh` (Chinesisch)
- `hi` (Hindi)
- `es` (Spanisch)
- `fr` (Französisch)

Beachten Sie, dass fastText keine Sätze verarbeiten kann, die Wörter in mehr als einer Sprache enthalten.

Die Methode `text_fasttext` kann optional das Feld `max_length` verwenden, das die maximale Anzahl von Token in einem Texteigenschaftswert angibt, die kodiert werden sollen. Anschließend wird die Zeichenfolge abgeschnitten. Dies kann die Leistung verbessern, wenn Texteigenschaftswerte lange Zeichenfolgen enthalten, da fastText alle Token unabhängig von der Zeichenfolgenlänge kodiert, wenn `max_length` nicht angegeben ist.

Dieses Beispiel gibt an, dass französische Filmtitel mit fastText kodiert werden:

```
{
    "file_name" : "nodes/movie.csv",
    "separator" : ",",
    "node" : ["~id", "movie"],
    "features" : [
      {
        "feature": ["title", "title", "text_fasttext"],
        "language": "fr",
        "max_length": 1024
      }
    ]
  }
```

Sentence BERT (SBERT)-Satzkodierung von Text-Features in Neptune ML

Neptune ML kann die Sequenz von Token in einem Zeichenfolgen-Eigenschaftswert mithilfe von Sentence BERT (SBERT)-Modellen in einen Realwertvektor fester Größe konvertieren. Neptune unterstützt zwei SBERT-Methoden: `text_sbert128` (die Standardmethode, wenn Sie nur `text_sbert` angeben) und `text_sbert512`. Der Unterschied zwischen den beiden Methoden ist die maximale Länge einer kodierten Texteigenschaftswert-Zeichenfolge. Bei der `text_sbert128`-Kodierung werden Textzeichenfolgen nach der Kodierung von 128 Token abgeschnitten, während

`text_sbert512` Textzeichenfolgen nach der Kodierung von 512 Token abschneidet. Daher erfordert `text_sbert512` mehr Verarbeitungszeit als `text_sbert128`. Beide Methoden sind langsamer als `text_fasttext`.

Die SBERT-Kodierung ist mehrsprachig. Daher müssen Sie keine Sprache für den Eigenschaftswerttext angeben, den Sie kodieren. SBERT unterstützt zahlreiche Sprachen und kann einen Satz kodieren, der mehr als eine Sprache enthält. Wenn Sie Eigenschaftswerte kodieren, die Text in einer oder mehreren Sprachen enthalten, die fastText nicht unterstützt, ist SBERT die empfohlene Kodierungsmethode.

Das folgende Beispiel gibt an, dass Filmtitel bis maximal 128 Token als SBERT kodiert werden:

```
{
    "file_name" : "nodes/movie.csv",
    "separator" : ",",
    "node" : ["~id", "movie"],
    "features" : [
      { "feature": ["title", "title", "text_sbert128"] }
    ]
  }
```

Word2Vec-Kodierung von Text-Features in Neptune ML

Neptune ML kann Zeichenfolgen-Eigenschaftswerte als Word2Vec-Feature kodieren. (Die Word2Vec-Algorithmen wurden ursprünglich von Google veröffentlicht.) Die Methode `text_word2vec` kodiert die Token in einer Zeichenfolge als dichten Vektor unter Verwendung eines von SpaCy trainierten Modells. Sie unterstützt nur die englische Sprache (unter Verwendung des Modells en_core_web_lg).

Das folgende Beispiel gibt an, dass Filmtitel mit Word2Vec kodiert werden:

```
{
    "file_name" : "nodes/movie.csv",
    "separator" : ",",
    "node" : ["~id", "movie"],
    "features" : [
      {
        "feature": ["title", "title", "text_word2vec"],
        "language": "en_core_web_lg"
      }
    ]
  }
```

Beachten Sie, dass das Sprachfeld optional ist, da das englische `en_core_web_lg`-Modell das einzige Modell ist, das Neptune unterstützt.

TF-IDF-Kodierung von Text-Features in Neptune ML

Neptune ML kann Texteigenschaftswerte als `text_tfidf`-Features kodieren. Diese Kodierung konvertiert die Reihenfolge der Wörter im Text mittels des Vektorisierers term frequency–inverse document frequency (TF-IDF) in einen numerischen Vektor, gefolgt von einer Operation zur Reduzierung der Dimensionalität.

TF-IDF (term frequency – inverse document frequency) ist ein numerischer Wert, der misst, wie wichtig ein Wort in einem Dokumentensatz ist. Er wird berechnet, indem die Häufigkeit, mit der ein Wort in einem bestimmten Eigenschaftswert vorkommt, durch die Gesamtzahl der Eigenschaftswerte dividiert wird, in denen es vorkommt.

Wenn beispielsweise das Wort „Kiss“ zweimal in einem bestimmten Filmtitel vorkommt (z. B. „Kiss Kiss Bang Bang“) und „Kiss“ im Titel von insgesamt 4 Filmen vorkommt, dann wäre der TF-IDF-Wert für „Kiss“ im Titel „Kiss Kiss Bang Bang“ `2 / 4` .

Der Vektor, der ursprünglich erstellt wird, hat d Dimensionen, wobei d die Anzahl der eindeutigen Begriffe in allen Eigenschaftswerten dieses Typs ist. Die Operation zur Reduzierung der Dimensionalität verwendet eine randomisierte Projektion mit geringer Dichte, um diese Zahl auf maximal 100 zu reduzieren. Anschließend wird das Vokabular eines Diagramms generiert, indem alle enthaltenen `text_tfidf`-Features zusammengeführt werden.

Sie können den TF-IDF-Vektorisierer auf verschiedene Arten steuern:

- **`max_features`** – Mithilfe des Parameters `max_features` können Sie die Anzahl der Begriffe in `text_tfidf`-Features auf die häufigsten einschränken. Wenn Sie beispielsweise `max_features` auf 100 festlegen, sind nur die 100 am häufigsten verwendeten Begriffe enthalten. Wenn Sie `max_features` nicht explizit festlegen, ist der Standardwert 5 000.
- **`min_df`** – Mithilfe des Parameters `min_df` können Sie die Anzahl der Begriffe in `text_tfidf`-Features auf Features einschränken, die mindestens mit einer angegebenen Häufigkeit im Dokumentsatz vorkommen. Wenn Sie beispielsweise `min_df` auf 5 festlegen, werden nur Begriffe verwendet, die in mindestens 5 verschiedenen Eigenschaftswerten vorkommen. Wenn Sie `min_df` nicht explizit festlegen, ist der Standardwert 2.
- **`ngram_range`** – Der Parameter `ngram_range` bestimmt, welche Wortkombinationen als Begriffe behandelt werden. Wenn Sie beispielsweise `ngram_range` auf `[2, 4]` festlegen, würden die folgenden 6 Begriffe im Titel „Kiss Kiss Bang Bang“ vorkommen:

- Begriffe mit 2 Wörtern: „Kiss Kiss“, „Kiss Bang“ und „Bang Bang“.
- Begriffe mit 3 Wörtern: „Kiss Kiss Bang“ und „Kiss Bang Bang“.
- Begriffe mit 4 Wörtern: „Kiss Kiss Bang Bang“.

Die Standardeinstellung für `ngram_range` ist `[1, 1]`.

Datetime-Features in Neptune ML

Neptune ML kann Teile von `datetime`-Eigenschaftswerten in kategorische Features konvertieren, indem sie als One-Hot-Arrays kodiert werden. Verwenden Sie den Parameter `datetime_parts`, um einen oder mehrere der folgenden Teile für die Kodierung anzugeben: `["year", "month", "weekday", "hour"]`. Wenn Sie `datetime_parts` nicht festlegen, werden standardmäßig alle vier Teile kodiert.

Wenn der Bereich der datetime-Werte beispielsweise die Jahre 2010 bis 2012 umfasst, sind die vier Teile des datetime-Eintrags `2011-04-22 01:16:34` wie folgt:

- year – `[0, 1, 0]`.

  Da der Zeitraum nur 3 Jahre umfasst (2010, 2011 und 2012), hat das One-Hot-Array drei Einträge, einen für jedes Jahr.
- month – `[0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0]`.

  Hier enthält das One-Hot-Array einen Eintrag für jeden Monat des Jahres.
- weekday – `[0, 0, 0, 0, 1, 0, 0]`.

  Nach dem ISO-8601-Standard ist Montag der erste Tag der Woche. Da der 22. April 2011 ein Freitag war, ist das entsprechende One-Hot-Wochentag-Array an der fünften Position aktiviert.
- 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]`.

  Die Stunde 01:00 Uhr ist in einem One-Hot-Array mit 24 Elementen festgelegt.

Tag des Monats, Minute und Sekunde sind nicht kategorisch kodiert.

Wenn der betreffende `datetime`-Bereich nur Daten innerhalb eines einzigen Jahres umfasst, wird kein `year`-Array kodiert.

Sie können eine Imputationsstrategie angeben, um fehlende `datetime`-Werte aufzufüllen, indem Sie den Parameter `imputer` und eine der für numerische Features verfügbaren Strategien verwenden.

Auto-Feature-Kodierung in Neptune ML

Anstatt die Feature-Kodierungsmethoden manuell anzugeben, die für die Eigenschaften im Diagramm verwendet werden sollen, können Sie `auto` als Feature-Kodierungsmethode festlegen. Neptune ML versucht dann, anhand des zugrunde liegenden Datentyps die beste Feature-Kodierung für jede Eigenschaft zu inferieren.

Dies sind einige der Heuristiken, die Neptune ML bei der Auswahl der geeigneten Feature-Kodierungen verwendet:

- Wenn die Eigenschaft nur numerische Werte besitzt und in numerische Datentypen konvertiert werden kann, kodiert Neptune ML sie im Allgemeinen als numerischen Wert. Wenn jedoch die Anzahl der eindeutigen Werte für die Eigenschaft weniger als 10 % der Gesamtzahl der Werte ist und die Kardinalität dieser eindeutigen Werte kleiner als 100 ist, verwendet Neptune ML eine kategorische Kodierung.
- Wenn die Eigenschaftswerte in einen `datetime`-Typ konvertiert werden können, kodiert Neptune ML sie als `datetime`-Feature.
- Wenn die Eigenschaftswerte in boolesche Werte (1/0 oder True/False) transformiert werden können, verwendet Neptune ML die Kategoriekodierung.
- Wenn es sich bei der Eigenschaft um eine Zeichenfolge handelt, in der mehr als 10 % ihrer Werte eindeutig sind und die durchschnittliche Anzahl von Token pro Wert größer oder gleich 3 ist, inferiert Neptune ML den Eigenschaftstyp als Text und erkennt automatisch die verwendete Sprache. Wenn es sich bei der erkannten Sprache um eine der von fastText unterstützten Sprachen handelt (Englisch, Chinesisch, Hindi, Spanisch und Französisch), verwendet Neptune ML `text_fasttext` zur Kodierung des Texts. Andernfalls verwendet Neptune ML `text_sbert`.
- Wenn die Eigenschaft eine nicht als Text-Feature klassifizierte Zeichenfolge ist, nimmt Neptune ML an, dass es sich um ein kategorisches Feature handelt, und verwendet die Kategoriekodierung.
- Wenn jeder Knoten einen eigenen eindeutigen Wert für eine Eigenschaft hat, die als Kategorie-Feature inferiert wird, löscht Neptune ML die Eigenschaft aus dem Trainingsdiagramm, da es sich wahrscheinlich um eine ID handelt, die für das Lernen keine Informationen bereitstellt.
- Wenn bekannt ist, dass die Eigenschaft gültige Neptune-Trennzeichen wie Semikolon („;“) enthält, kann Neptune ML die Eigenschaft nur als `MultiNumerical` oder `MultiCategorical` behandeln.

  - Neptune ML versucht zunächst, die Werte als numerische Features zu kodieren. Wenn dies erfolgreich ist, verwendet Neptune ML die numerische Kodierung, um numerische Vektor-Features zu erstellen.
  - Andernfalls kodiert Neptune ML die Werte als Werte mit mehreren Kategorien.
- Wenn Neptune ML den Datentyp der Werte einer Eigenschaft nicht ableiten kann, Neptune MLdrops die Eigenschaft aus dem Trainingsdiagramm.

Bearbeiten einer Trainingsdaten-Konfigurationsdatei

Der Neptune-Exportvorgang exportiert Neptune-ML-Daten aus einem Neptune-DB-Cluster in einen S3-Bucket. Er exportiert Knoten und Kanten getrennt in die Ordner `nodes/` und `edges/`. Er erstellt außerdem eine JSON-Konfigurationsdatei für Trainingsdaten, die standardmäßig den Namen `training-data-configuration.json` hat. Diese Datei enthält Informationen zum Schema des Diagramms, zu den Typen der Features, zu den Operationen für Feature-Transformation und -Normalisierung sowie zum Ziel-Feature für eine Klassifizierungs- oder Regressionsaufgabe.

Es kann Fälle geben, in denen Sie Konfigurationsdatei direkt ändern möchten. Ein solcher Fall liegt vor, wenn Sie ändern möchten, wie Features verarbeitet werden oder wie das Diagramm konstruiert wird, ohne den Export jedes Mal erneut ausführen zu müssen, wenn Sie die Spezifikation für die Machine-Learning-Aufgabe ändern, die Sie lösen möchten.

Bearbeiten der Trainingsdaten-Konfigurationsdatei

1. Laden Sie die Datei zu Ihrem lokalen Computer herunter.

   Wenn Sie nicht einen oder mehrere benannte Aufträge in dem an den Exportvorgang übergebenen Parameter `additionalParams/neptune_ml` angegeben haben, hat die Datei den Standardnamen, `training-data-configuration.json`. Sie können einen AWS CLI-Befehl wie diesen verwenden, um die Datei herunterzuladen:

   ```
   aws s3 cp \
     s3://(your Amazon S3 bucket)/(path to your export folder)/training-data-
   configuration.json \
     ./
   ```

2. Bearbeiten Sie die Datei mit einem Texteditor.

3. Laden Sie die geänderte Datei hoch. Laden Sie die geänderte Datei mit einem AWS CLI-Befehl wie dem folgenden wieder an den gleichen Speicherort in Amazon S3 hoch, von dem Sie sie heruntergeladen haben:

   ```
   aws s3 cp \
     training-data-configuration.json \
     s3://(your Amazon S3 bucket)/(path to your export folder)/training-data-
   configuration.json
   ```

Beispiel für eine JSON-Konfigurationsdatei für Trainingsdaten

Dies ist ein Beispiel für eine Trainingsdaten-Konfigurationsdatei, die ein Diagramm für eine Aufgabe zur Knotenklassifizierung beschreibt:

```
{
  "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" : [ ]
  ]
}
```

Die Struktur der JSON-Konfigurationsdateien für Trainingsdaten

Die Trainingskonfigurationsdatei verweist auf CSV-Dateien, die vom Exportvorgang in den Ordnern `nodes/` und `edges/` gespeichert wurden.

In jeder Datei unter `nodes/` werden Informationen zu Knoten gespeichert, die dieselbe Eigenschaftsdiagramm-Knotenbezeichnung besitzen. Jede Spalte in einer Knotendatei speichert entweder die Knoten-ID oder die Knoteneigenschaft. Die erste Zeile der Datei enthält eine Kopfzeile mit der `~id` oder dem Eigenschaftsnamen für jede Spalte.

In jeder Datei unter `edges/` werden Informationen zu Knoten gespeichert, die dieselbe Eigenschaftsdiagramm-Kantenbezeichnung besitzen. Jede Spalte in einer Knotendatei speichert entweder die Quellknoten-ID, die Zielknoten-ID oder die Kanteneigenschaft. Die erste Zeile der Datei enthält eine Kopfzeile mit `~from`, `~to` oder dem Eigenschaftsnamen für jede Spalte.

Die Trainingsdaten-Konfigurationsdatei besitzt auf der obersten Ebene drei Elemente:

```
{
  "version" : "v2.0",
  "query_engine" : "gremlin",
  "graph" : [ ... ]
}
```

- `version` – (Zeichenfolge) Die Version der verwendeten Konfigurationsdatei.
- `query_engine` – (Zeichenfolge) Die Abfragesprache für den Export der Diagrammdaten. Zurzeit ist nur „Gremlin“ gültig.
- `graph` – (JSON-Array) listet ein oder mehrere Konfigurationsobjekte auf, die Modellparameter für alle Knoten und Kanten enthalten, die verwendet werden sollen.

  Die Konfigurationsobjekte im Diagramm-Array haben die im nächsten Abschnitt beschriebene Struktur.

Inhalt eines im **`graph`**-Array aufgelisteten Konfigurationsobjekts

Ein Konfigurationsobjekt im `graph`-Array kann auf der obersten Ebene drei Knoten enthalten:

```
  {
    "edges"    : [ ... ],
    "nodes"    : [ ... ],
    "warnings" : [ ... ],
  }
```

- `edges` – (Array von JSON-Objekten) Jedes JSON-Objekt gibt eine Reihe von Parametern an, um die Behandlung einer Kante im Diagramm während Modellverarbeitung und -training zu definieren. Dieser Knoten wird nur mit der Gremlin-Engine verwendet.
- `nodes` – (Array von JSON-Objekten) Jedes JSON-Objekt gibt eine Reihe von Parametern an, um die Behandlung eines Knotens im Diagramm während Modellverarbeitung und -training zu definieren. Dieser Knoten wird nur mit der Gremlin-Engine verwendet.
- `warnings` – (Array von JSON-Objekten) Jedes Objekt enthält eine Warnung, die während des Datenexportvorgangs generiert wird.

Inhalt eines im **`edges`**-Array aufgelisteten Kantenkonfigurationsobjekts

Ein in einem `edges`-Array aufgelistetes Kantenkonfigurationsobjekt kann auf der obersten Ebene die folgenden Felder enthalten:

```
    {
      "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`** – Eine Zeichenfolge, die den Pfad zu einer CSV-Datei angibt, in der Informationen zu Kanten mit derselben Eigenschaftsdiagramm-Bezeichnung gespeichert werden.

  Die erste Zeile dieser Datei enthält eine Kopfzeile mit Spaltenbezeichnungen.

  Die ersten beiden Spaltenbezeichnungen sind `~from` und `~to`. In der ersten Spalte (die Spalte `~from`) wird die ID des Startknotens der Kante gespeichert. In der zweiten Spalte (die Spalte `~to`) wird die ID des Endknotens der Kante gespeichert.

  Die verbleibenden Spaltenbezeichnungen in der Kopfzeile geben für jede verbleibende Spalte den Namen der Kanteneigenschaft an, deren Werte in diese Spalte exportiert wurden.
- **`separator`** – Eine Zeichenfolge mit dem Trennzeichen, das die Spalten in dieser CSV-Datei trennt.

- **`source`** – Ein JSON-Array mit zwei Zeichenfolgen, die den Startknoten der Kante angeben. Die erste Zeichenfolge enthält den Kopfzeilennamen der Spalte, in der die Startknoten-ID gespeichert ist. Die zweite Zeichenfolge gibt den Knotentyp an.
- **`relation`** – Ein JSON-Array mit zwei Zeichenfolgen, die den Beziehungstyp der Kante angeben. Die erste Zeichenfolge enthält den Kopfzeilennamen der Spalte, in der der Beziehungsname (`relname`) gespeichert ist. Die zweite Zeichenfolge enthält das Präfix für den Beziehungsnamen (`prefixname`).

  Der vollständige Beziehungstyp besteht aus den beiden Zeichenfolgen mit einem Bindestrich zwischen ihnen wie folgt: *`prefixname`*`-`*`relname`*.

  Wenn die erste Zeichenfolge leer ist, haben alle Kanten den gleichen Beziehungstyp, nämlich die `prefixname`-Zeichenfolge.
- **`dest`** – Ein JSON-Array mit zwei Zeichenfolgen, die den Endknoten der Kante angeben. Die erste Zeichenfolge enthält den Kopfzeilennamen der Spalte, in der die Endknoten-ID gespeichert ist. Die zweite Zeichenfolge gibt den Knotentyp an.
- **`features`** – Ein JSON-Array mit Eigenschaftswert-Feature-Objekten. Jedes Eigenschaftswert-Feature--Objekt enthält die folgenden Felder:
  - feature – Ein JSON-Array mit drei Zeichenfolgen. Die erste Zeichenfolge enthält den Kopfzeilennamen der Spalte, in der der Eigenschaftswert gespeichert ist. Die zweite Zeichenfolge enthält den Feature-Namen. Die dritte Zeichenfolge enthält den Feature-Typ.
  - norm – (Optional) Gibt die Normalisierungsmethode an, die auf die Eigenschaftswerte angewendet werden soll.

- **`labels`** – Ein JSON-Array von Objekten. Jedes dieser Objekte definiert ein Ziel-Feature der Kanten und gibt die Proportionen der Kanten für die Trainings- und Validierungsphasen an. Jedes Objekt enthält die folgenden Felder:
  - label – Ein JSON-Array mit zwei Zeichenfolgen. Die erste Zeichenfolge enthält den Kopfzeilennamen der Spalte, in der der Ziel-Feature-Eigenschaftswert gespeichert ist. Die zweite Zeichenfolge gibt einen der folgenden Zielaufgabentypen an:
    - `"classification"` – Eine Aufgabe zur Kantenklassifizierung. Die Eigenschaftswerte in der Spalte, die durch die erste Zeichenfolge im `label`-Array identifiziert wird, werden als kategorische Werte behandelt. Bei einer Aufgabe zur Kantenklassifizierung darf die erste Zeichenfolge im `label`-Array nicht leer sein.
    - `"regression"` – Eine Kantenregressionsaufgabe. Die Eigenschaftswerte in der Spalte, die durch die erste Zeichenfolge im `label`-Array identifiziert wird, werden als numerische Werte

behandelt. Für eine Aufgabe zur Kantenregression darf die erste Zeichenfolge im `label`-Array nicht leer sein.

- `"link_prediction"` – Eine Aufgabe zur Linkvorhersage. Es sind keine Eigenschaftswerte erforderlich. Bei einer Aufgabe zur Linkvorhersage wird die erste Zeichenfolge im `label`-Array ignoriert.

- **`split_rate`** – Ein JSON-Array mit drei Zahlen zwischen null und eins, die zusammen eins ergeben und eine Schätzung des Anteils der Knoten darstellen, die in der Trainings-, Validierungs- und Testphase jeweils verwendet werden. Sie können dieses Feld oder das Feld `custom_split_filenames` definieren, aber nicht beide. Siehe split_rate.
- **`custom_split_filenames`** – Ein JSON-Objekt, das die Dateinamen für die Dateien angibt, die die Trainings-, Validierungs- und Testpopulationen definieren. Sie können dieses Feld oder das Feld `split_rate` definieren, aber nicht beide. Weitere Informationen finden Sie unter Benutzerdefinierte train-validation-test Proportionen.

Inhalt eines im **`nodes`**-Array aufgelisteten Knotenkonfigurationsobjekts

Ein in einem `nodes`-Array aufgelistetes Knotenkonfigurationsobjekt kann die folgenden Felder enthalten:

```
{
  "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`** – Eine Zeichenfolge, die den Pfad zu einer CSV-Datei angibt, in der Informationen zu Knoten mit derselben Eigenschaftsdiagramm-Bezeichnung gespeichert werden.

  Die erste Zeile dieser Datei enthält eine Kopfzeile mit Spaltenbezeichnungen.

  Die erste Spaltenbezeichnung ist `~id` und die erste Spalte (die Spalte `~id`) speichert die Knoten-ID.

  Die verbleibenden Spaltenbezeichnungen in der Kopfzeile geben für jede verbleibende Spalte den Namen der Knoteneigenschaft an, deren Werte in diese Spalte exportiert wurden.

- **`separator`** – Eine Zeichenfolge mit dem Trennzeichen, das die Spalten in dieser CSV-Datei trennt.
- **`node`** – Ein JSON-Array mit zwei Zeichenfolgen. Die erste Zeichenfolge enthält den Header-Namen der Spalte, in der der Knoten gespeichert ist IDs. Die zweite Zeichenfolge gibt den Knotentyp im Diagramm an, der einer Eigenschaftsdiagrammbezeichnung des Knotens entspricht.
- **`features`** – Ein JSON-Array mit Knoten-Feature-Objekten. Siehe Inhalt eines Feature-Objekts, das in einem `features`-Array für einen Knoten oder eine Kante aufgelistet ist.
- **`labels`** – Ein JSON-Array mit Knotenbezeichnungsobjekten. Siehe Inhalt eines im `labels`-Knoten-Array aufgelisteten Knotenbezeichnungsobjekts.

Inhalt eines Feature-Objekts, das in einem **`features`**-Array für einen Knoten oder eine Kante aufgelistet ist

Ein in einem `features`-Array aufgelistetes Knoten-Feature-Objekt kann auf der obersten Ebene die folgenden Felder enthalten:

- **`feature`** – Ein JSON-Array mit drei Zeichenfolgen. Die erste Zeichenfolge enthält den Kopfzeilennamen der Spalte, in der der Eigenschaftswert für das Feature gespeichert ist. Die zweite Zeichenfolge enthält den Feature-Namen.

  Die dritte Zeichenfolge enthält den Feature-Typ. Gültige Feature-Typen werden in Mögliche Werte des Typfeldes für Features aufgelistet.
- **`norm`** – Dies ist ein Pflichtfeld für numerische Features. Es gibt die Normalisierungsmethode an, die für numerische Werte verwendet werden soll: Gültige Werte sind `"none"`, `"min-max"` und „Standard“. Details dazu finden Sie unter Das Normfeld.
- **`language`** – Das Feld für die Sprache gibt die Sprache an, die in Texteigenschaftswerten verwendet wird. Die Nutzung ist von der Textkodierungsmethode abhängig:
  - Dieses Feld ist für die `text_fasttext`-Kodierung erforderlich und es muss eine der folgenden Sprachen angegeben werden:
    - `en` (Englisch)
    - `zh` (Chinesisch)
    - `hi` (Hindi)
    - `es` (Spanisch)
    - `fr` (Französisch)

`text_fasttext` kann jedoch nicht mehr als eine Sprache gleichzeitig verarbeiten.

- Dieses Feld wird für die `text_sbert`-Kodierung nicht verwendet, da die SBERT-Kodierung mehrsprachig ist.
- Dieses Feld ist für die `text_word2vec`-Kodierung optional, da `text_word2vec` nur Englisch unterstützt. Wenn vorhanden, muss der Name des englischsprachigen Modells angegeben werden:

```
"language" : "en_core_web_lg"
```

- Dieses Feld wird für die `tfidf`-Kodierung nicht verwendet.

- **`max_length`** – Dieses Feld ist optional für `text_fasttext`-Features. Es gibt die maximale Anzahl von Token in einem Eingabe-Text-Feature an, die kodiert werden. Text, der eingegeben wird, nachdem `max_length` erreicht wurde, wird ignoriert. Wenn Sie beispielsweise max_length auf 128 festlegen, werden alle Token in einer Textsequenz nach dem 128. Zeichen ignoriert.
- **`separator`** – Dieses Feld wird optional mit den Features `category`, `numerical` und `auto` verwendet. Es gibt ein Zeichen an, das für die Aufteilung eines Eigenschaftswerts in mehrere kategorische oder numerische Werte verwendet werden kann:

  Siehe Das Trennfeld.
- **`range`** – Dies ist ein Pflichtfeld für `bucket_numerical`-Features. Es gibt den Bereich der numerischen Werte an, die in Buckets aufgeteilt werden sollen.

  Siehe Das Bereichsfeld.
- **`bucket_cnt`** – Dies ist ein Pflichtfeld für `bucket_numerical`-Features. Es gibt die Anzahl der Buckets an, in die der numerische Bereich unterteilt werden soll, der durch den Parameter `range` definiert wird.

  Siehe Bucket-numerische Features in Neptune ML.
- **`slide_window_size`** – Dieses Feld wird optional mit `bucket_numerical`-Features verwendet, um mehr als einem Bucket Werte zuzuweisen.

  Siehe Das Feld slide_window_size.
- **`imputer`** – Dieses Feld wird optional mit `numerical`-, `bucket_numerical`- und `datetime`-Features verwendet, um eine Imputationstechnik zum Ausfüllen fehlender Werte bereitzustellen. Die unterstützten Imputationstechniken sind `"mean"`, `"median"` und `"most_frequent"`.

Siehe Das Imputer-Feld.

- **`max_features`** – Dieses Feld wird optional von `text_tfidf`-Features verwendet, um die maximale Anzahl der Begriffe anzugeben, die kodiert werden sollen:

  Siehe Das Feld max_features.

- **`min_df`** – Dieses Feld wird optional von `text_tfidf`-Features verwendet, um die Mindesthäufigkeit von Begriffen anzugeben, die kodiert werden sollen:

  Siehe Das Feld min_df.

- **`ngram_range`** – Dieses Feld wird optional von `text_tfidf`-Features verwendet, um den Bereich von Wörtern oder Token anzugeben, die als potenzielle einzelne Begriffe zur Kodierung betrachtet werden sollen.

  Siehe Das Feld ngram_range.

- **`datetime_parts`** – Dieses Feld wird optional von `datetime`-Features verwendet, um anzugeben, welche Teile des datetime-Werts kategorisch kodiert werden sollen.

  Siehe Das Feld datetime_parts.

Inhalt eines im **`labels`**-Knoten-Array aufgelisteten Knotenbezeichnungsobjekts

Ein in einem `labels`-Knoten-Array aufgelistetes Bezeichnungsobjekt definiert ein Knotenziel-Feature und gibt die Proportionen der Knoten an, die in den Trainings-, Validierungs- und Testphasen verwendet werden. Jedes Objekt kann die folgenden Felder enthalten:

```
    {
      "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`** – Ein JSON-Array mit zwei Zeichenfolgen. Die erste Zeichenfolge enthält den Kopfzeilennamen der Spalte, in der die Eigenschaftswerte für das Feature gespeichert sind. Die zweite Zeichenfolge gibt den Zielaufgabentyp an, der Folgendes sein kann:

  - `"classification"` – Eine Aufgabe zur Knotenklassifizierung. Die Eigenschaftswerte in der angegebenen Spalte werden verwendet, um ein kategorisches Feature zu erstellen.
  - `"regression"` – Eine Aufgabe zur Knotenregression. Die Eigenschaftswerte in der angegebenen Spalte werden verwendet, um ein numerisches Feature zu erstellen.
- **`split_rate`** – Ein JSON-Array mit drei Zahlen zwischen null und eins, die zusammen eins ergeben und eine Schätzung des Anteils der Knoten darstellen, die in der Trainings-, Validierungs- und Testphase jeweils verwendet werden. Siehe split_rate.
- **`custom_split_filenames`** – Ein JSON-Objekt, das die Dateinamen für die Dateien angibt, die die Trainings-, Validierungs- und Testpopulationen definieren. Sie können dieses Feld oder das Feld `split_rate` definieren, aber nicht beide. Weitere Informationen finden Sie unter Benutzerdefinierte train-validation-test Proportionen.
- **`separator`** – Eine Zeichenfolge, die das Trennzeichen enthält, das kategorische Feature-Werte für eine Klassifizierungsaufgabe trennt.

> **Note**
>
> Wenn kein Bezeichnungsobjekt sowohl für Kanten als auch für Knoten angegeben wird, wird automatisch angenommen, dass es sich bei der Aufgabe um eine Linkvorhersage handelt. Die Kanten werden randomisiert in 90 % zum Training und 10 % zur Validierung aufgeteilt.

Benutzerdefinierte train-validation-test Proportionen

Standardmäßig wird der Parameter `split_rate` von Neptune ML verwendet, um das Diagramm anhand der im Parameter definierten Anteile randomisiert in Trainings-, Validierungs- und Testpopulationen aufzuteilen. Um eine genauere Kontrolle über die in diesen Populationen verwendeten Entitäten zu erhalten, können Dateien zu ihrer expliziten Definition erstellt werden. Anschließend kann die Trainingsdaten-Konfigurationsdatei bearbeitet werden, um diese indizierenden Dateien den Populationen zuzuordnen. Diese Zuordnung wird durch ein JSON-Objekt für den Schlüssel `custom_split_filesnames` in der Trainingskonfigurationsdatei spezifiziert. Wenn diese Option verwendet wird, müssen für die Schlüssel `train` und `validation` Dateinamen angegeben werden, optional auch für den Schlüssel `test`.

Die Formatierung dieser Dateien sollte dem Gremlin-Datenformat entsprechen. Insbesondere bei Aufgaben auf Knotenebene sollte jede Datei eine Spalte mit der `~id` Überschrift enthalten, in der der Knoten aufgeführt ist IDs, und bei Aufgaben auf Kantenebene sollten die Dateien die Quell `~from`

- `~to` bzw. Zielknoten der Kanten spezifizieren und angeben. Diese Dateien müssen am selben Amazon-S3-Speicherort platziert werden wie die exportierten Daten, die für die Datenverarbeitung verwendet werden (siehe `outputS3Path`).

Bei Aufgaben zur Klassifizierung oder Regression von Eigenschaften können diese Dateien optional die Bezeichnungen für die Machine-Learning-Aufgabe definieren. In diesem Fall müssen die Dateien eine Eigenschaftsspalte mit dem Kopfzeilennamen haben, der in der Trainingsdaten-Konfigurationsdatei definiert ist. Wenn Eigenschaftsbezeichnungen sowohl in den exportierten Knoten- und Kantendateien als auch in den benutzerdefiniert aufgeteilten Dateien definiert sind, haben die benutzerdefiniert aufgeteilten Dateien Vorrang.

Trainieren eines Modells mit Neptune ML

Nachdem Sie die Daten verarbeitet haben, die Sie aus Neptune für das Modelltraining exportiert haben, können Sie einen Modelltrainingsjob mit einem Befehl wie dem folgenden starten:

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"
```

Weitere Informationen finden Sie unter start-ml-model-training-job in der Befehlsreferenz. 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.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**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

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"
      }'
```

Details zur Verwendung dieses Befehls werden in Der Befehl modeltraining beschrieben. Dort finden Sie auch Informationen dazu, wie Sie ausgeführte Aufträge abrufen und beenden und alle ausgeführten Aufträge auflisten.

Sie können auch eine `previousModelTrainingJobId` angeben, um mittels Informationen aus einem abgeschlossenen Neptune-ML-Modell-Trainingsauftrag die Hyperparametersuche in einem neuen Trainingsauftrag zu beschleunigen. Dies ist während erneuter Modelltrainings anhand neuer Diagrammdaten als auch während inkrementeller Trainings anhand derselben Diagrammdaten nützlich. Verwenden Sie einen Befehl wie diesen:

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)"
```

Weitere Informationen finden Sie unter start-ml-model-training-job in der AWS CLI Befehlsreferenz.

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**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

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)"
      }'
```

Sie können Ihre eigene Modellimplementierung in der Neptune-ML-Trainingsinfrastruktur trainieren, indem Sie ein `customModelTrainingParameters`-Objekt bereitstellen, z. B.:

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)"
  }'
```

Weitere Informationen finden Sie unter start-ml-model-training-job in der AWS CLI Befehlsreferenz.

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**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

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)"
        }
```

```
    }'
```

Weitere Informationen, z. B. zum Abrufen des Status eines ausgeführten Auftrags, zum Beenden eines ausgeführten Auftrags und zum Auflisten aller ausgeführten Aufträge, finden Sie unter Der Befehl modeltraining. Informationen zur Implementierung eines benutzerdefinierten Modells finden Sie unter Benutzerdefinierte Modelle in Neptune ML.

Themen

- Modelle und Modelltraining in Amazon Neptune ML
- Anpassen von Modell-Hyperparameter-Konfigurationen in Neptune ML
- Bewährte Modelltrainingsmethoden

Modelle und Modelltraining in Amazon Neptune ML

Neptune ML verwendet Graph Neural Networks (GNN), um Modelle für die verschiedenen Machine-Learning-Aufgaben zu erstellen. Es wurde gezeigt, dass neuronale Graphennetzwerke state-of-the-art Ergebnisse für Aufgaben des maschinellen Lernens mit Graphen erzielen und hervorragend darin sind, informative Muster aus graphstrukturierten Daten zu extrahieren.

Neuronale Netzwerke (GNNs) in Neptune ML grafisch darstellen

Graph Neural Networks (GNNs) gehören zu einer Familie neuronaler Netze, die Knotenrepräsentationen berechnen, indem sie die Struktur und Merkmale nahegelegener Knoten berücksichtigen. GNNs ergänzen andere traditionelle Methoden des maschinellen Lernens und neuronaler Netzwerke, die für Graphdaten nicht gut geeignet sind.

GNNs werden verwendet, um Aufgaben des maschinellen Lernens wie Knotenklassifizierung und Regression (Vorhersage der Eigenschaften von Knoten) und Kantenklassifizierung und Regression (Vorhersage der Eigenschaften von Kanten) oder Linkvorhersage (Vorhersage, ob zwei Knoten im Graphen miteinander verbunden sein sollten oder nicht) zu lösen.

Im Allgemeinen umfasst die Verwendung eines GNN für eine Machine-Learning-Aufgabe zwei Phasen:

- Eine Kodierungsphase, in der das GNN für jeden Knoten im Diagramm einen d-dimensionalen Vektor berechnet. Diese Vektoren werden auch Repräsentationen oder Einbettungen genannt.
- Eine Dekodierungsphase, in der auf Grundlage der kodierten Repräsentationen Vorhersagen erstellt werden.

Bei der Klassifizierung und Regression von Knoten werden die Knotenrepräsentationen direkt für die Klassifizierungs- und Regressionsaufgaben verwendet. Bei der Klassifizierung und Regression von Kanten werden die Knotendarstellungen der Vorfallknoten an einer Kante als Eingabe für die Klassifizierung oder Regression verwendet. Für die Linkvorhersage wird anhand eines Paars von Knotenrepräsentationen und einer Kantentypdarstellung ein Kantenwahrscheinlichkeitswert berechnet.

Die Deep Graph Library (DGL) ermöglicht die effiziente Definition und Schulung dieser Aufgaben. GNNs

Verschiedene GNN-Modelle sind unter der Formulierung der Nachrichtenübergabe konsolidiert. In dieser Ansicht wird die Darstellung eines Knotens in einem Diagramm anhand der Darstellungen

der Knotennachbarn (der Nachrichten) zusammen mit der ursprünglichen Darstellung des Knotens berechnet. In Neptune ML wird die ursprüngliche Darstellung eines Knotens von den Features abgeleitet, die aus dessen Eigenschaften extrahiert wurden. Sie kann auch erlernbar und von der Identität des Knotens abhängig sein.

Neptune ML ermöglicht auch die Verkettung von Knoten-Features und erlernbaren Knotendarstellungen, um als ursprüngliche Knotendarstellung verwendet zu werden.

Für die verschiedenen Aufgaben in Neptune ML, die Diagramme mit Knoteneigenschaften beinhalten, wird in der Kodierungsphase das Relational Graph Convolutional Network (R-GCN) verwendet. R-GCN ist eine GNN-Architektur, die gut für Diagramme mit mehreren Knoten- und Kantentypen geeignet ist (als heterogene Diagramme bezeichnet).

Das R-GCN-Netzwerk besteht aus einer festen Anzahl von Ebenen, die aufeinander gestapelt sind. Jede R-GCN-Ebene verwendet ihre erlernbaren Modellparameter, um Informationen aus der unmittelbaren 1-Hop-Nachbarschaft eines Knotens zu aggregieren. Da nachfolgende Ebenen die Ausgabedarstellungen der vorherigen Ebene als Eingabe verwenden, ist der Radius der Diagrammnachbarschaft, der die endgültige Einbettung eines Knotens beeinflusst, von der Anzahl der Ebenen (`num-layer`) des R-GCN-Netzwerks abhängig.

Das bedeutet beispielsweise, dass ein Netzwerk mit 2 Ebenen Informationen aus Knoten verwendet, die 2 Hops entfernt sind.

Weitere Informationen dazu finden Sie unter Umfassender Überblick über GNNs neuronale Graphennetzwerke. Weitere Informationen zur Deep Graph Library (DGL) finden Sie auf der DGL-Website. Ein praktisches Tutorial zur Verwendung von DGL mit GNNs finden Sie unter Lernen neuronaler Graphennetze mit der Deep Graph Library.

Trainieren von Graph Neural Networks

Im Machine Learning wird der Vorgang, in dem ein Modell lernt, gute Vorhersagen für eine Aufgabe zu erstelen, als Modelltraining bezeichnet. Dies erfolgt in der Regel durch die Angabe eines Optimierungsziels und eines Algorithmus für die Ausführung dieser Optimierung.

Dieser Vorgang wird verwendet, um ein GNN für die Erstellung guter Darstellungen auch der nachgelagerten Aufgaben zu trainieren. Für diese Aufgabe wird eine Zielfunktion erstellt, die während des Modelltrainings minimiert wird. Für die Knotenklassifizierung verwenden CrossEntropyLosswir beispielsweise das Ziel, wodurch Fehlklassifizierungen bestraft werden, und für die Knotenregression minimieren wir. MeanSquareError

Das Ziel ist in der Regel eine Verlustfunktion, bei der die Modellvorhersagen für einen bestimmten Datenpunkt mit dem Grundwahrheitswert für diesen Datenpunkt verglichen werden. Sie gibt den Verlustwert zurück, der angibt, wie weit die Vorhersagen des Modells abweichen. Das Ziel des Trainings ist, den Verlust zu minimieren und sicherzustellen, dass Modellvorhersagen der Grundwahrheit nahe kommen.

Der Optimierungsalgorithmus, der beim Deep Learning für das Training verwendet wird, ist in der Regel eine Variante des Gradientenabstiegs. In Neptune ML wird Adam verwendet, ein Algorithmus für die gradientenbasierte Optimierung erster Ordnung für stochastische Zielfunktionen, der auf adaptiven Schätzungen von Momenten niedrigerer Ordnung basiert.

Während das Modelltraining sicherstellen soll, dass die erlernten Modellparameter nahe bei den Mindestwerten der Zielfunktion liegen, ist die Gesamtleistung eines Modells auch von dessen Hyperparametern abhängig. Dies sind Modelleinstellungen, die nicht vom Trainingsalgorithmus erlernt werden. Beispielsweise ist die Dimensionalität der erlernten Knotendarstellung (`num-hidden`) ein Hyperparameter, der sich auf die Modellleistung auswirkt. Daher wird im Machine Learning in der Regel eine Hyperparameter-Optimierung (HPO) durchgeführt, um geeignete Hyperparameter auszuwählen.

Neptune ML verwendet einen SageMaker KI-Hyperparameter-Tuning-Job, um mehrere Instanzen von Modelltraining mit unterschiedlichen Hyperparameter-Konfigurationen zu starten, um zu versuchen, das beste Modell für eine Reihe von Hyperparameter-Einstellungen zu finden. Siehe Anpassen von Modell-Hyperparameter-Konfigurationen in Neptune ML.

Modelle zur Einbettung von Wissensdiagrammen in Neptune ML

Wissensgraphen (KGs) sind Graphen, die Informationen über verschiedene Entitäten (Knoten) und ihre Beziehungen (Kanten) kodieren. In Neptune ML werden standardmäßig Modelle zur Einbettung von Wissensdiagrammen für Linkvorhersagen verwendet, wenn ein Diagramm keine Knoteneigenschaften, sondern lediglich Beziehungen zu anderen Knoten enthält. Obwohl für diese Diagramme auch R-GCN-Modelle mit erlernbaren Einbettungen verwendet werden können, indem der Modelltyp als `"rgcn"` angegeben wird, sind Modelle zur Einbettung von Wissensdiagrammen einfacher und eine effektive Lösung für das Erlernen von Darstellungen für umfangreiche Wissensdiagramme.

Modelle zur Einbettung von Wissensdiagrammen werden in Aufgaben für die Linkvorhersage verwendet, um die Knoten oder Beziehungen vorherzusagen, die ein Tripel (**`h, r, t`**) bilden, wobei **`h`** der Quellknoten, **`r`** der Beziehungstyp und **`t`** der Zielknoten ist.

Die in Neptune ML implementierten Modelle zur Einbettung von Wissensdiagrammen sind `distmult`, `transE` und `rotatE`. Weitere Informationen zu Modellen zur Einbettung von Wissensdiagrammen finden Sie unter DGL-KE.

Trainieren benutzerdefinierter Modelle in Neptune ML

Mit Neptune ML können Sie für bestimmte Szenarien eigene benutzerdefinierte Modelle definieren und implementieren. Informationen zur Implementierung eines benutzerdefinierten Modells und zum Trainieren dieses Modells in der Neptune-ML-Infrastruktur finden Sie unter Benutzerdefinierte Modelle in Neptune ML.

Anpassen von Modell-Hyperparameter-Konfigurationen in Neptune ML

Wenn Sie einen Neptune-ML-Modelltrainingsauftrag starten, verwendet Neptune ML automatisch die aus dem vorherigen Datenverarbeitungsauftrag inferierten Informationen. Es verwendet die Informationen, um Hyperparameter-Konfigurationsbereiche zu generieren, die verwendet werden, um einen SageMaker KI-Hyperparameter-Tuning-Job zu erstellen, um mehrere Modelle für Ihre Aufgabe zu trainieren. So müssen Sie keine lange Liste von Hyperparameterwerten für die Modelle angeben, mit denen trainiert werden soll. Stattdessen werden die Hyperparameterbereiche und Standardwerte des Modells auf Grundlage von Aufgabentyp, Diagrammtyp und Einstellungen des Optimierungsauftrags ausgewählt.

Sie können jedoch die Standard-Hyperparameterkonfiguration überschreiben und benutzerdefinierte Hyperparameter bereitstellen, indem Sie eine vom Datenverarbeitungsauftrag generierte JSON-Konfigurationsdatei ändern.

Mithilfe der Neptune-ML-API modelTraining können Sie mehrere Einstellungen für Hyperparameter-Optimierungsaufträge auf hoher Ebene wie `maxHPONumberOfTrainingJobs`, `maxHPOParallelTrainingJobs` und `trainingInstanceType` steuern. Zur genaueren Steuerung der Modellhyperparameter können Sie die Datei `model-HPO-configuration.json` ändern, die vom Datenverarbeitungsauftrag generiert wird. Die Datei wird an dem Amazon-S3-Speicherort gespeichert, den Sie für die Ausgabe von Verarbeitungsaufträgen angegeben haben.

Sie können die Datei herunterladen, zur Überschreibung der Standard-Hyperparameterkonfigurationen bearbeiten und wieder zum selben Amazon-S3-Speicherort hochladen. Ändern Sie den Namen der Datei nicht und befolgen Sie bei der Bearbeitung die folgenden Anweisungen.

Herunterladen der Datei von Amazon S3:

```
aws s3 cp \
  s3://(bucket name)/(path to output folder)/model-HPO-configuration.json \
  ./
```

Wenn die Bearbeitung abgeschlossen ist, laden Sie die Datei wieder zum ursprünglichen Speicherort hoch:

```
aws s3 cp \
  model-HPO-configuration.json \
  s3://(bucket name)/(path to output folder)/model-HPO-configuration.json
```

Struktur der Datei **model-HPO-configuration.json**

Die Datei `model-HPO-configuration.json` gibt das Modell an, das trainiert werden soll, den `task_type` für das Machine Learning sowie die Hyperparameter, die für die verschiedenen Modelltrainingsausführungen variabel oder fest sein sollten.

Die Hyperparameter werden verschiedenen Stufen zugeordnet, die die Priorität der Hyperparameter beim Aufruf des Hyperparameter-Optimierungsauftrags angeben:

- Tier-1-Hyperparameter haben die höchste Priorität. Wenn Sie `maxHPONumberOfTrainingJobs` auf einen Wert unter 10 festlegen, werden nur Tier-1-Hyperparameter optimiert. Die übrigen Hyperparameter verwenden die Standardwerte.
- Tier-2-Hyperparameter haben eine niedrigere Priorität. Wenn es also mehr als 10, aber insgesamt weniger als 50 Trainingsaufträge für einen Optimierungsjob gibt, werden sowohl Tier-1- als auch Tier-2-Hyperparameter optimiert.
- Tier-3-Hyperparameter werden nur dann zusammen mit Tier-1- und Tier-2- optimiert, wenn es insgesamt mehr als 50 Trainingsaufträge gibt.
- Und schließlich werden feste Hyperparameter überhaupt nicht optimiert und verwenden stets die Standardwerte.

Beispiel für eine **model-HPO-configuration.json**-Datei

Das Folgende ist eine `model-HPO-configuration.json`-Beispieldatei:

```
{
  "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
      }
    ]
  }
]
```

```
}
```

Elemente einer **`model-HPO-configuration.json`**-Datei

Die Datei enthält ein JSON-Objekt mit einem einzigen Array auf oberster Ebene mit dem Namen `models` und einem einzigen Modellkonfigurationsobjekt. Achten Sie beim Anpassen der Datei darauf, dass das `models`-Array nur ein einziges Modellkonfigurationsobjekt enthält. Wenn Ihre Datei mehr als ein Modellkonfigurationsobjekt enthält, schlägt der Optimierungsauftrag mit einer Warnung fehl.

Das Modellkonfigurationsobjekt enthält auf der obersten Ebene die folgenden Elemente:

- **`model`** – (Zeichenfolge) Der Modelltyp, der trainiert werden soll (nicht ändern). Gültige Werte sind:
  - `"rgcn"` – Dies ist die Standardeinstellung für Aufgaben zur Klassifizierung und Regression von Knoten sowie heterogene Linkvorhersageaufgaben.
  - `"transe"` – Dies ist die Standardeinstellung für KGE-Linkvorhersageaufgaben.
  - `"distmult"` – Dies ist ein alternativer Modelltyp für KGE-Linkvorhersageaufgaben.
  - `"rotate"` – Dies ist ein alternativer Modelltyp für KGE-Linkvorhersageaufgaben.

  Sie sollten den `model`-Wert nicht direkt ändern, da für verschiedene Modelltypen häufig sehr unterschiedliche Hyperparameter gelten, was nach dem Start des Trainingsauftrags zu Analysefehlern führen kann.

  Um den Modelltyp zu ändern, verwenden Sie den Parameter `modelName` in der modelTraining-API, statt den Modelltyp in der `model-HPO-configuration.json`-Datei zu ändern.

  Sie können den Modelltyp ändern und detaillierte Änderungen für die Hyperparameter ausführen, indem Sie die Modellkonfigurations-Standardvorlage des Modells kopieren, das Sie verwenden möchten, und in die `model-HPO-configuration.json`-Datei einfügen. Wenn der inferierte Aufgabentyp mehrere Modelle unterstützt, befindet sich am Amazon-S3-Speicherort der Datei `model-HPO-configuration.json` ein Ordner namens `hpo-configuration-templates`. Dieser Ordner enthält alle Standard-Hyperparameter-Konfigurationen für die übrigen Modelle für die Aufgabe.

  Wenn Sie beispielsweise die Modell- und Hyperparameterkonfigurationen für eine KGE-Linkvorhersageaufgabe vom `transe`-Standardmodell in ein `distmult`-Modell ändern möchten, fügen Sie einfach den Inhalt der Datei `hpo-configuration-templates/distmult.json`

in die Datei `model-HPO-configuration.json` und bearbeiten dann die Hyperparameter wie notwendig.

> **Note**
>
> Wenn Sie den Parameter `modelName` in der `modelTraining`-API festlegen sowie `model` und die Hyperparameterspezifikation in der Datei `model-HPO-configuration.json` ändern und sich diese unterscheiden, hat der Wert für `model` in der Datei `model-HPO-configuration.json` Priorität und der Wert für `modelName` wird ignoriert.

- **`task_type`** – (Zeichenfolge) Der Machine-Learning-Aufgabentyp, der vom Datenverarbeitungsauftrag inferiert oder direkt an den Datenverarbeitungsauftrag übergeben wird (nicht ändern). Folgende sind gültige Werte:
  - `"node_class"`
  - `"node_regression"`
  - `"link_prediction"`

  Der Datenverarbeitungsauftrag inferiert den Aufgabentyp durch die Untersuchung des exportierten Datensatzes und der generierten Trainingsauftrags-Konfigurationsdatei auf Eigenschaften des Datensatzes.

  Dieser Wert sollte nicht geändert werden. Wenn Sie eine andere Aufgabe trainieren möchten, müssen Sie einen neuen Datenverarbeitungsauftrag ausführen. Wenn der Wert für `task_type` nicht wie erwartet ist, sollten Sie die Eingaben für den Datenverarbeitungsauftrag überprüfen, um sicherzustellen, dass sie korrekt sind. Dies umfasst die Parameter für die `modelTraining`-API sowie in der durch den Datenexportvorgang generierten Trainingsauftrags-Konfigurationsdatei.
- **`eval_metric`** – (Zeichenfolge) Die Auswertungsmetrik sollte zur Auswertung der Modellleistung und zur Auswahl des Modells mit der besten Leistung für alle HPO-Ausführungen verwendet werden. Folgende sind gültige Werte:
  - `"acc"` – Genauigkeit der Standardklassifizierung. Dies ist die Standardeinstellung für Klassifizierungsaufgaben mit einer einzigen Bezeichnung, wenn während der Datenverarbeitung keine unausgewogenen Bezeichnungen gefunden werden. In diesem Fall ist die Standardeinstellung `"F1"`.
  - `"acc_topk"` – Die Häufigkeit, mit der sich die richtige Bezeichnung unter den Top-Vorhersagen für **k** befindet. Sie können den Wert für **k** auch festlegen, indem Sie `topk` als zusätzlichen Schlüssel übergeben.

- `"F1"` – Das F1-Ergebnis.
- `"mse"` – Mean-Squared-Fehler-Metrik für Regressionsaufgaben.
- `"mrr"` – Mean-Reciprocal-Rang-Metrik.
- `"precision"` – Die Modellpräzision, berechnet als Verhältnis der echten positiven Ergebnisse zu den vorhergesagten positiven Ergebnissen: `= true-positives / (true-positives + false-positives)`.
- `"recall"` – Die Modellsensitivität, berechnet als Verhältnis der echten positiven Ergebnisse zu den tatsächlichen Ergebnissen: `= true-positives / (true-positives + false-negatives)`.
- `"roc_auc"` – Die Fläche unter der ROC-Kurve. Dies ist die Standardeinstellung für Klassifizierungen mit mehreren Bezeichnungen.

Um die Metrik beispielsweise in F1 zu ändern, ändern Sie den Wert für `eval_metric` wie folgt:

```
"  eval_metric": {
    "metric": "F1",
  },
```

Um die Metrik in eine `topk`-Präzisionsbewertung zu ändern, würden Sie `eval_metric` wie folgt ändern:

```
  "eval_metric": {
    "metric": "acc_topk",
    "topk": 2
  },
```

- **`eval_frequency`** – (Objekt) Gibt an, wie häufig die Leistung des Modells im Validierungssatz während des Trainings überprüft werden soll. Auf Grundlage der Validierungsleistung kann dann ein vorzeitiger Stopp eingeleitet und das beste Modell gespeichert werden.

  Das `eval_frequency`-Objekt enthält zwei Elemente, `"type"` und `"value"`. Beispiel:

```
  "eval_frequency": {
    "type":  "evaluate_every_pct",
    "value":  0.1
  },
```

  Gültige Werte für `type` sind:

- **`evaluate_every_pct`** – Gibt den Prozentsatz der Trainings an, die für jede Auswertung abgeschlossen werden müssen.

  Für `evaluate_every_pct` enthält das Feld `"value"` eine Gleitkommazahl zwischen null und eins, die diesen Prozentsatz ausdrückt.

- **`evaluate_every_batch`** – Gibt die Anzahl der Trainings-Batches an, die für jede Auswertung abgeschlossen werden müssen.

  Für `"value"` enthält das Feld `evaluate_every_batch` eine Ganzzahl, die diese Batch-Anzahl ausdrückt.

- **`evaluate_every_epoch`** – Gibt die Anzahl der Epochen pro Auswertung an, wobei eine neue Epoche stets um Mitternacht beginnt.

  Für `evaluate_every_epoch` enthält das Feld `"value"` eine Ganzzahl, die diese Epochen-Anzahl ausdrückt.

Die Standardeinstellung für `eval_frequency` ist:

```
"eval_frequency": {
  "type":  "evaluate_every_epoch",
  "value":  1
},
```

- **`1-tier-param`** – (Erforderlich) Ein Array von Tier-1-Hyperparametern.

  Wenn Sie keine Hyperparameter optimieren möchten, können Sie diesen Wert auf ein leeres Array festlegen. Dies hat keinen Einfluss auf die Gesamtzahl der Trainingsjobs, die durch den SageMaker KI-Hyperparameter-Tuning-Job gestartet wurden. Es bedeutet lediglich, dass alle Trainingsaufträge mit dem gleichen Satz von Hyperparametern ausgeführt werden, wenn es mehr als 1, aber weniger als 10 gibt.

  Wenn Sie andererseits alle optimierbaren Hyperparameter mit gleicher Bedeutung behandeln möchten, können Sie alle Hyperparameter in dieses Array einfügen.

- **`2-tier-param`** – (Erforderlich) Ein Array von Tier-2-Hyperparametern.

  Diese Parameter werden nur optimiert, wenn der Wert für `maxHPONumberOfTrainingJobs` größer als 10 ist. Andernfalls werden die Standardwerte verwendet.

Wenn Sie über ein Trainingsbudget von maximal 10 Trainingsaufträgen verfügen oder aus einem anderen Grund keine Tier-2-Hyperparameter verwenden möchten, jedoch alle optimierbaren Hyperparameter optimieren möchten, können Sie diesen Wert auf ein leeres Array festlegen.

- **`3-tier-param`** – (Erforderlich) Ein Array von Tier-3-Hyperparametern.

  Diese Parameter werden nur optimiert, wenn der Wert für `maxHPONumberOfTrainingJobs` größer als 50 ist. Andernfalls werden die Standardwerte verwendet.

  Wenn Sie keine Tier-3-Hyperparameter optimieren möchten, können Sie diesen Wert auf ein leeres Array festlegen.

- **`fixed-param`** – (Erforderlich) Ein Array von festen Hyperparametern, die nur die Standardwerte annehmen und sich in unterschiedlichen Trainingsaufträgen nicht unterscheiden.

  Wenn alle Hyperparameter variabel sein sollen, können Sie diesen Wert auf ein leeres Array festlegen und entweder den Wert für `maxHPONumberOfTrainingJobs` groß genug festlegen, um alle Stufen variabel zu machen, oder alle Hyperparameter auf Tier-1 festlegen.

Das JSON-Objekt, das jeden Hyperparameter in `1-tier-param`, `2-tier-param`, `3-tier-param` und `fixed-param` darstellt, enthält die folgenden Elemente:

- **`param`** – (Zeichenfolge) Der Name des Hyperparameters (nicht ändern).

  Siehe die Liste der gültigen Hyperparameternamen in Neptune ML.

- **`type`** – (Zeichenfolge) Der Typ des Hyperparameters (nicht ändern).

  Gültige Typen sind `bool`, `float` und `int`.

- **`default`** – (Zeichenfolge) Der Standardwert für den Hyperparameter.

  Sie können einen neuen Standardwert festlegen.

Optimierbare Hyperparameter können auch die folgenden Elemente enthalten:

- **`range`** – (Array) Der Bereich für einen kontinuierlichen optimierbaren Hyperparameter.

  Dies sollte ein Array mit zwei Werten sein, dem Minimum und dem Maximum für den Bereich (`[min, max]`).

- **`options`** – (Array) Die Optionen für einen kategorischen optimierbaren Hyperparameter.

Dieses Array sollte alle relevanten Optionen enthalten:

```
"options" : [value1, value2, ... valuen]
```

- **`inc_strategy`** – (Zeichenfolge) Der Typ der inkrementellen Änderung für kontinuierliche optimierbare Hyperparameterbereiche (nicht ändern).

  Gültige Werte sind `log`, `linear` und `power2`. Gilt nur, wenn der Bereichsschlüssel festgelegt ist.

  Wenn Sie dies ändern, wird möglicherweise nicht der gesamte Bereich des Hyperparameters für die Optimierung verwendet.

- **`inc_val`** – (Gleitkommazahl) Der Betrag, um den sich aufeinanderfolgende Inkremente für kontinuierliche optimierbare Hyperparameter unterscheiden (nicht ändern).

  Gilt nur, wenn der Bereichsschlüssel festgelegt ist.

  Wenn Sie dies ändern, wird möglicherweise nicht der gesamte Bereich des Hyperparameters für die Optimierung verwendet.

- **`node_strategy`** – (Zeichenfolge) Gibt an, dass sich der effektive Bereich für diesen Hyperparameter abhängig von der Anzahl der Knoten im Diagramm ändern soll (nicht ändern).

  Gültige Werte sind `"perM"` (pro Million), `"per10M"` (pro 10 Millionen) und `"per100M"` (pro 100 Millionen).

  Ändern Sie den Wert für `range`, statt diesen Wert zu ändern.

- **`edge_strategy`** – (Zeichenfolge) Gibt an, dass sich der effektive Bereich für diesen Hyperparameter abhängig von der Anzahl der Kanten im Diagramm ändern soll (nicht ändern).

  Gültige Werte sind `"perM"` (pro Million), `"per10M"` (pro 10 Millionen) und `"per100M"` (pro 100 Millionen).

  Ändern Sie den Wert für `range`, statt diesen Wert zu ändern.

Liste aller Hyperparameter in Neptune ML

Die folgende Liste enthält alle Hyperparameter, die in Neptune ML festgelegt werden können, für jeden Modelltyp und jede Aufgabe. Da diese Hyperparameter nicht auf alle Modelltype

anwendbar sind, ist es wichtig, dass Sie in der Datei `model-HPO-configuration.json` nur die Hyperparameter festlegen, die in der Vorlage für das von Ihnen verwendete Modell enthalten sind.

- **`batch-size`** – Die Größe des Zielknoten-Batches, der in einem einzigen Vorwärtspass verwendet wird. Typ: `int`.

  Wenn Sie dies auf einen sehr viel größeren Wert festlegen, kann es zu Arbeitsspeicherproblemen beim Trainieren auf GPU-Instances kommen.
- **`concat-node-embed`** – Gibt an, ob die ursprüngliche Darstellung eines Knotens durch die Verkettung der verarbeiteten Features mit erlernbaren ursprünglichen Knoteneinbettungen erstellt werden soll, um die Aussagekraft des Modells zu erhöhen. Typ: `bool`.
- **`dropout`** – Die Dropout-Wahrscheinlichkeit, die auf Dropout-Ebenen angewendet wird. Typ: `float`.

- **`edge-num-hidden`** – Die Größe der ausgeblendeten Ebene oder die Anzahl der Einheiten für das Kanten-Feature-Modul. Wird nur verwendet, wenn `use-edge-features` auf `True` festgelegt ist. Typ: Gleitkommazahl
- **`enable-early-stop`** – Legt fest, ob das Feature für vorzeitiges Stoppen verwendet werden soll oder nicht. Typ: `bool`. Standardwert: `true`.

  Mit diesem booleschen Parameter können Sie das Feature für vorzeitiges Stoppen deaktivieren.
- **`fanout`** – Die Anzahl der Nachbarn, die für einen Zielknoten während der Probenahme von Nachbarn untersucht werden sollen. Typ: `int`.

  Dieser Wert ist eng mit `num-layers` verbunden und sollte sich stets auf derselben Hyperparameter-Ebene befinden. Das liegt daran, dass Sie für jede potenzielle GNN-Ebene einen Fanout angeben können.

  Da dieser Hyperparameter zu einer stark schwankenden Modellleistung führen kann, sollte er fest sein oder als Tier-2- oder Tier-3-Hyperparameter festgelegt werden. Wenn Sie dies auf einen großen Wert festlegen, kann es zu Arbeitsspeicherproblemen beim Trainieren auf GPU-Instances kommen.
- **`gamma`** – Der Margenwert in der Bewertungsfunktion. Typ: `float`.

  Dies gilt nur für KGE-Modelle für die Linkvorhersage.
- **`l2norm`** – Der im Optimierer verwendete Weight-Decay-Wert, der den Gewichtungen eine L2-Normalisierungssanktion auferlegt. Typ: `bool`.

- **`layer-norm`** – Gibt an, ob die Ebenennormalisierung für `rgcn`-Modelle verwendet werden soll. Typ: `bool`.
- **`low-mem`** – Gibt an, ob auf Kosten der Geschwindigkeit eine Implementierung der Beziehungsnachrichten-Übergabefunktion mit wenig Arbeitsspeicher verwendet werden soll. Typ: `bool`.

- **`lr`** – Die Lernrate. Typ: `float`.

  Dies sollte als Tier-1-Hyperparameter festgelegt werden.
- **`neg-share`** – Gibt bei Linkvorhersagen an, ob Kanten mit positiven Proben auch negative Kantenproben teilen können. Typ: `bool`.
- **`num-bases`** – Die Anzahl der Basen für die Basiszerlegung in einem `rgcn`-Modell. Ein Wert für `num-bases`, der kleiner als die Anzahl der Kantentypen im Diagramm ist, dient zur Regularisierung des `rgcn`-Modells. Typ: `int`.
- **`num-epochs`** – Die Anzahl der Trainingsepochen, die ausgeführt werden sollen. Typ: `int`.

  Eine Epoche ist ein vollständiger Trainingsdurchgang durch das Diagramm.
- **`num-hidden`** – Die Größe der ausgeblendeten Ebene oder die Anzahl der Einheiten. Typ: `int`.

  Dies legt auch die anfängliche Einbettungsgröße für Knoten ohne Features fest.

  Wenn Sie diesen Wert auf einen viel höheren Wert setzen, ohne ihn zu reduzieren, `batch-size` kann dies zu out-of-memory Problemen beim Training auf der GPU-Instanz führen.
- **`num-layer`** – Die Anzahl der GNN-Ebenen im Modell. Typ: `int`.

  Dieser Wert ist eng mit dem Fanout-Parameter verbunden und sollte sich auf derselben Hyperparameter-Ebene wie der Fanout-Parameter befinden.

  Da dies zu einer stark schwankenden Modellleistung führen kann, sollte der Wert fest sein oder als Tier-2- oder Tier-3-Hyperparameter festgelegt werden.
- **`num-negs`** – Bei Linkvorhersagen ist dies die Anzahl der negativen Proben pro positiver Probe. Typ: `int`.
- **`per-feat-name-embed`** – Gibt an, ob jedes Feature eingebettet werden soll, indem es unabhängig transformiert wird, bevor Features kombiniert werden. Typ: `bool`.

Bei Festlegung auf `true` wird jedes Feature pro Knoten unabhängig in eine feste Dimensionsgröße transformiert, bevor alle transformierten Features für den Knoten verkettet und weiter in die Dimension `num_hidden` transformiert werden.

Bei Festlegung auf `false` werden die Features ohne Feature-spezifische Transformationen verkettet.

- **`regularization-coef`** – Bei Linkvorhersagen ist dies der Koeffizient des Regularisierungsverlusts. Typ: `float`.
- **`rel-part`** – Gibt an, ob für `KGE`-Linkvorhersagen Beziehungspartitionen verwendet werden sollen. Typ: `bool`.
- **`sparse-lr`** – Die Lernrate für Einbettungen erlernbarer Knoten. Typ: `float`.

  Erlernbare anfängliche Knoteneinbettungen werden für Knoten ohne Features oder bei Festlegung von `concat-node-embed` verwendet. Die Parameter der Sparse-Einbettungsebene für erlernbare Knoten werden mithilfe eines getrennten Optimierers trainiert, der eine eigene Lernrate haben kann.
- **`use-class-weight`** – Gibt an, ob auf unausgewogene Klassifizierungsaufgaben Klassengewichtungen angewendet werden sollen. Bei Festlegung auf `true`, werden die Bezeichnungszahlen verwendet, um für jede Klassenbezeichnung eine Gewichtung festzulegen. Typ: `bool`.
- **`use-edge-features`** – Gibt an, ob während der Nachrichtenübergabe Kanten-Features verwendet werden sollen. Bei Festlegung auf `true` wird der RGCN-Ebene ein benutzerdefiniertes Kanten-Feature-Modul für Kantentypen mit Features hinzugefügt. Typ: `bool`.
- **`use-self-loop`** – Gibt an, ob Selbstschleifen in das Training eines `rgcn`-Modells einbezogen werden sollen. Typ: `bool`.
- **`window-for-early-stop`** – Vergleicht die Anzahl der letzten Validierungsergebnisse mit dem Durchschnitt, um über einen vorzeitigen Stopp zu entscheiden. Der Standardtyp ist 3. type=int. Siehe auch Vorzeitiges Stoppen des Modelltrainings in Neptune ML. Typ: `int`. Standardwert: `3`.

  Siehe .

Anpassen von Hyperparametern in Neptune ML

Wenn Sie die Datei `model-HPO-configuration.json` bearbeiten, sind dies die häufigsten Arten von Änderungen:

- Bearbeiten Sie die Mindest- und and/or Höchstwerte der `range` Hyperparameter.
- Legen Sie einen Hyperparameter auf einen festen Wert fest, indem Sie ihn zum Abschnitt `fixed-param` verschieben und den Standardwert auf den festen Wert festlegen, den er annehmen soll.
- Ändern Sie die Priorität eines Hyperparameters, indem Sie ihn in einer bestimmten Ebene platzieren, den Bereich bearbeiten und sicherstellen, dass der Standardwert korrekt festgelegt ist.

Bewährte Modelltrainingsmethoden

Sie können die Leistung von Neptune ML-Modellen auf verschiedene Arten verbessern.

Auswahl der richtigen Knoteneigenschaft

Möglicherweise sind nicht alle Eigenschaften im Diagramm für Ihre Machine-Learning-Aufgaben sinnvoll oder relevant. Irrelevante Eigenschaften sollten vom Datenexport ausgeschlossen werden.

Dies sind einige bewährte Methoden:

- Lassen Sie sich von Fachexperten helfen, die Bedeutung von Features und die Möglichkeit ihrer Verwendung für Vorhersagen zu bewerten.
- Entfernen Sie die Features, die Sie als überflüssig oder irrelevant betrachten, um das Datenrauschen und die Zahl der nicht wichtigen Korrelationen zu reduzieren.
- Führen Sie während der Erstellung des Modells Iterationen durch. Passen Sie Features, Feature-Kombinationen und Optimierungsziele über die Zeit an.

im Abschnitt Feature-Verarbeitung im Amazon-Machine-Learning-Entwicklerhandbuch enthält weitere Anleitungen für die Feature-Verarbeitung, die für Neptune ML relevant sind.

Behandlung von Ausreißer-Datenpunkten

Ein Ausreißer ist ein Datenpunkt, der sich erheblich von den übrigen Daten unterscheidet. Datenausreißer können den Trainingsvorgang korrumpieren oder fehlleiten, was zu längeren Trainingszeiten oder weniger genauen Modellen führt. Wenn sie nicht wirklich wichtig sind, sollten Sie Ausreißer vor dem Exportieren der Daten entfernen.

Entfernen duplizierter Knoten und Kanten

In Neptune gespeicherte Diagramme können duplizierte Knoten oder Kanten enthalten. Diese redundanten Elemente führen beim Trainieren für ML-Modelle zu Rauschen. Entfernen Sie duplizierte Knoten oder Kanten, bevor Sie die Daten exportieren.

Optimieren der Diagrammstruktur

Wenn das Diagramm exportiert wird, können Sie die Verarbeitung von Features und die Konstruktion des Diagramms ändern, um die Modellleistung zu verbessern.

Dies sind einige bewährte Methoden:

- Wenn eine Kanteneigenschaft die Bedeutung von Kantenkategorien hat, lohnt es sich in einigen Fällen, sie in Kantentypen zu konvertieren.
- Die Standard-Normalisierungsrichtlinie für eine numerische Eigenschaft ist `min-max`. In einigen Fällen funktionieren andere Normalisierungsrichtlinien jedoch besser. Sie können die Eigenschaft vorab verarbeiten und die Normalisierungsrichtlinie wie in Elemente einer `model-HPO-configuration.json`-Datei beschrieben.
- Beim Exportvorgang werden automatisch Feature-Typen basierend auf Eigenschaftstypen generiert. Beispielsweise werden `String`-Eigenschaften als kategorische Features und `Float`- und `Int`-Eigenschaften als numerische Features behandelt. Wenn notwendig, können Sie den Feature-Typ nach dem Export ändern (siehe Elemente einer `model-HPO-configuration.json`-Datei).

Optimieren von Hyperparameter-Bereichen und -Standardwerten

Die Datenverarbeitungsoperation leitet Hyperparameter-Konfigurationsbereiche aus dem Diagramm ab. Wenn die generierten Hyperparameterbereiche und Standardwerte des Modells für Ihre Diagrammdaten nicht gut funktionieren, können Sie die HPO-Konfigurationsdatei bearbeiten, um Ihre eigene Hyperparameter-Optimierungsstrategie zu erstellen.

Dies sind einige bewährte Methoden:

- Wenn das Diagramm groß wird, reicht die Standardgröße ausgeblendeter Dimensionen möglicherweise nicht aus, um alle Informationen aufzunehmen. Sie können den Hyperparameter `num-hidden` ändern, um die Größe ausgeblendeter Dimensionen zu steuern.
- Bei Knowledge Graph Embedding (KGE)-Modellen sollten Sie vielleicht das spezifische verwendete Modell auf der Basis von Diagrammstruktur und Budget ändern.

  `TrainsE`Modelle haben Schwierigkeiten, mit one-to-many (1-N), many-to-one (N-1) und many-to-many (N-N) Beziehungen umzugehen. `DistMult`Modelle haben Schwierigkeiten, mit symmetrischen Beziehungen umzugehen. `RotatE`ist gut darin, alle Arten von Beziehungen zu modellieren, ist aber teurer als `TrainsE` und `DistMult` während des Trainings.
- In einigen Fällen, in denen sowohl die Knoten-ID als auch Informationen zu Knoten-Features wichtig sind, sollten Sie das Neptune-ML-Modell mit `` `concat-node-embed` `` anweisen, die anfängliche Darstellung eines Knotens durch die Verkettung seiner Features mit den anfänglichen Einbettungen abzurufen.

- Wenn Sie für einige Hyperparameter eine angemessene Leistung erzielen, können Sie die Hyperparameter-Suchumgebung entsprechend diesen Ergebnissen anpassen.

Vorzeitiges Stoppen des Modelltrainings in Neptune ML

Ein vorzeitiger Stopp kann die Ausführungszeit und die Kosten für das Modelltraining deutlich reduzieren, ohne die Modellleistung zu beeinträchtigen. Dies verhindert auch, dass sich das Modell zu stark an die Trainingsdaten anpasst.

Ein vorzeitiger Abbruch ist von regelmäßigen Messungen der Validierungssatzleistung abhängig. Zunächst wird die Leistung mit fortschreitendem Training besser. Wenn sich das Modell jedoch zu stark anpasst, nimmt sie wieder ab. Das Feature für vorzeitige Stopps identifiziert den Punkt, ab dem sich ein Modell zu sehr anpasst, und stoppt das Modelltraining an diesem Punkt.

Neptune ML überwacht die Validierungsmetrikaufrufe und vergleicht die neueste Validierungsmetrik mit dem Durchschnitt der Validierungsmetriken der letzten **n** Auswertungen, wobei **n** ein Zahlensatz ist, der den Parameter `window-for-early-stop` verwendet. Sobald die Validierungsmetrik schlechter als dieser Durchschnitt ist, stoppt Neptune ML das Modelltraining und speichert das bisher beste Modell.

Sie können das vorzeitige Stoppen wie folgt mit den folgenden Parametern steuern:

- **`window-for-early-stop`** – Der Wert dieses Parameters ist eine Ganzzahl und gibt die Zahl der letzten Validierungsergebnisse an, die für die Berechnung des Durchschnitts verwendet werden sollen, um über einen vorzeitigen Stopp zu entscheiden. Der Standardwert ist `3`.
- **`enable-early-stop`** – Mit diesem booleschen Parameter können Sie das Feature für vorzeitiges Stoppen deaktivieren. Der Standardwert ist `true`.

Vorzeitiges Stoppen des HPO-Vorgangs in Neptune ML

Die Early-Stop-Funktion in Neptune ML stoppt mithilfe der SageMaker AI HPO-Warmstartfunktion auch Trainingsjobs, die im Vergleich zu anderen Trainingsjobs nicht gut abschneiden. Auch dies kann die Kosten senken und die HPO-Qualität verbessern.

In Ausführen eines Warmstart-Optimierungsaufgabe für Hyperparameter finden Sie eine Beschreibung der Funktionsweise.

Der Warmstart ermöglicht die Übergabe von Informationen aus früheren Trainingsaufträgen an nachfolgende Trainingsaufträge und hat zwei entscheidende Vorteile:

- Erstens werden die Ergebnisse vorheriger Trainingsaufträge für die Auswahl guter Kombinationen von Hyperparametern verwendet, um sie im neuen Optimierungsauftrag zu durchsuchen.
- Zweitens ermöglicht der Warmstart vorzeitige Stopps, um auf weitere Modellausführungen zugreifen zu können, was die Optimierungszeit reduziert.

Dieses Feature ist in Neptune ML automatisch aktiviert und ermöglicht Ihnen, ein Gleichgewicht zwischen Trainingszeit und Leistung herzustellen. Wenn Sie mit der Leistung des aktuellen Modells zufrieden sind, können Sie dieses Modell verwenden. Andernfalls laufen Sie mehr, HPOs die mit den Ergebnissen früherer Läufe warmgestartet wurden, um ein besseres Modell zu finden.

Professionelle Support-Services

AWS bietet professionelle Support-Services, um Ihnen bei Problemen bei Ihren Projekten im Bereich maschinelles Lernen auf Neptune zu helfen. Wenn Sie nicht weiterkommen, nehmen Sie bitte Kontakt mit dem AWS -Support auf.

Verwenden trainierter Modelle zur Generierung neuer Modellartefakte

Mit dem Neptune-ML-Befehl für die Modelltransformation können Sie mit vorab trainierten Modellparametern Modellartefakte wie Knoteneinbettungen für verarbeitete Diagrammdaten berechnen.

Modelltransformation für inkrementelle Inferenzen

Im inkrementellen Model-Inferenz-Workflow können Sie, nachdem Sie die aktualisierten Grafikdaten, die Sie aus Neptune exportiert haben, verarbeitet haben, einen Modelltransformationsjob mit einem Befehl wie dem folgenden starten:

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/"
```

Weitere Informationen finden Sie unter start-ml-model-transform-job in der Befehlsreferenz. 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.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**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

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/"
      }'
```

Anschließend können Sie die ID dieses Auftrags an den API-Aufruf create-endpoints übergeben, um einen neuen Endpunkt zu erstellen oder einen vorhandenen Endpunkt mit den neuen Modellartefakten zu aktualisieren, die von diesem Auftrag generiert wurden. So kann der neue oder aktualisierte Endpunkt Modellvorhersagen für die aktualisierten Diagrammdaten bereitstellen.

Modelltransformation für Trainingsaufträge

Sie können auch einen `trainingJobName` Parameter angeben, um Modellartefakte für alle SageMaker KI-Trainingsjobs zu generieren, die während des Neptune ML-Modelltrainings gestartet wurden. Da ein Trainingsjob im Neptune-ML-Modell potenziell viele SageMaker KI-Schulungsjobs starten kann, haben Sie die Flexibilität, einen Inferenzendpunkt zu erstellen, der auf einem dieser SageMaker KI-Schulungsjobs basiert.

Beispiel:

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/"
```

Weitere Informationen finden Sie unter start-ml-model-transform-job in der Befehlsreferenz. 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.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**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

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/"
      }'
```

Wenn der ursprüngliche Trainingsauftrag für ein vom Benutzer bereitgestelltes benutzerdefiniertes Modell bestimmt war, müssen Sie beim Aufrufen einer Modelltransformation ein

`customModelTransformParameters`-Objekt einfügen. Informationen zur Implementierung eines benutzerdefinierten Modells finden Sie unter Benutzerdefinierte Modelle in Neptune ML.

> **Note**
>
> Der `modeltransform` Befehl führt die Modelltransformation immer für den besten SageMaker KI-Trainingsjob für dieses Training aus.

Weitere Informationen zu Modelltransformationsaufträgen finden Sie unter Der Befehl modeltransform.

Artefakte, die von Modelltrainings in Neptune ML erzeugt wurden

Nach dem Modelltraining verwendet Neptune ML die am besten trainierten Modellparameter, um Modellartefakte zu generieren, die für den Start des Inferenzendpunkts und die Bereitstellung von Modellvorhersagen notwendig sind. Diese Artefakte werden für den Trainingsjob zusammengefasst und im Amazon S3-Ausgabespeicherort des besten SageMaker KI-Schulungsjobs gespeichert.

Die folgenden Abschnitte beschreiben, was in den Modellartefakten für die verschiedenen Aufgaben enthalten ist, und wie vorhandene, bereits trainierte Modelle vom Modelltransformationsbefehl verwendet werden, um auch für neue Diagrammdaten Artefakte zu erzeugen.

Artefakte, die für verschiedene Aufgaben generiert werden

Der Inhalt der vom Training generierten Modellartefakte ist von der Machine-Learning-Zielaufgabe abhängig:

- Klassifizierung und Regression von Knoten – Für Vorhersagen von Knoteneigenschaften gehören zu den Artefakten Modellparameter, Knoteneinbettungen aus dem GNN-Encoder, Modellvorhersagen für Knoten im Trainingsdiagramm und einige Konfigurationsdateien für den Inferenzendpunkt. Bei Aufgaben für die Klassifizierung und Regression von Aufgaben werden zur Reduzierung der Abfragelatenz Modellvorhersagen für Knoten, die während des Trainings vorhanden sind, vorab berechnet.
- Klassifizierung und Regression von Kanten – Für Vorhersagen von Kanteneigenschaften gehören zu den Artefakten auch Modellparameter und Knoteneinbettungen. Die Parameter des Modell-Decoders sind für die Inferenz besonders wichtig, da die Vorhersagen für die Klassifizierung oder Regression von Kanten durch die Anwendung des Modell-Decoders auf die Einbettungen des Quell- und Zieleckpunkts einer Kante berechnet werden.
- Linkvorhersage – Für Linkvorhersagen ist neben den Artefakten, die für die Vorhersage der Kanteneigenschaften generiert werden, auch das DGL-Diagramm als Artefakt enthalten, da Linkvorhersagen die Ausführung von Vorhersagen durch das Trainingsdiagramm erfordern. Das Ziel von Linkvorhersagen besteht in der Vorhersage der Zieleckpunkte, die wahrscheinlich mit einem Quelleckpunkt kombiniert werden, um eine Kante eines bestimmten Typs im Diagramm zu bilden. Hierzu werden die Knoteneinbettung des Quelleckpunkts und eine erlernte Darstellung des Kantentyps mit den Knoteneinbettungen aller möglichen Zieleckpunkte kombiniert, sodass für jeden Zieleckpunkt ein Wert für die Kantenwahrscheinlichkeit berechnet wird. Die Werte werden anschließend sortiert, um die potenziellen Zieleckpunkte zu ordnen und die besten Kandidaten zurückzugeben.

Für jeden Aufgabentyp werden die Gewichtungen des Graph-Neural-Network-Modells aus der DGL im Modellartefakt gespeichert. Auf diese Weise kann Neptune ML neue Modellausgaben berechnen, wenn das Diagramm geändert wird (induktive Inferenz), und vorab berechnete Vorhersagen und Einbettungen (transduktive Inferenz) verwenden, um die Latenz zu reduzieren.

Generieren neuer Modellartefakte

Die nach dem Modelltraining in Neptune ML generierten Modellartefakte sind direkt mit dem Trainingsvorgang verbunden. Die vorab berechneten Einbettungen und Vorhersagen sind daher nur für Entitäten im ursprünglichen Trainingsdiagramm vorhanden. Obwohl der induktive Inferenzmodus für Neptune-ML-Endpunkte Vorhersagen für neue Entitäten in Echtzeit berechnen kann, sollten Sie Batch-Vorhersagen für neue Entitäten ohne Abfrage eines Endpunkts generieren.

Um Batch-Modellvorhersagen für neue Entitäten zu erhalten, die dem Diagramm hinzugefügt wurden, müssen für die neuen Diagrammdaten neue Modellartefakte neu berechnet werden. Dies erfolgt mittels des Befehls `modeltransform`. Sie verwenden den Befehl `modeltransform`, wenn Sie nur Batch-Vorhersagen ohne Einrichtung eines Endpunkts erhalten möchten oder wenn alle Vorhersagen generiert werden sollen, damit Sie diese zurück zum Diagramm schreiben können.

Da Modelltrainings am Ende des Trainingsvorgangs implizit eine Modelltransformation durchführen, führen Trainingsaufträge stets eine Neuberechnung der Modellartefakte anhand der Trainingsdiagrammdaten durch. Der Befehl `modeltransform` kann Modellartefakte jedoch auch anhand von Diagrammdaten berechnen, die nicht zum Trainieren eines Modells verwendet wurden. Hierzu müssen die neuen Diagrammdaten mit denselben Feature-Kodierungen wie die ursprünglichen Diagrammdaten verarbeitet werden und dasselbe Diagrammschema verwenden.

Erstellen Sie zunächst einen Klon des Datenverarbeitungsauftrags, der für die ursprünglichen Trainingsdiagrammdaten ausgeführt wurde. Führen Sie diesen dann für die neuen Diagrammdaten aus (siehe Verarbeiten aktualisierter Diagrammdaten für Neptune ML). Rufen Sie dann den Befehl `modeltransform` mit der neuen `dataProcessingJobId` und der alten `modelTrainingJobId` auf, um die Modellartefakte anhand der aktualisierten Diagrammdaten neu zu berechnen.

Für die Vorhersage von Knoteneigenschaften werden die Knoteneinbettungen und Vorhersagen anhand der neuen Diagrammdaten neu berechnet, auch für Knoten im ursprünglichen Trainingsdiagramm.

Für die Vorhersage von Kanteneigenschaften und Linkvorhersagen werden die Knoteneinbettungen ebenfalls neu berechnet. Diese überschreiben alle vorhandenen Knoteneinbettungen. Um die

Knoteneinbettungen neu zu berechnen, wendet Neptune ML den erlernten GNN-Encoder aus dem zuvor trainierten Modell auf die Knoten der neuen Diagrammdaten mit ihren neuen Features an.

Für Knoten ohne Features werden die erlernten anfänglichen Darstellungen aus dem ursprünglichen Modelltraining wiederverwendet. Für neue Knoten ohne Features, die nicht im ursprünglichen Trainingsdiagramm vorhanden waren, initialisiert Neptune ML die Darstellung als Durchschnitt der erlernten anfänglichen Knotendarstellungen des Knotentyps im ursprünglichen Trainingsdiagramm. Dies kann zu Leistungsabfällen bei Modellvorhersagen führen, wenn es zahlreiche neue Knoten ohne Features gibt, da alle zur durchschnittlichen anfänglichen Einbettung für diesen Knotentyp initialisiert werden.

Wenn Ihr Modell mit `concat-node-embed` als „true“ trainiert wird, werden die anfänglichen Knotendarstellungen durch die Verkettung der Knoten-Features mit der erlernbaren Anfangsdarstellung erstellt. Daher verwendet die anfängliche Knotendarstellung neuer Knoten für das aktualisierte Diagramm ebenfalls die durchschnittlichen anfänglichen Knoteneinbettungen, verkettet mit neuen Knoten-Features.

Darüber hinaus wird das Löschen von Knoten zurzeit nicht unterstützt. Wenn Knoten aus dem aktualisierten Diagramm entfernt wurden, müssen Sie das Modell anhand der aktualisierten Diagrammdaten neu trainieren.

Bei der Neuberechnung der Modellartefakte werden die erlernten Modellparameter für ein neues Diagramm wiederverwendet. Dies sollte nur erfolgen, wenn das neue Diagramm dem alten Diagramm sehr ähnlich ist. Wenn das neue Diagramm nicht ausreichend ähnlich ist, müssen Sie das Modell neu trainieren, um eine vergleichbare Modellleistung mit den neuen Diagrammdaten zu erzielen. Was als ausreichend ähnlich gilt, ist von der Struktur der Diagrammdaten abhängig. Als Faustregel sollten Sie Ihr Modell jedoch neu trainieren, wenn sich Ihre neuen Daten um mehr als 10–20 % von den ursprünglichen Trainingsdiagrammdaten unterscheiden.

Für Diagramme, in denen alle Knoten Features besitzen, gilt das obere Ende des Schwellenwerts (20 % Unterschied). Für Diagramme, in denen viele Knoten keine Features besitzen und die neuen, dem Diagramm hinzugefügten Knoten keine Eigenschaften besitzen, ist der untere Schwellenwert (10 % Unterschied) möglicherweise sogar zu hoch.

Weitere Informationen zu Modelltransformationsaufträgen finden Sie unter Der Befehl modeltransform.

Benutzerdefinierte Modelle in Neptune ML

> **Note**
>
> Die Unterstützung benutzerdefinierter Neptune ML-Modelle basiert auf einer älteren Version von Python 3. Um benutzerdefinierte GNN-Modelle mit up-to-date Abhängigkeiten zu erstellen und auszuführen, verwenden Sie on. GraphStorm SageMaker Induktive Inferenz in Echtzeit wird zurzeit für benutzerdefinierte Modelle nicht unterstützt.

In Neptune ML können Sie mithilfe von Python Ihre eigenen benutzerdefinierten Modellimplementierungen definieren. Sie können in der Neptune-ML-Infrastruktur benutzerdefinierte Modelle wie die integrierten Modelle trainieren, bereitstellen und verwenden, um mittels Diagrammabfragen Vorhersagen zu erhalten.

Sie können mit der Implementierung eines eigenen benutzerdefinierten Modells in Python beginnen, indem Sie den Beispielen im Neptune-ML-Toolkit-Beispielen folgen und die im Neptune-ML-Toolkit bereitgestellten Modellkomponenten verwenden. In den folgenden Abschnitten finden Sie weitere Details.

Inhalt

- Übersicht über benutzerdefinierte Modelle in Neptune ML
  - Wann sollte ein benutzerdefiniertes Modell in Neptune ML verwendet werden
  - Workflow für die Entwicklung und Verwendung eines benutzerdefinierten Modells in Neptune ML
- Entwicklung kundenspezifischer Modelle in Neptune ML
  - Entwicklung benutzerdefinierter Modelltrainingsskripts in Neptune ML
  - Entwicklung benutzerdefinierter Modelltransformationsskripts in Neptune ML
  - Benutzerdefinierte model-hpo-configuration.json-Datei in Neptune ML
  - Lokale Tests Ihrer benutzerdefinierten Modellimplementierung in Neptune ML

Übersicht über benutzerdefinierte Modelle in Neptune ML

Wann sollte ein benutzerdefiniertes Modell in Neptune ML verwendet werden

Die integrierten Modelle von Neptune ML behandeln alle von Neptune ML unterstützten Standardaufgaben. Es kann jedoch Fälle geben, in denen Sie das Modell für eine bestimmte Aufgabe genauer steuern möchten oder den Modelltrainingsprozess anpassen müssen. Sie sollten in den folgenden Situationen ein benutzerdefiniertes Modell verwenden:

- Die Feature-Kodierung für Text-Features sehr großer Textmodelle muss auf einer GPU ausgeführt werden.
- Sie möchten ein eigenes benutzerdefiniertes Graph-Neural-Network (GNN)-Modell verwenden, das in der Deep Graph Library (DGL) entwickelt wurde.
- Sie möchten tabellarische Modelle oder Ensemblemodelle für die Klassifizierung und Regression von Knoten verwenden.

Workflow für die Entwicklung und Verwendung eines benutzerdefinierten Modells in Neptune ML

Die Unterstützung benutzerdefinierter Modelle in Neptune ML soll die nahtlose Integration in vorhandene Neptune-ML-Workflows ermöglichen. Dabei wird benutzerdefinierter Code in Ihrem Quellmodul in der Neptune-ML-Infrastruktur ausgeführt, um das Modell zu trainieren. Genau wie bei einem integrierten Modus startet Neptune ML automatisch einen SageMaker HyperParameter KI-Tuning-Job und wählt anhand der Bewertungsmetrik das beste Modell aus. Anschließend werden über die in Ihrem Quellmodul bereitgestellte Implementierung Modellartefakte zur Bereitstellung generiert.

Datenexport, Trainingskonfiguration und Datenvorverarbeitung sind für benutzerdefinierte und integrierte Modelle identisch.

Nach der Datenvorverarbeitung können Sie Ihre benutzerdefinierte Modellimplementierung iterativ und interaktiv mit Python entwickeln und testen. Wenn Ihr Modell für die Produktion bereit ist, können Sie das resultierende Python-Modul wie folgt zu Amazon S3 hochladen:

```
aws s3 cp --recursive (source path to module) s3://(bucket name)/(destination path for
 your module)
```

Anschließend können Sie das Modell über den normalen Standard-Daten-Workflow oder den inkrementellen Daten-Workflow in der Produktion bereitstellen. Es gibt jedoch einige Unterschiede.

Für das Modelltraining anhand eines benutzerdefinierten Modells müssen Sie der Neptune-ML-Modelltrainings-API das JSON-Objekt `customModelTrainingParameters` bereitstellen, um sicherzustellen, dass Ihr benutzerdefinierter Code verwendet wird. Die Felder im Objekt `customModelTrainingParameters` sind:

- **`sourceS3DirectoryPath`** – (Erforderlich) Der Pfad zum Amazon-S3-Speicherort des Python-Moduls, das Ihr Modell implementiert. Dieser Pfad muss auf einen gültigen, vorhandenen Amazon-S3-Speicherort verweisen, der mindestens ein Trainingsskript, ein Transformationsskript und die Datei `model-hpo-configuration.json` enthält.
- **`trainingEntryPointScript`** – (Optional) Der Name des Einstiegspunkts in Ihrem Modul für ein Skript, das Modelltrainings durchführt und Hyperparameter als Befehlszeilenargumente verwendet, einschließlich fester Hyperparameter.

  Standardwert: `training.py`.
- **`transformEntryPointScript`** – (Optional) Der Name des Einstiegspunkts in Ihrem Modul für ein Skript, das ausgeführt werden soll, nachdem das beste Modell aus der Hyperparametersuche identifiziert wurde, um die für die Modellbereitstellung notwendigen Modellartefakte zu berechnen. Es sollte ohne Befehlszeilenargumente ausgeführt werden können.

  Standardwert: `transform.py`.

Beispiel:

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)"
  }'
```

Weitere Informationen finden Sie unter start-ml-model-training-job in der AWS CLI Befehlsreferenz.

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**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

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)"
        }
```

```
    }'
```

Um eine benutzerdefinierte Modelltransformation zu aktivieren, müssen Sie der Neptune-ML-Modelltransformations-API das JSON-Objekt `customModelTransformParameters` mit Feldwerten bereitstellen, die mit den gespeicherten Modellparametern aus dem Trainingsauftrag kompatibel sind. Das Objekt `customModelTransformParameters` enthält die folgenden Felder:

- **`sourceS3DirectoryPath`** – (Erforderlich) Der Pfad zum Amazon-S3-Speicherort des Python-Moduls, das Ihr Modell implementiert. Dieser Pfad muss auf einen gültigen, vorhandenen Amazon-S3-Speicherort verweisen, der mindestens ein Trainingsskript, ein Transformationsskript und die Datei `model-hpo-configuration.json` enthält.
- **`transformEntryPointScript`** – (Optional) Der Name des Einstiegspunkts in Ihrem Modul für ein Skript, das ausgeführt werden soll, nachdem das beste Modell aus der Hyperparametersuche identifiziert wurde, um die für die Modellbereitstellung notwendigen Modellartefakte zu berechnen. Es sollte ohne Befehlszeilenargumente ausgeführt werden können.

  Standardwert: `transform.py`.

Beispiel:

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)"
  }'
```

Weitere Informationen finden Sie unter start-ml-model-transform-job in der AWS CLI Befehlsreferenz.

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**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

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)"
        }
      }'
```

Entwicklung kundenspezifischer Modelle in Neptune ML

Eine gute Möglichkeit, mit der Entwicklung benutzerdefinierter Modelle zu beginnen, besteht darin, den Beispielen im Neptune-ML-Toolkit zu folgen, um Ihr Trainingsmodul zu strukturieren und zu schreiben. Das Neptune-ML-Toolkit implementiert auch modularisierte Diagramm-ML-Modellkomponenten im modelzoo, die Sie stapeln und zur Erstellung Ihres benutzerdefinierten Modells verwenden können.

Darüber hinaus bietet das Toolkit Dienstprogrammfunktionen, mit denen Sie während Modelltraining und Modelltransformation die notwendigen Artefakte erzeugen können. Sie können dieses Python-Paket in Ihre benutzerdefinierte Implementierung importieren. Alle im Toolkit enthaltenen Funktionen oder Module sind auch in der Neptune-ML-Trainingsumgebung verfügbar.

Wenn Ihr Python-Modul zusätzliche externe Abhängigkeiten besitzt, können Sie diese zusätzlichen Abhängigkeiten einfügen, indem Sie im Verzeichnis Ihres Moduls die Datei `requirements.txt` erstellen. Die in der Datei `requirements.txt` aufgelisteten Pakete werden anschließend installiert, bevor Ihr Trainingsskript ausgeführt wird.

Das Python-Modul, das Ihr benutzerdefiniertes Modell implementiert, muss mindestens Folgendes enthalten:

- Einen Einstiegspunkt für ein Trainingsskript
- Einen Einstiegspunkt für ein Transformationsskript
- Eine `model-hpo-configuration.json`-Datei

Entwicklung benutzerdefinierter Modelltrainingsskripts in Neptune ML

Ihr benutzerdefiniertes Modelltrainingsskript sollte ein ausführbares Python-Skript sein, siehe das Beispielskript `train.py` im Neptune-ML-Toolkit. Es muss Hyperparameternamen und -werte als Befehlszeilenargumente akzeptieren. Während des Modelltrainings werden die Hyperparameternamen aus der Datei `model-hpo-configuration.json` abgerufen. Die Hyperparameterwerte liegen entweder innerhalb des gültigen Hyperparameterbereichs, wenn der Hyperparameter optimierbar ist, oder nehmen den Standard-Hyperparameterwert an, wenn er nicht optimierbar ist.

Ihr Trainingsskript wird auf einer SageMaker AI-Trainingsinstanz mit einer Syntax wie der folgenden ausgeführt:

```
python3 (script entry point) --(1st parameter) (1st value) --(2nd parameter) (2nd
 value) (...)
```

Für alle Aufgaben AutoTrainer sendet das Neptune ML zusätzlich zu den von Ihnen angegebenen Hyperparametern mehrere erforderliche Parameter an Ihr Trainingsskript, und Ihr Skript muss in der Lage sein, diese zusätzlichen Parameter zu verarbeiten, um ordnungsgemäß zu funktionieren.

Diese zusätzlichen erforderlichen Parameter sind von der jeweiligen Aufgabe abhängig:

Für die Klassifizierung oder Regression von Knoten

- **`task`** – Der intern von Neptune ML verwendete Aufgabentyp. Für die Knotenklassifikation ist dies `node_class`, für die Knotenregression `node_regression`.
- **`model`** – Der intern von Neptune ML verwendete Modellname, in diesem Fall `custom`.
- **`name`** – Der Name der intern von Neptune ML verwendeten Aufgabe, in diesem Fall `node_class-custom` für die Knotenklassifizierung und `node_regression-custom` für die Knotenregression.
- **`target_ntype`** – Der Name des Knotentyps für die Klassifizierung oder Regression.
- **`property`** – Der Name der Knoteneigenschaft für die Klassifizierung oder Regression.

Für Linkvorhersagen

- **`task`** – Der intern von Neptune ML verwendete Aufgabentyp. Für Linkvorhersagen ist dies `link_predict`.
- **`model`** – Der intern von Neptune ML verwendete Modellname, in diesem Fall `custom`.
- **`name`** – Der intern von Neptune ML verwendete Aufgabenname, in diesem Fall `link_predict-custom`.

Für die Klassifizierung oder Regression von Kanten

- **`task`** – Der intern von Neptune ML verwendete Aufgabentyp. Für die Kantenklassifikation ist dies `edge_class`, für die Kantenregression `edge_regression`.
- **`model`** – Der intern von Neptune ML verwendete Modellname, in diesem Fall `custom`.
- **`name`** – Der Name der intern von Neptune ML verwendeten Aufgabe, in diesem Fall `edge_class-custom` für die Kantenklassifizierung und `edge_regression-custom` für die Kantenregression.

- **`target_etype`** – Der Name des Kantentyps für die Klassifizierung oder Regression.
- **`property`** – Der Name der Kanteneigenschaft für die Klassifizierung oder Regression.

Ihr Skript sollte die Modellparameter sowie alle weiteren Artefakte speichern, die am Ende des Trainings benötigt werden.

Sie können die Dienstprogrammfunktionen des Neptune-ML-Toolkits verwenden, um den Speicherort der verarbeiteten Diagrammdaten, den Speicherort der Modellparameter und die auf der Trainings-Instance verfügbaren GPU-Geräte zu bestimmen. Beispiele für die Verwendung dieser Dienstprogrammfunktionen finden Sie im Beispiel-Trainingsskript train.py.

Entwicklung benutzerdefinierter Modelltransformationsskripts in Neptune ML

Ein Transformationsskript ist erforderlich, um den inkrementellen Neptune-ML-Workflow für Modellinferenzen für neue Diagramme zu nutzen, ohne das Modell neu zu trainieren. Auch wenn alle für die Modellbereitstellung notwendigen Artefakte durch das Trainingsskript generiert werden, müssen Sie dennoch ein Transformationsskript bereitstellen, wenn Sie aktualisierte Modelle generieren möchten, ohne das Modell neu zu trainieren.

> **Note**
>
> Induktive Inferenz in Echtzeit wird zurzeit für benutzerdefinierte Modelle nicht unterstützt.

Ihr benutzerdefiniertes Modelltransformationsskript sollte ein ausführbares Python-Skript sein, siehe das Beispielskript transform.py im Neptune-ML-Toolkit. Da dieses Skript während des Modelltrainings ohne Befehlszeilenargumente aufgerufen wird, müssen alle Befehlszeilenargumente, die das Skript akzeptiert, Standardwerte haben.

Das Skript läuft auf einer SageMaker KI-Trainingsinstanz mit einer Syntax wie dieser:

```
python3 (your transform script entry point)
```

Ihr Transformationsskript benötigt verschiedene Informationen, z. B.:

- Der Speicherort der verarbeiteten Diagrammdaten.
- Der Speicherort der Modellparameter und der Speicherort für neue Modellartefakte.
- Die auf der Instance verfügbaren Geräte.

- Die Hyperparameter, die das beste Modell generiert haben.

Diese Eingaben werden über Neptune-ML-Dienstprogrammfunktionen abgerufen, die Ihr Skript aufrufen kann. Beispiele hierfür finden Sie im Beispielskript transform.py im Toolkit.

Das Skript sollte Knoteneinbettungen, Knoten-ID-Zuordnungen und alle anderen Artefakte speichern, die für die Modellbereitstellung für die einzelnen Aufgaben notwendig sind. Weitere Informationen zu den für verschiedene Neptune-ML-Aufgaben erforderlichen Modellartefakten finden Sie in der Dokumentation zu Modellartefakten.

Benutzerdefinierte **`model-hpo-configuration.json`**-Datei in Neptune ML

Die Datei `model-hpo-configuration.json` definiert Hyperparameter für Ihr benutzerdefiniertes Modell. Sie hat dasselbe Format wie die Datei `model-hpo-configuration.json`, die mit den integrierten Neptune-ML-Modellen verwendet wird, und hat Priorität gegenüber der Version, die von Neptune ML automatisch generiert und zum Speicherort Ihrer verarbeiteten Daten hochgeladen wird.

Wenn Sie Ihrem Modell einen neuen Hyperparameter hinzufügen, müssen Sie in dieser Datei auch einen Eintrag für den Hyperparameter hinzufügen, damit dieser an Ihr Trainingsskript übergeben wird.

Sie müssen einen Bereich für einen Hyperparameter angeben, wenn er optimierbar sein soll, und diesen als `tier-1`-, `tier-2`- oder `tier-3`-Parameter festlegen. Der Hyperparameter wird optimiert, wenn die Gesamtzahl der konfigurierten Trainingsaufträge die Optimierung von Hyperparametern in seiner Stufe zulässt. Für einen nicht optimierbaren Parameter müssen Sie einen Standardwert angeben und den Hyperparameter zum Abschnitt `fixed-param` der Datei hinzufügen. Ein Beispiel dafür, wie Sie dies tun, finden Sie in der Beispieldatei `model-hpo-configuration.json` im Toolkit.

Sie müssen auch die Metrikdefinition angeben, anhand SageMaker derer der HyperParameter KI-Optimierungsjob die trainierten Kandidatenmodelle bewertet. Hierzu fügen Sie der Datei `model-hpo-configuration.json` das JSON-Objekt `eval_metric` wie folgt hinzu:

```
"eval_metric": {
  "tuning_objective": {
      "MetricName": "(metric_name)",
      "Type": "Maximize"
  },
  "metric_definitions": [
    {
      "Name": "(metric_name)",
```

```
      "Regex": "(metric regular expression)"
    }
  ]
},
```

Das `metric_definitions` Array im `eval_metric` Objekt listet Objekte mit Metrikdefinitionen für jede Metrik auf, die SageMaker KI aus der Trainingsinstanz extrahieren soll. Jedes Metrikdefinitionsobjekt besitzt einen `Name`-Schlüssel, mit dem Sie einen Namen für die Metrik angeben können (z. B. „Genauigkeit“, „F1“ usw.). Mit diesem `Regex`-Schlüssel können Sie eine Zeichenfolge mit einem regulären Ausdruck angeben, der mit der Ausgabe dieser Metrik in den Trainingsprotokollen übereinstimmt. Weitere Informationen zur Definition von Metriken finden Sie auf der Seite SageMaker KI HyperParameter Tuning.

Mit dem Objekt `tuning_objective` in `eval_metric` können Sie dann angeben, welche der Metriken in `metric_definitions` als die Auswertungsmetrik verwendet werden soll, die als Zielmetrik für die Hyperparameter-Optimierung dient. Der Wert für `MetricName` muss mit dem Wert von `Name` in einer der Definitionen in `metric_definitions` übereinstimmen. Der Wert für `Type` sollte entweder „Maximieren“ oder „Minimieren“ lauten, je nachdem, ob die Metrik als greater-is-better (wie „Genauigkeit“) oder less-is-better (wie „“ mean-squared-error interpretiert werden soll).

Fehler in diesem Abschnitt der `model-hpo-configuration.json` Datei können dazu führen, dass der API-Job für das Neptune ML-Modelltraining fehlschlägt, da der SageMaker AI HyperParameter Tuning-Job nicht in der Lage ist, das beste Modell auszuwählen.

Lokale Tests Ihrer benutzerdefinierten Modellimplementierung in Neptune ML

Sie können Ihren Code lokal in der Conda-Umgebung des Neptune-ML-Toolkits ausführen, um Ihr Modell zu testen und zu validieren. Wenn Sie auf einer Neptune-Notebook-Instance entwickeln, wird die Conda-Umgebung auf der Neptune-Notebook-Instanz vorinstalliert. Wenn Sie auf einer anderen Instance entwickeln, müssen Sie die Anweisungen für die lokale Einrichtung im Neptune-ML-Toolkit befolgen.

Die Conda-Umgebung reproduziert exakt die Umgebung, in der Ihr Modell ausgeführt werden wird, wenn Sie die Modelltrainings-API aufrufen. Alle Beispielskripts für Training und Transformation ermöglichen Ihnen die Übergabe des Befehlszeilen-Flags `--local`, um die Skripts zum einfachen Debuggen in einer lokalen Umgebung auszuführen. Sie sollten diese Umgebung bei der Entwicklung Ihres eigenen Modells nutzen, um Ihre Modellimplementierung interaktiv und iterativ zu testen. Beim Modelltraining in der Produktionstrainingsumgebung von Neptune ML wird dieser Parameter ausgelassen.

Erstellen eines Inferenzendpunkts zur Abfrage

Mit einem Inferenzendpunkt können Sie ein einzelnes Modell abfragen, das im Rahmen des Modelltrainingsprozesses erstellt wurde. Der Endpunkt wird dem Modell eines bestimmten Typs mit der besten Leistung angefügt, das beim Training generiert werden konnte. Der Endpunkt kann anschließend Gremlin-Abfragen von Neptune akzeptieren und Modellvorhersagen für die Eingaben in den Abfragen zurückgeben. Nach der Erstellung bleibt ein Inferenzendpunkt aktiv, bis Sie ihn löschen.

Verwalten von Inferenzendpunkten für Neptune ML

Nachdem Sie das Modelltraining mit Daten abgeschlossen haben, die Sie aus Neptune exportiert haben, können Sie mit einem Befehl wie dem folgenden einen Inferenzendpunkt erstellen:

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)"
```

Weitere Informationen finden Sie create-ml-endpointin der AWS CLI Befehlsreferenz.

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**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

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)"
      }'
```

Sie können auf ähnliche Weise einen Inferenzendpunkt anhand eines Modells erstellen, das durch einen abgeschlossenen Modelltransformationsauftrag erstellt wurde:

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)"
```

Weitere Informationen finden Sie create-ml-endpointin der AWS CLI Befehlsreferenz.

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**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

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)"
      }'
```

Die Einzelheiten zur Verwendung dieser Befehle werden in Der Befehl endpoints beschrieben. Dort finden Sie auch Informationen zum Abrufen des Status eines Endpunkts, zum Löschen eines Endpunkts und zum Auflisten aller Inferenzendpunkte.

Inferenzabfragen in Neptune ML

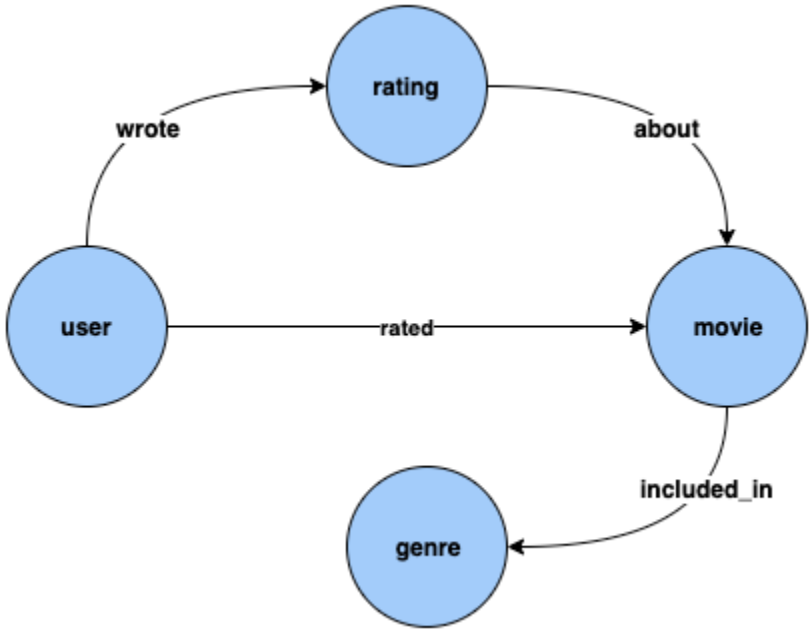
Sie können Gremlin oder SPARQL verwenden, um einen Neptune-ML-Inferenzendpunkt abzufragen. Die induktive Inferenz in Echtzeit wird derzeit jedoch nur für Gremlin-Abfragen unterstützt.

Gremlin-Inferenzabfragen in Neptune ML

Wie in Neptune-ML-Fähigkeiten beschrieben, unterstützt Neptune ML Trainingsmodelle, die die folgenden Arten von Inferenzaufgaben ausführen können:

- Knotenklassifizierung – Sagt das kategorische Feature einer Eckpunkteigenschaft vorher.
- Knotenregression – Sagt eine numerische Eigenschaft eines Eckpunkts vorher.
- Kantenklassifizierung – Sagt das kategorische Feature einer Kanteneigenschaft vorher.
- Kantenregression – Sagt eine numerische Eigenschaft eines Eckpunkts vorher.
- Linkvorhersage – Sagt anhand eines Quellknotens und einer ausgehenden Kante Zielknoten oder anhand eines Zielknotens und einer eingehenden Kante Quellknoten voraus.

Wir können diese verschiedenen Aufgaben anhand von Beispielen veranschaulichen, die den von GroupLens Research bereitgestellten MovieLens 100.000 Datensatz verwenden. Dieser Datensatz besteht aus Filmen, Benutzern und Bewertungen der Filme durch die Benutzer. Aus diesen Daten haben wir das folgende Eigenschaftsdiagramm erstellt:



Knotenklassifizierung: Im Datensatz oben ist `Genre` ein Eckpunkttyp, der über die Kante `included_in` mit dem Eckpunkttyp `Movie` verbunden ist. Wenn der Datensatz jedoch so angepasst wird, dass `Genre` zum kategorischen Feature für den Eckpunkttyp `Movie` wird, kann das Problem der Ableitung des Werts für `Genre` für neue, zum Wissensdiagramm hinzugefügte Filme mit Knotenklassifizierungsmodellen gelöst werden.

Knotenregression: Bei Betrachtung des Eckpunkttyps `Rating`, der Eigenschaften wie `timestamp` und `score` besitzt, kann das Problem der Ableitung des numerischen Werts für `Score` für `Rating` mit Knotenregressionsmodellen gelöst werden.

Kantenklassifizierung: Wenn wir bei einer `Rated` Kante über eine Eigenschaft verfügen, `Scale` die einen der Werte,,,,, haben kann `Love` `Like` `Dislike` `Neutral``Hate`, das Problem der Rückschlüsse auf die Kante `Scale` für eine neue `Rated` movies/ratings Kante ebenfalls mithilfe von Kantenklassifizierungsmodellen gelöst werden.

Kantenregression: Ähnlich kann für die gleiche `Rated`-Kante mit einer Eigenschaft `Score`, die einen numerischen Wert für die Bewertung enthält, kann dies anhand von Kantenregressionsmodellen abgeleitet werden.

Linkvorhersage: Probleme wie die Suche nach den zehn Benutzern, die einen bestimmten Film am wahrscheinlichsten bewerten werden, oder der Suche nach den zehn Filmen, die ein bestimmter Benutzer am wahrscheinlichsten bewerten wird, werden mit Linkvorhersagen gelöst.

> **Note**
>
> Für Neptune-ML-Anwendungsfälle gibt es einen sehr umfangreichen Satz von Notebooks, die Ihnen ein praktisches Verständnis jedes Anwendungsfalls vermitteln sollen. Sie können diese Notizbücher zusammen mit Ihrem Neptune-Cluster erstellen, wenn Sie die Neptune CloudFormation ML-Vorlage verwenden, um einen Neptune-ML-Cluster zu erstellen. Diese Notebooks sind auch auf github verfügbar.

Themen

- In Gremlin-Inferenzabfragen verwendete Neptune-ML-Prädikate
- Gremlin-Knotenklassifizierungsabfragen in Neptune ML
- Gremlin-Abfragen zur Knotenregression in Neptune ML
- Gremlin-Kantenklassifizierungsabfragen in Neptune ML
- Gremlin-Abfragen zur Kantenregression in Neptune ML
- Gremlin-Linkvorhersageabfragen mit Linkvorhersagemodellen in Neptune ML
- Liste der Ausnahmen für Neptune-ML-Gremlin-Inferenzabfragen

In Gremlin-Inferenzabfragen verwendete Neptune-ML-Prädikate

`Neptune#ml.deterministic`

Dieses Prädikat ist eine Option für induktive Inferenzabfragen, d. h. für Abfragen, die das Prädikat `Neptune#ml.inductiveInference` enthalten.

Bei Verwendung der induktiven Inferenz erstellt die Neptune-Engine das entsprechende Unterdiagramm zur Auswertung des trainierten GNN-Modells. Die Anforderungen dieses Unterdiagramms sind von den Parametern des endgültigen Modells abhängig. Insbesondere bestimmt der Parameter `num-layer` die Anzahl der Traversierungs-Hops von den Zielknoten oder -kanten. Der Parameter `fanouts` gibt an, wie viele Nachbarn bei jedem Hop abgefragt werden sollen (siehe HPO-Parameter).

Standardmäßig werden induktive Inferenzabfragen nicht deterministisch ausgeführt, d. h. Neptune erstellt die Nachbarschaft in randomisierter Weise. Diese normalen Abfragen randomisierter Nachbarschaften führen manchmal zu unterschiedlichen Vorhersagen.

Wenn Sie `Neptune#ml.deterministic` in eine induktive Inferenzabfrage einfügen, versucht die Neptune-Engine, Nachbarn deterministisch abzufragen, sodass mehrere Aufrufe derselben Abfrage jedes Mal dieselben Ergebnisse zurückgeben. Es kann jedoch nicht garantiert werden, dass die Ergebnisse vollständig deterministisch sind, da Änderungen des zugrunde liegenden Diagramms und Artefakte verteilter Systeme weiter zu Schwankungen führen können.

Sie fügen das Prädikat `Neptune#ml.deterministic` in eine Abfrage wie die folgende ein:

```
.with("Neptune#ml.deterministic")
```

Wenn das Prädikat `Neptune#ml.deterministic` in einer Abfrage enthalten ist, die nicht auch `Neptune#ml.inductiveInference` enthält, wird es einfach ignoriert.

`Neptune#ml.disableInductiveInferenceMetadataCache`

Dieses Prädikat ist eine Option für induktive Inferenzabfragen, d. h. für Abfragen, die das Prädikat `Neptune#ml.inductiveInference` enthalten.

Für induktive Inferenzabfragen verwendet Neptune eine in Amazon S3 gespeicherte Metadatendatei, um bei der Erstellung der Nachbarschaft die Anzahl der Hops und den Fanout zu bestimmen. Neptune speichert diese Modellmetadaten normalerweise im Cache, um zu vermeiden, dass die

Datei wiederholt aus Amazon S3 abgerufen wird. Das Caching kann durch Einfügen des Prädikats `Neptune#ml.disableInductiveInferenceMetadataCache` deaktiviert werden. Obwohl es für Neptune langsamer sein kann, die Metadaten direkt von Amazon S3 abzurufen, ist es nützlich, wenn der SageMaker KI-Endpunkt nach einem erneuten Training oder einer Transformation aktualisiert wurde und der Cache veraltet ist.

Sie fügen das Prädikat `Neptune#ml.disableInductiveInferenceMetadataCache` in eine Abfrage wie folgt ein:

```
.with("Neptune#ml.disableInductiveInferenceMetadataCache")
```

Dies ist ein Beispielabfrage in einem Jupyter-Notebook:

```
%%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`

Das Prädikat `Neptune#ml.endpoint` wird in einem `with()`-Schritt verwendet, um den Inferenzendpunkt anzugeben, wenn notwendig:

```
 .with("Neptune#ml.endpoint", "the model's SageMaker AI inference endpoint")
```

Sie können den Endpunkt anhand der `id` oder der URL identifizieren. Beispiel:

```
 .with( "Neptune#ml.endpoint", "node-classification-movie-lens-endpoint" )
```

Oder:

```
 .with( "Neptune#ml.endpoint", "https://runtime.sagemaker.us-east-1.amazonaws.com/
endpoints/node-classification-movie-lens-endpoint/invocations" )
```

> **Note**
>
> Wenn Sie den Parameter`neptune_ml_endpoint` in Ihrer Neptune-DB-Cluster-Parametergruppe auf den Endpunkt `id` oder die URL festlegen, müssen Sie das Prädikat `Neptune#ml.endpoint` nicht in jede Abfrage einfügen.

`Neptune#ml.iamRoleArn`

`Neptune#ml.iamRoleArn`wird in einem `with()` Schritt verwendet, um den ARN der SageMaker AI-Ausführungs-IAM-Rolle anzugeben, falls erforderlich:

```
.with("Neptune#ml.iamRoleArn", "the ARN for the SageMaker AI execution IAM role")
```

Informationen zum Erstellen der IAM-Rolle für die SageMaker AI-Ausführung finden Sie unter. Erstellen Sie eine benutzerdefinierte Rolle NeptuneSageMaker IAMRole

> **Note**
>
> Wenn Sie den `neptune_ml_iam_role` Parameter in Ihrer Neptune DB-Cluster-Parametergruppe auf den ARN Ihrer IAM-Rolle für die SageMaker KI-Ausführung festlegen, müssen Sie das `Neptune#ml.iamRoleArn` Prädikat nicht in jede Abfrage aufnehmen.

Neptune#ml.inductiveInference

Die transduktive Inferenz ist in Gremlin standardmäßig aktiviert. Um eine induktive Inferenzabfrage in Echtzeit zu erstellen, fügen Sie das Prädikat `Neptune#ml.inductiveInference` wie folgt hinzu:

```
.with("Neptune#ml.inductiveInference")
```

Wenn Ihr Diagramm dynamisch ist, ist die induktive Inferenz häufig die beste Wahl. Wenn Ihr Diagramm jedoch statisch ist, ist die transduktive Inferenz schneller und effizienter.

`Neptune#ml.limit`

Das Prädikat `Neptune#ml.limit` begrenzt optional die Anzahl der Ergebnisse, die pro Entität zurückgegeben werden:

```
.with( "Neptune#ml.limit", 2 )
```

Standardmäßig ist der Grenzwert 1. Die maximale Anzahl, die festgelegt werden kann, ist 100.

Neptune#ml.threshold

Das Prädikat `Neptune#ml.threshold` legt optional einen Mindestwert für Ergebniswerte fest:

```
.with( "Neptune#ml.threshold", 0.5D )
```

So werden alle Ergebnisse verworfen, deren Werte unter dem angegebenen Schwellenwert liegen.

Neptune#ml.classification

Das `Neptune#ml.classification` Prädikat wird an den `properties()` Schritt angehängt, um festzulegen, dass die Eigenschaften vom SageMaker KI-Endpunkt des Knotenklassifikationsmodells abgerufen werden müssen:

```
.properties( "property key of the node classification
model" ).with( "Neptune#ml.classification" )
```

Neptune#ml.regression

Das `Neptune#ml.regression` Prädikat wird an den `properties()` Schritt angehängt, um festzulegen, dass die Eigenschaften vom SageMaker KI-Endpunkt des Knotenregressionsmodells abgerufen werden müssen:

```
.properties( "property key of the node regression
model" ).with( "Neptune#ml.regression" )
```

Neptune#ml.prediction

Das Prädikat `Neptune#ml.prediction` wird an die Schritte `in()` und `out()` angefügt, um festzulegen, dass es sich um eine Linkvorhersage-Abfrage handelt:

```
.in("edge label of the link prediction
model").with("Neptune#ml.prediction").hasLabel("target node label")
```

`Neptune#ml.score`

Das Prädikat `Neptune#ml.score` wird in Abfragen zur Klassifizierung von Gremlin-Knoten oder -Kanten verwendet, um Machine-Learning-Konfidenzwerte abzurufen. Das Prädikat `Neptune#ml.score` sollte zusammen mit dem Abfrageprädikat im Schritt `properties()` übergeben werden, um ML-Konfidenzwerte für Abfragen zur Klassifizierung von Knoten oder Kanten zu erhalten.

Sie finden ein Beispiel für die Knotenklassifizierung mit weiteren Beispielen für die Knotenklassifizierung und ein Beispiel für die Kantenklassifizierung im Abschnitt für die Kantenklassifizierung.

Gremlin-Knotenklassifizierungsabfragen in Neptune ML

Für die Gremlin-Knotenklassifizierung in Neptune ML:

- Das Modell wird anhand einer einzelnen Eigenschaft der Eckpunkte trainiert. Der Satz der eindeutigen Werte dieser Eigenschaft wird als Satz von Knotenklassen oder einfach Klassen bezeichnet.
- Die Knotenklasse oder der kategorische Eigenschaftswert einer Eckpunkteigenschaft kann aus dem Knotenklassifikationsmodell abgeleitet werden. Dies ist nützlich, wenn diese Eigenschaft dem Eckpunkt noch nicht angefügt ist.
- Um eine oder mehrere Klassen aus einem Knotenklassifikationsmodell abzurufen, müssen Sie den Schritt `with()` mit dem Prädikat `Neptune#ml.classification` verwenden, um den Schritt `properties()` zu konfigurieren. Das Ausgabeformat ist dem Ausgabeformat für Eckpunkteigenschaften ähnlich.

> **Note**
>
> Die Knotenklassifizierung funktioniert nur mit Zeichenfolgen-Eigenschaftswerten. Das bedeutet, dass numerische Eigenschaftswerte wie `0` oder `1` nicht unterstützt werden, auch wenn die Zeichenfolgenäquivalente `"0"` und `"1"` unterstützt werden. Ähnlich funktionieren die booleschen Eigenschaftswerte `true` und `false` nicht, während `"true"` und `"false"` funktionieren.

Dies ist ein Beispiel für eine Abfrage zur Knotenklassifizierung:

```
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")
```

Die Ausgabe dieser Abfrage sieht ähnlich wie folgt aus:

```
==>vp[genre->Action]
==>vp[genre->Crime]
==>vp[genre->Comedy]
```

In der Abfrage oben werden die Schritte `V()` und `properties()` wie folgt verwendet:

Der Schritt `V()` enthält den Satz von Eckpunkten, für die Sie die Klassen aus dem Knotenklassifizierungsmodell abrufen möchten:

```
 .V( "movie_1", "movie_2", "movie_3" )
```

Der Schritt `properties()` enthält den Schlüssel, mit dem das Modell trainiert wurde, und besitzt `.with("Neptune#ml.classification")`, um anzugeben, dass es sich um eine ML-Inferenzabfrage zur Knotenklassifizierung handelt.

Der Schritt `properties().with("Neptune#ml.classification")` unterstützt zurzeit mehrere Eigenschaftsschlüssel nicht. Die folgende Abfrage führt beispielsweise zu einer Ausnahme:

```
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")
```

Die spezifische Fehlermeldung finden Sie in der Liste der Neptune-ML-Ausnahmen.

Der Schritt `properties().with("Neptune#ml.classification")` kann in Kombination mit einem der folgenden Schritte verwendet werden:

- `value()`
- `value().is()`

- `hasValue()`
- `has(value,"")`
- `key()`
- `key().is()`
- `hasKey()`
- `has(key,"")`
- `path()`

Weitere Abfragen zur Knotenklassifizierung

Wenn sowohl der Inferenzendpunkt als auch die entsprechende IAM-Rolle in der DB-Cluster-Parametergruppe gespeichert wurden, kann eine Knotenklassifizierungsabfrage so einfach wie die folgende sein:

```
g.V("movie_1", "movie_2",
 "movie_3").properties("genre").with("Neptune#ml.classification")
```

Mit dem Schritt `union()` können Sie Eckpunkteigenschaften und Klassen in einer Abfrage kombinieren:

```
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")
 )
```

Sie können auch eine unbegrenzte Abfrage wie die folgende erstellen:

```
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")
```

Sie können die Knotenklassen zusammen mit Eckpunkten abrufen, indem Sie den Schritt `select()` zusammen mit dem Schritt `as()` ausführen:

```
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")
```

Sie können auch nach Knotenklassen filtern, wie in den folgenden Beispielen gezeigt:

```
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"))
```

Sie können mithilfe des Prädikats `Neptune#ml.score` einen Konfidenzwert für die Knotenklassifizierung abrufen:

```
 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")
```

Die Antwort würde wie folgt aussehen:

```
==>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]
```

Verwenden der induktiven Inferenz in einer Abfrage zur Knotenklassifizierung

Angenommen, Sie fügen einem vorhandenen Diagramm in einem Jupyter-Notebook einen neuen Knoten wie folgt hinzu:

```
%%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')
```

Sie könnten dann eine induktive Inferenzabfrage verwenden, um ein Genre und einen Konfidenzwert abzurufen, die den neuen Knoten widerspiegeln:

```
%%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")
```

Wenn Sie die Abfrage jedoch mehrmals ausgeführt haben, erhalten Sie möglicherweise etwas andere Ergebnisse:

```
# First time
==>vp[genre->Action]
==>vp[Neptune#ml.score->0.12345678]

# Second time
==>vp[genre->Action]
==>vp[Neptune#ml.score->0.21365921]
```

Sie könnten dieselbe Abfrage deterministisch machen:

```
%%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")
```

In diesem Fall wären die Ergebnisse jedes Mal ungefähr gleich:

```
# First time
==>vp[genre->Action]
==>vp[Neptune#ml.score->0.12345678]
# Second time
==>vp[genre->Action]
==>vp[Neptune#ml.score->0.12345678]
```

Gremlin-Abfragen zur Knotenregression in Neptune ML

Die Knotenregression ist der Knotenklassifizierung ähnlich. Der aus dem Regressionsmodell für jeden Knoten abgeleitete Wert ist jedoch numerisch. Sie können für die Knotenregression die gleichen Gremlin-Abfragen wie für die Knotenklassifizierung verwenden, mit einigen Unterschieden:

- In Neptune ML beziehen sich Knoten auf Eckpunkte.
- Der Schritt `properties()` hat statt `properties().with("Neptune#ml.regression")` das Format `properties().with("Neptune#ml.classification")`.
- Die Prädikate `"Neptune#ml.limit` und `"Neptune#ml.threshold"` sind nicht anwendbar.
- Wenn Sie nach dem Wert filtern, müssen Sie einen numerischen Wert angeben.

Dies ist ein Beispiel für eine Abfrage zur Eckklassifizierung:

```
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")
```

Sie können mithilfe eines Regressionsmodells nach dem abgeleiteten Wert filtern, wie in den folgenden Beispielen gezeigt:

```
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))
```

Verwenden der induktiven Inferenz in einer Knotenregressionsabfrage

Angenommen, Sie fügen einem vorhandenen Diagramm in einem Jupyter-Notebook einen neuen Knoten wie folgt hinzu:

```
%%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')
```

Sie könnten dann eine induktive Inferenzabfrage verwenden, um eine Bewertung zu erhalten, die den neuen Knoten berücksichtigt:

```
%%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")
```

Da die Abfrage nicht deterministisch ist, kann sie je nach Nachbarschaft zu etwas anderen Ergebnissen führen, wenn Sie sie mehrmals ausführen:

```
# First time
==>vp[rating->9.1]

# Second time
==>vp[rating->8.9]
```

Wenn Sie konsistentere Ergebnisse benötigen, können Sie die Abfrage deterministisch gestalten:

```
%%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")
```

Jetzt werden die Ergebnisse jedes Mal ungefähr gleich sein:

```
# First time
==>vp[rating->9.1]

# Second time
==>vp[rating->9.1]
```

Gremlin-Kantenklassifizierungsabfragen in Neptune ML

Für die Gremlin-Kantenklassifizierung in Neptune ML:

- Das Modell wird anhand einer einzelnen Eigenschaft der Kanten trainiert. Der Satz der eindeutigen Werte dieser Eigenschaft wird als Satz von Klassen bezeichnet.
- Der Klassen- oder kategorische Eigenschaftswert einer Kante kann aus dem Kantenklassifizierungsmodell abgeleitet werden. Dies ist nützlich, wenn diese Eigenschaft noch nicht mit der Kante verknüpft ist.
- Um eine oder mehrere Klassen aus einem Kantenklassifikationsmodell abzurufen, müssen Sie den Schritt `with()` mit dem Prädikat `"Neptune#ml.classification"` verwenden, um den Schritt `properties()` zu konfigurieren. Das Ausgabeformat ist dem Ausgabeformat für Kanteneigenschaften ähnlich.

> **Note**
>
> Die Kantenklassifizierung funktioniert nur mit Zeichenfolgen-Eigenschaftswerten. Das bedeutet, dass numerische Eigenschaftswerte wie `0` oder `1` nicht unterstützt werden, auch wenn die Zeichenfolgenäquivalente `"0"` und `"1"` unterstützt werden. Ähnlich funktionieren die booleschen Eigenschaftswerte `true` und `false` nicht, während `"true"` und `"false"` funktionieren.

Dies ist ein Beispiel für eine Abfrage zur Kantenklassifizierung, bei der mithilfe des Prädikats `Neptune#ml.score` ein Konfidenzwert angefordert wird:

```
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")
```

Die Antwort würde wie folgt aussehen:

```
==>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]
```

Syntax einer Gremlin-Kantenklassifizierungsabfrage

Für ein einfaches Diagramm, in dem `User` der Kopf- und Endknoten ist und `Relationship` die Kante ist, die sie verbindet, wäre dies beispielsweise eine Abfrage zur Kantenklassifizierung:

```
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")
```

Die Ausgabe dieser Abfrage sieht ähnlich wie folgt aus:

```
==>p[knows_by->"Family"]
==>p[knows_by->"Friends"]
==>p[knows_by->"Colleagues"]
```

In der Abfrage oben werden die Schritte `E()` und `properties()` wie folgt verwendet:

- Der Schritt `E()` enthält den Satz von Kanten, für die Sie die Klassen aus dem Kantenklassifizierungsmodell abrufen möchten:

  ```
  .E("relationship_1","relationship_2","relationship_3")
  ```

- Der Schritt `properties()` enthält den Schlüssel, mit dem das Modell trainiert wurde, und besitzt `.with("Neptune#ml.classification")`, um anzugeben, dass es sich um eine ML-Inferenzabfrage zur Kantenklassifizierung handelt.

Der Schritt `properties().with("Neptune#ml.classification")` unterstützt zurzeit mehrere Eigenschaftsschlüssel nicht. Die folgende Abfrage löst beispielsweise eine Ausnahme aus:

```
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")
```

Spezifische Fehlermeldungen finden Sie unter Liste der Ausnahmen für Neptune-ML-Gremlin-Inferenzabfragen.

Der Schritt `properties().with("Neptune#ml.classification")` kann in Kombination mit einem der folgenden Schritte verwendet werden:

- `value()`
- `value().is()`
- `hasValue()`
- `has(value,"")`
- `key()`
- `key().is()`
- `hasKey()`
- `has(key,"")`
- `path()`

Verwenden der induktiven Inferenz in einer Abfrage zur Kantenklassifizierung

Angenommen, Sie fügen einem vorhandenen Diagramm in einem Jupyter-Notebook eine neue Kante wie folgt hinzu:

```
%%gremlin
g.V('1').as('fromV')
.V('2').as('toV')
```

```
.addE('eLabel1').from('fromV').to('toV').property(id, 'e101')
```

Sie könnten dann eine induktive Inferenzabfrage verwenden, um eine Skala zu erhalten, die die neue Kante berücksichtigt:

```
%%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")
```

Da die Abfrage nicht deterministisch ist, kann sie je nach randomisierter Nachbarschaft zu etwas anderen Ergebnissen führen, wenn sie mehrmals ausgeführt wird:

```
# First time
==>vp[scale->Like]
==>vp[Neptune#ml.score->0.12345678]

# Second time
==>vp[scale->Like]
==>vp[Neptune#ml.score->0.21365921]
```

Wenn Sie konsistentere Ergebnisse benötigen, können Sie die Abfrage deterministisch gestalten:

```
%%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")
```

Jetzt sind die Ergebnisse bei jeder Ausführung der Abfrage mehr oder weniger gleich:

```
# First time
==>vp[scale->Like]
==>vp[Neptune#ml.score->0.12345678]

# Second time
==>vp[scale->Like]
```

```
==>vp[Neptune#ml.score->0.12345678]
```

Gremlin-Abfragen zur Kantenregression in Neptune ML

Die Kantenregression ist der Kantenklassifizierung ähnlich. Der aus dem ML-Modell abgeleitete Wert ist jedoch numerisch. Neptune ML unterstützt für die Kantenregression dieselben Abfragen wie für die Klassifizierung.

Wichtige Punkte, die beachtet werden müssen, sind:

- Sie müssen das ML-Prädikat `"Neptune#ml.regression"` verwenden, um den Schritt `properties()` für diesen Anwendungsfall zu konfigurieren.
- Die Prädikate `"Neptune#ml.limit"` und `"Neptune#ml.threshold"` sind in diesem Anwendungsfall nicht anwendbar.
- Um nach dem Wert zu filtern, müssen Sie den Wert als numerisch angeben.

Syntax einer Gremlin-Kantenregressionsabfrage

Für ein einfaches Diagramm, bei dem `User` der Kopfknoten, `Movie` der Endknoten und `Rated` die Kante ist, die diese verbindet, wäre dies ein Beispiel für eine Kantenregressionsabfrage, die den numerischen Wert der Bewertung (hier als Punktzahl bezeichnet) für die Kante `Rated` sucht:

```
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")
```

Sie können auch nach einem Wert filtern, der aus dem ML-Regressionsmodell abgeleitet wurde. Für die vorhandenen `Rated`-Kanten (von `User` bis`Movie`), identifiziert durch`"rating_1"`, `"rating_2"` und `"rating_3"`, wobei die Kanteneigenschaft `Score` für diese Bewertungen nicht vorhanden ist, können Sie eine Abfrage wie die folgende verwenden, um `Score` für die Kanten abzuleiten, deren Wert größer oder gleich 9 ist:

```
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))
```

Verwenden der induktiven Inferenz in einer Kantenregressionsabfrage

Angenommen, Sie fügen einem vorhandenen Diagramm in einem Jupyter-Notebook eine neue Kante wie folgt hinzu:

```
%%gremlin
g.V('1').as('fromV')
.V('2').as('toV')
.addE('eLabel1').from('fromV').to('toV').property(id, 'e101')
```

Sie könnten dann eine induktive Inferenzabfrage verwenden, um eine Punktzahl zu erhalten, die die neue Kante berücksichtigt:

```
%%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")
```

Da die Abfrage nicht deterministisch ist, kann sie je nach randomisierter Nachbarschaft zu etwas anderen Ergebnissen führen, wenn sie mehrmals ausgeführt wird:

```
# First time
==>ep[score->96]

# Second time
==>ep[score->91]
```

Wenn Sie konsistentere Ergebnisse benötigen, können Sie die Abfrage deterministisch gestalten:

```
%%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")
```

Jetzt sind die Ergebnisse bei jeder Ausführung der Abfrage mehr oder weniger gleich:

```
# First time
==>ep[score->96]

# Second time
==>ep[score->96]
```

Gremlin-Linkvorhersageabfragen mit Linkvorhersagemodellen in Neptune ML

Modelle zur Linkvorhersage können Probleme wie die folgenden lösen:

- Vorhersage des Kopfknotens: Mit welchen Eckpunkten ist ein bestimmter Eckpunkt wahrscheinlich verbunden, wenn ein Eckpunkt und ein Kantentyp vorhanden sind?
- Vorhersage des Endknotens: Mit welchen Eckpunkten ist ein bestimmter Eckpunkt wahrscheinlich verbunden, wenn ein Eckpunkt und eine Kantenbezeichnung vorhanden sind?

> **Note**
>
> Die Kantenvorhersage wird in Neptune ML noch nicht unterstützt.

Stellen Sie sich für die folgenden Beispiele ein einfaches Diagramm mit den Eckpunkten `User` und `Movie` vor, die durch die Kante `Rated` verbunden sind.

Dies ist ein Beispiel für eine Abfrage zur Vorhersage des Kopfknotens, um die fünf Benutzer vorherzusagen, die die Filme `"movie_1"`, `"movie_2"` und `"movie_3"` am wahrscheinlichsten bewerten werden:

```
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")
```

Dies ist eine ähnliche Abfrage zur Vorhersage des Endknotens, um die fünf Filme vorherzusagen, die der Benutzer `"user_1"` wahrscheinlich bewerten wird:

```
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")
```

Sowohl die Kantenbezeichnung als auch die vorhergesagte Eckpunktbezeichnung sind erforderlich. Wird eins von beiden ausgelassen, wird eine Ausnahme ausgelöst. Beispielsweise löst die folgende Abfrage ohne vorhergesagte Eckpunktbezeichnung eine Ausnahme aus:

```
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")
```

In ähnlicher Weise löst die folgende Abfrage ohne Kantenbezeichnung eine Ausnahme aus:

```
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")
```

Die spezifische, von diesen Ausnahmen zurückgegebene Fehlermeldung finden Sie in der Liste der Neptune-ML-Ausnahmen.

Weitere Abfragen zur Linkvorhersage

Sie können Schritt `select()` mit Schritt `as()` verwenden, um die vorhergesagten Eckpunkte zusammen mit den eingegebenen Eckpunkten auszugeben:

```
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")
```

Sie können unbegrenzte Abfragen wie die folgenden erstellen:

```
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")
```

Verwenden der induktiven Inferenz in einer Linkvorhersageabfrage

Angenommen, Sie fügen einem vorhandenen Diagramm in einem Jupyter-Notebook einen neuen Knoten wie folgt hinzu:

```
%%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')
```

Sie könnten dann eine induktive Inferenzabfrage verwenden, um den Kopfknoten unter Berücksichtigung des neuen Knotens vorherzusagen:

```
%%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")
```

Ergebnis:

```
==>V[2]
```

Sie könnten dann auf ähnliche Weise mit einer induktiven Inferenzabfrage den Endknoten unter Berücksichtigung des neuen Knotens vorhersagen:

```
%%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")
```

Ergebnis:

```
==>V[1]
```

Liste der Ausnahmen für Neptune-ML-Gremlin-Inferenzabfragen

Dies ist eine umfassende Liste von Ausnahmen, die bei der Ausführung von Neptune ML Gremlin-Inferenzabfragen auftreten können. Diese Ausnahmen decken eine Reihe von Problemen ab, von Problemen mit der angegebenen IAM-Rolle oder dem angegebenen Endpunkt bis hin zu nicht unterstützten Gremlin-Schritten und Einschränkungen bei der Anzahl der ML-Inferenzabfragen pro Abfrage. Jeder Eintrag enthält eine ausführliche Meldung, in der das Problem beschrieben wird.

- **`BadRequestException`** – Die Anmeldeinformationen für die angegebene Rolle können nicht geladen werden.

  Meldung: `Unable to load credentials for role: the specified IAM Role ARN.`
- **`BadRequestException`**— Die angegebene IAM-Rolle ist nicht berechtigt, den SageMaker KI-Endpunkt aufzurufen.

  Meldung: `User: the specified IAM Role ARN is not authorized to perform: sagemaker:InvokeEndpoint on resource: the specified endpoint.`
- **`BadRequestException`** – Der angegebene Endpunkt ist nicht vorhanden.

  Meldung: `Endpoint the specified endpoint not found.`
- **`InternalFailureException`** – Es können keine Neptune-ML-Metadaten für induktive Inferenzen in Echtzeit aus Amazon S3 abgerufen werden.

  Meldung: `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 kann die Metadatendatei für induktive Inferenzen in Echtzeit nicht in Amazon S3 finden.

Meldung: `Neptune ML cannot find the metadata file for Real-Time Inductive Inference in S3.`

- **`InvalidParameterException`** – Die Syntax des angegebenen Endpunkts ist nicht gültig.

  Meldung: `Invalid endpoint provided for external service query.`

- **`InvalidParameterException`**— Der angegebene SageMaker Ausführungs-IAM-Rollen-ARN ist syntaktisch nicht gültig.

  Meldung: `Invalid IAM role ARN provided for external service query.`

- **`InvalidParameterException`** – In Schritt `properties()` einer Abfrage sind mehrere Eigenschaftsschlüssel angegeben.

  Meldung: `ML inference queries are currently supported for one property key.`

- **`InvalidParameterException`** – In einer Abfrage sind mehrere Kantenbezeichnungen angegeben.

  Meldung: `ML inference are currently supported only with one edge label.`

- **`InvalidParameterException`** – In einer Abfrage sind mehrere Einschränkungen für Eckpunktbezeichnungen angegeben.

  Meldung: `ML inference are currently supported only with one vertex label constraint.`

- **`InvalidParameterException`** – In derselben Abfrage sind sowohl das Prädikat `Neptune#ml.classification` als auch das Prädikat `Neptune#ml.regression` vorhanden.

  Meldung: `Both regression and classification ML predicates cannot be specified in the query.`

- **`InvalidParameterException`** – In Schritt `in()` oder `out()` einer Abfrage zur Linkvorhersage wurde mehr als eine Kantenbezeichnung angegeben.

  Meldung: `ML inference are currently supported only with one edge label.`

- **`InvalidParameterException`** – Es wurde mehr als ein Eigenschaftsschlüssel mit Neptune#ml.score angegeben.

  Meldung: `Neptune ML inference queries are currently supported for one property key and one Neptune#ml.score property key.`

- **`MissingParameterException`** – Der Endpunkt wurde weder in der Abfrage noch als DB-Cluster-Parameter angegeben.

  Meldung: `No endpoint provided for external service query.`

- MissingParameterException— Die IAM-Rolle für die SageMaker AI-Ausführung wurde weder in der Abfrage noch als DB-Cluster-Parameter angegeben.

  Meldung: `No IAM role ARN provided for external service query.`

- **`MissingParameterException`** – In Schritt `properties()` einer Abfrage fehlt der Eigenschaftsschlüssel.

  Meldung: `Property key needs to be specified using properties() step for ML inference queries.`

- **`MissingParameterException`** – In Schritt `in()` oder `out()` einer Abfrage zur Linkvorhersage wurde keine Kantenbezeichnung angegeben.

  Meldung: `Edge label needs to be specified while using in() or out() step for ML inference queries.`

- **`MissingParameterException`** – Es wurde kein Eigenschaftsschlüssel mit Neptune#ml.score angegeben.

  Meldung: `Property key needs to be specified along with Neptune#ml.score property key while using the properties() step for Neptune ML inference queries.`

- **`UnsupportedOperationException`** – In einer Abfrage zur Linkvorhersage wird Schritt `both()` verwendet.

  Meldung: `ML inference queries are currently not supported with both() step.`

- **`UnsupportedOperationException`** – In Schritt `has()` mit Schritt `in()` oder `out()` einer Abfrage zur Linkvorhersage-Abfrage wurde keine vorhergesagte Eckpunktbezeichnung angegeben.

  Meldung: `Predicted vertex label needs to be specified using has() step for ML inference queries.`

- **`UnsupportedOperationException`** – Induktive Gremlin-ML-Inferenzabfragen werden zurzeit nicht mit nicht optimierten Schritten unterstützt.

Meldung: `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`** – Neptune-ML-Inferenzabfragen werden zurzeit nicht innerhalb des Schritts `repeat` unterstützt.

  Meldung: `Neptune ML inference queries are currently not supported inside a repeat step.`

- **`UnsupportedOperationException`** – Zurzeit wird nur eine Neptune-ML-Inferenzabfrage pro Gremlin-Abfrage unterstützt.

  Meldung: `Neptune ML inference queries are currently supported only with one ML inference query per gremlin query.`

SPARQL-Inferenzabfragen in Neptune ML

Neptune ML setzt das RDF-Diagramm in ein Eigenschaftsdiagramm um, um die ML-Aufgabe zu modellieren. Derzeit werden die folgenden Anwendungsfälle unterstützt:

- Objektklassifizierung – Vorhersage des kategorischen Features eines Objekts.
- Objektregression – Vorhersage der numerischen Eigenschaft eines Objekts.
- Objektvorhersage – Vorhersage eines Objekts anhand eines Subjekts und einer Beziehung.
- Subjektvorhersage – Vorhersage eines Subjekts anhand eines Objekts und einer Beziehung.

> **Note**
>
> Neptune ML unterstützt keine Anwendungsfälle für Subjektklassifizierung und -regression mit SPARQL.

In SPARQL-Inferenzabfragen verwendete Neptune-ML-Prädikate

Die folgenden Prädikate werden für SPARQL-Inferenzen verwendet:

Prädikat

Gibt das Timeout für die Verbindung mit dem Remoteserver an. Dies sollte nicht mit dem Timeout für Abfrageanforderungen verwechselt werden, bei dem es sich um die maximale Zeit handelt, die der Server für die Bearbeitung einer Anforderung benötigen darf.

Wenn das Abfragetimeout vor dem durch das Prädikat `neptune-ml:timeout` angegebenen Service-Timeout eintritt, wird auch die Service-Verbindung abgebrochen.

Prädikat **`neptune-ml:outputClass`**

Das Prädikat `neptune-ml:outputClass` wird nur für die Definition der Klasse des vorhergesagten Objekts für die Objektvorhersage oder des vorhergesagten Subjekts für die Subjektvorhersage verwendet.

Prädikat **`neptune-ml:outputScore`**

Das Prädikat `neptune-ml:outputScore` ist eine positive Zahl zur Darstellung der Wahrscheinlichkeit, mit der die Ausgabe eines Machine-Learning-Modells korrekt ist.

Prädikat **neptune-ml:modelType**

Das Prädikat `neptune-ml:modelType` gibt den Typ des Machine-Learning-Modells an, das trainiert wird:

- `OBJECT_CLASSIFICATION`
- `OBJECT_REGRESSION`
- `OBJECT_PREDICTION`
- `SUBJECT_PREDICTION`

Prädikat **neptune-ml:input**

Das `neptune-ml:input` Prädikat bezieht sich auf die Liste der als Eingaben für Neptune ML URIs verwendeten.

Prädikat **neptune-ml:output**

Das Prädikat `neptune-ml:output` verweist auf die Liste der Bindungssätze, in denen Neptune ML Ergebnisse zurückgibt.

Prädikat **neptune-ml:predicate**

Das Prädikat `neptune-ml:predicate` wird je nach ausgeführter Aufgabe unterschiedlich verwendet:

- Für die Objekt- oder Subjektvorhersage: Definition des Typs des Prädikats (Kanten- oder Beziehungstyp).
- Für die Objektklassifizierung und -regression: Definition des Literals (der Eigenschaft), die vorhergesagt werden soll.

Prädikat **neptune-ml:batchSize**

Das Prädikat `neptune-ml:batchSize` gibt die Eingabegröße für den Remote-Serviceaufruf an.

Beispiele für die SPARQL-Objektklassifizierung

Für die SPARQL-Objektklassifizierung in Neptune ML wird das Modell anhand eines Prädikatwerts trainiert. Dies ist nützlich, wenn das betreffende Prädikat für ein bestimmtes Subjekt noch nicht vorhanden ist.

Mit dem Objektklassifizierungsmodell können nur kategorische Prädikatwerte abgeleitet werden.

Die folgende Abfrage versucht, den Prädikatwert für <http://www.example.org/team> für alle Eingaben des Typs `foaf:Person` vorherzusagen:

```
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 .
  }
}
```

Diese Abfrage kann wie folgt angepasst werden:

```
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 .
  }
}
```

Beispiele für die SPARQL-Objektregression

Die Objektregression ist der Objektklassifizierung ähnlich. Der aus dem Regressionsmodell für jeden Knoten abgeleitete Prädikatwert ist jedoch numerisch. Sie können für die Objektregression dieselben SPARQL-Abfragen wie für die Objektklassifizierung verwenden, mit der Ausnahme, dass die Prädikate `the Neptune#ml.limit` und `Neptune#ml.threshold` nicht anwendbar sind.

Die folgende Abfrage versucht, den Prädikatwert für <http://www.example.org/accountbalance> für alle Eingaben des Typs `foaf:Person` vorherzusagen:

```
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 .
  }
}
```

Diese Abfrage kann wie folgt angepasst werden:

```
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 .
  }
}
```

Beispiel für eine SPARQL-Objektvorhersage

Die Objektvorhersage sagt den Objektwert für ein bestimmtes Subjekt und Prädikat voraus.

Mit der folgenden Abfrage zur Objektvorhersage soll vorhergesagt werden, welchen Film eine Eingabe des Typs `foaf:Person` mit einem „Like“ bewerten würde:

```
?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> .
  }
}
```

Die Abfrage selbst könnte wie folgt angepasst werden:

```
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> .
  }
}
```

Beispiel für eine SPARQL-Subjektvorhersage

Die Subjektvorhersage sagt den Subjektwert für ein bestimmtes Objekt und Prädikat voraus.

Die folgende Abfrage sagt beispielsweise voraus, wer (des Typs `foaf:User`) sich einen bestimmten Film ansehen wird:

```
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 der Ausnahmen für Neptune-ML-SPARQL-Inferenzabfragen

- **BadRequestException** – Meldung: 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** – Meldung: 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** – Meldung: Invalid predicate *(predicate name)* provided for external service http://aws.amazon.com/neptune/vocab/v01/services/ml#inference query.
- **BadRequestException** – Meldung: The SERVICE http://aws.amazon.com/neptune/vocab/v01/services/ml#inference expects the predicate *(predicate name)* to be defined
- **BadRequestException** – Meldung: 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** – Meldung: The SERVICE http://aws.amazon.com/neptune/vocab/v01/services/ml#inference expects the input *(parameter name)* to be a constant, found: *(type)*
- **BadRequestException** – Meldung: The SERVICE http://aws.amazon.com/neptune/vocab/v01/services/ml#inference is expected to return only 1 value
- **BadRequestException** – Meldung: "The SERVICE http://aws.amazon.com/neptune/vocab/v01/services/ml#inference only allows StatementPatternNodes
- **BadRequestException** – Meldung: The SERVICE http://aws.amazon.com/neptune/vocab/v01/services/ml#inference does not allow the predicate *(predicate name)*
- **BadRequestException** – Meldung: The SERVICE http://aws.amazon.com/neptune/vocab/v01/services/ml#inference predicates cannot be variables, found: *(type)*

- **`BadRequestException`** – Meldung: `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)`*

Neptune-ML-Management-API-Referenz

Inhalt

- Datenverarbeitung mit dem Befehl dataprocessing
  - Erstellen eines Datenverarbeitungsauftrags mit dem Neptune-ML-Befehl dataprocessing
  - Abrufen des Status eines Datenverarbeitungsauftrags mit dem Neptune ML-Befehl dataprocessing
  - Stoppen eines Datenverarbeitungsauftrags mit dem Neptune-ML-Befehl dataprocessing
  - Auflisten aktiver Datenverarbeitungsaufträge mit dem Neptune-ML-Befehl dataprocessing
- Modelltraining mit dem Befehl modeltraining
  - Erstellen eines Modelltrainingsauftrags mit dem Neptune-ML-Befehl modeltraining
  - Abrufen des Status eines Modelltrainingsauftrags mit dem Neptune-ML-Befehl modeltraining
  - Stoppen eines Modelltrainingsauftrags mit dem Neptune-ML-Befehl modeltraining
  - Auflisten aktiver Modelltrainingsaufträge mit dem Neptune-ML-Befehl modeltraining
- Modelltransformation mit dem Befehl modeltransform
  - Erstellen eines Modelltransformationsauftrags mit dem Neptune-ML-Befehl modeltransform
  - Abrufen des Status eines Modelltransformationsauftrags mit dem Neptune-ML-Befehl modeltransform
  - Stoppen eines Modelltransformationsauftrags mit dem Neptune-ML-Befehl modeltransform
  - Auflisten aktiver Modelltransformationsaufträge mit dem Neptune-ML-Befehl modeltransform
- Verwalten von Inferenzendpunkten mit dem Befehl endpoints
  - Erstellen eines Inferenzendpunkts mit dem Neptune ML-Befehl endpoints
  - Abrufen des Status eines Inferenzendpunkts mit dem Neptune-ML-Befehl endpoints
  - Löschen eines Instance-Endpunkts mit dem Neptune-ML-Befehl endpoints
  - Auflisten von Inferenzendpunkten mit dem Neptune ML-Befehl endpoints
- Neptune-ML-Management-API – Fehler und Ausnahmen

Datenverarbeitung mit dem Befehl **`dataprocessing`**

Mit dem Neptune-ML-Befehl `dataprocessing` können Sie einen Datenverarbeitungsauftrag erstellen, dessen Status überprüfen, ihn beenden oder alle aktiven Datenverarbeitungsaufträge auflisten.

Erstellen eines Datenverarbeitungsauftrags mit dem Neptune-ML-Befehl **`dataprocessing`**

Der Neptune-ML-Befehl `dataprocessing` zum Erstellen eines neuen Auftrags sieht in der Regel wie folgt aus:

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)"
```

Weitere Informationen finden Sie unter start-ml-data-processing-job in der AWS CLI Befehlsreferenz.

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**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

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)"
      }'
```

Ein Befehl zum Initiieren einer inkrementellen Neuverarbeitung sieht wie folgt aus:

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)"
```

Weitere Informationen finden Sie unter start-ml-data-processing-job in der AWS CLI Befehlsreferenz.

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**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

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)"
     }'
```

Parameter für die Erstellung eines **`dataprocessing`**-Auftrags

- **`id`** – (Optional) Eine eindeutige ID für den neuen Auftrag.

  Typ: Zeichenfolge. Standardwert: eine automatisch generierte UUID.

- **`previousDataProcessingJobId`** – (Optional) Die Auftrags-ID eines abgeschlossenen Datenverarbeitungsauftrags, der für eine frühere Version der Daten ausgeführt wurde.

  Typ: Zeichenfolge. Standardwert: keiner.

  Hinweis: Verwenden Sie diese Option für die inkrementelle Datenverarbeitung, um das Modell zu aktualisieren, wenn die Diagrammdaten geändert wurden (jedoch nicht, wenn Daten gelöscht wurden).

- **`inputDataS3Location`**— (Erforderlich) Die URI des Amazon S3 S3-Standorts, an den SageMaker KI die für die Ausführung des Datenverarbeitungsjobs erforderlichen Daten herunterladen soll.

  Typ: Zeichenfolge.

- **`processedDataS3Location`**— (Erforderlich) Die URI des Amazon S3 S3-Standorts, an dem SageMaker KI die Ergebnisse eines Datenverarbeitungsauftrags speichern soll.

  Typ: Zeichenfolge.

- **`sagemakerIamRoleArn`**— (Optional) Der ARN einer IAM-Rolle für die SageMaker KI-Ausführung.

  Typ: Zeichenfolge. Hinweis: Er muss in der DB-Cluster-Parametergruppe aufgelistet werden, andernfalls tritt ein Fehler auf.

- **`neptuneIamRoleArn`**— (Optional) Der Amazon-Ressourcenname (ARN) einer IAM-Rolle, die SageMaker KI übernehmen kann, um Aufgaben in Ihrem Namen auszuführen.

  Typ: Zeichenfolge. Hinweis: Er muss in der DB-Cluster-Parametergruppe aufgelistet werden, andernfalls tritt ein Fehler auf.

- **`processingInstanceType`** – (Optional) Der Typ der ML-Instance, die während der Datenverarbeitung verwendet wird. Der Speicher sollte groß genug für den verarbeiteten Datensatz sein.

  Typ: Zeichenfolge. Standardwert: Der kleinste `ml.r5`-Typ, dessen Arbeitsspeicher zehnmal größer als die Größe der exportierten Diagrammdaten auf der Festplatte ist.

  Hinweis: Neptune ML kann den Instance-Typ automatisch auswählen. Siehe Auswahl einer Instance für die Datenverarbeitung.

- **`processingInstanceVolumeSizeInGB`** – (Optional) Die Größe des Festplattenvolumens der verarbeitenden Instance. Da sowohl Eingabedaten als auch verarbeitete Daten auf der Festplatte gespeichert werden, muss das Volume groß genug für beide Datensätze sein.

  Typ: Ganzzahl. Standardwert: `0`.

  Hinweis: Wenn nicht angegeben oder 0, wählt Neptune ML die Volume-Größe automatisch auf Grundlage der Datengröße aus.

- **`processingTimeOutInSeconds`** – (Optional) Timeout in Sekunden für den Datenverarbeitungsauftrag.

Typ: Ganzzahl. Default: `86,400` (1 Tag).

- **`modelType`** – (Optional) Einer der beiden Modelltypen, die Neptune ML zurzeit unterstützt: heterogene Diagrammmodelle (`heterogeneous`) und Wissensdiagramme (`kge`).

  Typ: Zeichenfolge. Standardwert: keiner.

  Hinweis: Wenn nicht angegeben, wählt Neptune ML dem Modelltyp automatisch auf Grundlage der Daten aus.

- **`configFileName`** – (Optional) Eine Datenspezifikationsdatei, die das Laden der exportierten Diagrammdaten für das Training beschreibt. Die Datei wird automatisch vom Neptune-Export-Toolkit generiert.

  Typ: Zeichenfolge. Standardwert: `training-data-configuration.json`.

- **`subnets`**— (Optional) Die Subnetze in IDs der Neptune VPC.

  Typ: Auflistung von Zeichenfolgen. Standardwert: keiner.

- **`securityGroupIds`**— (Optional) Die VPC-Sicherheitsgruppe IDs.

  Typ: Auflistung von Zeichenfolgen. Standardwert: keiner.

- **`volumeEncryptionKMSKey`**— (Optional) Der Schlüssel AWS Key Management Service (AWS KMS), den SageMaker AI verwendet, um Daten auf dem Speichervolume zu verschlüsseln, das den ML-Compute-Instances zugeordnet ist, die den Verarbeitungsjob ausführen.

  Typ: Zeichenfolge. Standardwert: keiner.

- **`enableInterContainerTrafficEncryption`** – (Optional) Aktiviert oder deaktiviert die Verschlüsselung des Datenverkehrs zwischen Containern bei Trainings- oder Hyperparameter-Optimierungsaufträgen.

  Typ: boolescher Wert. Standardwert: True.

  > **Note**
  >
  > Der Parameter `enableInterContainerTrafficEncryption` ist nur in Engine-Version 1.2.0.2.R3 verfügbar.

- **`s3OutputEncryptionKMSKey`**— (Optional) Der Schlüssel AWS Key Management Service (AWS KMS), den SageMaker KI verwendet, um die Ausgabe des Trainingsjobs zu verschlüsseln.

Typ: Zeichenfolge. Standardwert: keiner.

Abrufen des Status eines Datenverarbeitungsauftrags mit dem Neptune ML-Befehl `dataprocessing`

Dies ist ein Beispiel für den Neptune-ML-Befehl `dataprocessing` für den Abruf des Status eines Auftrags:

AWS CLI

```
aws neptunedata get-ml-data-processing-job \
  --endpoint-url https://your-neptune-endpoint:port \
  --id "(the job ID)"
```

Weitere Informationen finden Sie unter get-ml-data-processing-job in der AWS CLI Befehlsreferenz.

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**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

curl

```
curl -s \
  "https://your-neptune-endpoint:port/ml/dataprocessing/(the job ID)" \
  | python -m json.tool
```

Parameter für den Abruf des Status des **dataprocessing**-Auftrags

- **id** – (Erforderlich) Die eindeutige ID des Datenverarbeitungsauftrags.

  Typ: Zeichenfolge.
- **neptuneIamRoleArn**— (Optional) Der ARN einer IAM-Rolle, die Neptune Zugriff auf SageMaker KI- und Amazon S3 S3-Ressourcen bietet.

  Typ: Zeichenfolge. Hinweis: Er muss in der DB-Cluster-Parametergruppe aufgelistet werden, andernfalls tritt ein Fehler auf.

Stoppen eines Datenverarbeitungsauftrags mit dem Neptune-ML-Befehl `dataprocessing`

Dies ist ein Beispiel für den Neptune-ML-Befehl `dataprocessing` für das Stoppen eines Auftrags:

AWS CLI

```
aws neptunedata cancel-ml-data-processing-job \
  --endpoint-url https://your-neptune-endpoint:port \
  --id "(the job ID)"
```

Um auch Amazon S3 S3-Artefakte zu bereinigen:

```
aws neptunedata cancel-ml-data-processing-job \
```

```
  --endpoint-url https://your-neptune-endpoint:port \
  --id "(the job ID)" \
  --clean
```

Weitere Informationen finden Sie unter cancel-ml-data-processing-job in der AWS CLI Befehlsreferenz.

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
```

Um auch Amazon S3 S3-Artefakte zu bereinigen:

```
awscurl "https://your-neptune-endpoint:port/ml/dataprocessing/(the job ID)?
clean=true" \
  --region us-east-1 \
  --service neptune-db \
  -X DELETE
```

> **Note**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

curl

```
curl -s \
  -X DELETE "https://your-neptune-endpoint:port/ml/dataprocessing/(the job ID)"
```

Oder:

```
curl -s \
  -X DELETE "https://your-neptune-endpoint:port/ml/dataprocessing/(the job ID)?
clean=true"
```

Parameter für das Stoppen eines **`dataprocessing`**-Auftrags

- **`id`** – (Erforderlich) Die eindeutige ID des Datenverarbeitungsauftrags.

  Typ: Zeichenfolge.
- **`neptuneIamRoleArn`**— (Optional) Der ARN einer IAM-Rolle, die Neptune Zugriff auf SageMaker KI- und Amazon S3 S3-Ressourcen bietet.

  Typ: Zeichenfolge. Hinweis: Er muss in der DB-Cluster-Parametergruppe aufgelistet werden, andernfalls tritt ein Fehler auf.
- **`clean`** – (Optional) Dieses Flag gibt an, dass alle Amazon-S3-Artefakte gelöscht werden sollen, wenn der Auftrag gestoppt wird.

  Typ: boolescher Wert. Standardwert: `FALSE`.

Auflisten aktiver Datenverarbeitungsaufträge mit dem Neptune-ML-Befehl **`dataprocessing`**

Dies ist ein Beispiel für den Neptune-ML-Befehl `dataprocessing` für das Auflisten aktiver Aufträge:

AWS CLI

```
aws neptunedata list-ml-data-processing-jobs \
  --endpoint-url https://your-neptune-endpoint:port
```

Um die Anzahl der Ergebnisse zu begrenzen:

```
aws neptunedata list-ml-data-processing-jobs \
  --endpoint-url https://your-neptune-endpoint:port \
  --max-items 3
```

Weitere Informationen finden Sie unter list-ml-data-processing-jobs in der AWS CLI Befehlsreferenz.

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
```

Um die Anzahl der Ergebnisse zu begrenzen:

```
awscurl "https://your-neptune-endpoint:port/ml/dataprocessing?maxItems=3" \
```

```
  --region us-east-1 \
  --service neptune-db \
  -X GET
```

> **Note**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

curl

```
curl -s "https://your-neptune-endpoint:port/ml/dataprocessing"
```

Oder:

```
curl -s "https://your-neptune-endpoint:port/ml/dataprocessing?maxItems=3"
```

Parameter für das Auflisten von **`dataprocessing`**-Aufträgen

- **`maxItems`** – (Optional) Die maximale Anzahl der Elemente, die zurückgegeben werden sollen.

  Typ: Ganzzahl. Standardwert: `10`. Maximal zulässiger Wert: `1024`.
- **`neptuneIamRoleArn`**— (Optional) Der ARN einer IAM-Rolle, die Neptune Zugriff auf SageMaker KI- und Amazon S3 S3-Ressourcen bietet.

  Typ: Zeichenfolge. Hinweis: Er muss in der DB-Cluster-Parametergruppe aufgelistet werden, andernfalls tritt ein Fehler auf.

Modelltraining mit dem Befehl **`modeltraining`**

Mit dem Neptune-ML-Befehl `modeltraining` können Sie einen Modelltrainingsauftrag erstellen, seinen Status überprüfen, ihn beenden oder alle aktiven Modelltrainingsaufträge auflisten.

Erstellen eines Modelltrainingsauftrags mit dem Neptune-ML-Befehl **`modeltraining`**

Ein Neptune-ML-Befehl `modeltraining` zum Erstellen eines vollständig neuen Auftrags sieht in der Regel wie folgt aus:

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"
```

Weitere Informationen finden Sie unter start-ml-model-training-job in der AWS CLI Befehlsreferenz.

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**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

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"
      }'
```

Ein Neptune-ML-Befehl `modeltraining` zum Erstellen eines Aktualisierungsauftrags für das inkrementelle Modelltraining sieht wie folgt aus:

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)"
```

Weitere Informationen finden Sie unter start-ml-model-training-job in der AWS CLI Befehlsreferenz.

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**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

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)"
      }'
```

Ein Neptune-ML-Befehl `modeltraining` zum Erstellen eines neuen Auftrags mit einer vom Benutzer bereitgestellten benutzerdefinierten Modellimplementierung sieht wie folgt aus:

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)"
  }'
```

Weitere Informationen finden Sie unter start-ml-model-training-job in der AWS CLI Befehlsreferenz.

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**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

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)"
        }
```

```
    }'
```

Parameter für die Erstellung eines **`modeltraining`**-Auftrags

- **`id`** – (Optional) Eine eindeutige ID für den neuen Auftrag.

  Typ: Zeichenfolge. Standardwert: eine automatisch generierte UUID.
- **`dataProcessingJobId`** – (Erforderlich) Die Auftrags-ID des abgeschlossenen Datenverarbeitungsauftrags, der die Daten für das Training erstellt hat.

  Typ: Zeichenfolge.
- **`trainModelS3Location`** – (Erforderlich) Der Speicherort in Amazon S3, an dem die Modellartefakte gespeichert werden sollen.

  Typ: Zeichenfolge.
- **`previousModelTrainingJobId`** – (Optional) Die Auftrags-ID eines abgeschlossenen Modelltrainingsauftrags, den Sie anhand von aktualisierten Daten schrittweise aktualisieren möchten.

  Typ: Zeichenfolge. Standardwert: keiner.
- **`sagemakerIamRoleArn`**— (Optional) Der ARN einer IAM-Rolle für die SageMaker KI-Ausführung.

  Typ: Zeichenfolge. Hinweis: Er muss in der DB-Cluster-Parametergruppe aufgelistet werden, andernfalls tritt ein Fehler auf.
- **`neptuneIamRoleArn`**— (Optional) Der ARN einer IAM-Rolle, die Neptune Zugriff auf SageMaker KI- und Amazon S3 S3-Ressourcen bietet.

  Typ: Zeichenfolge. Hinweis: Er muss in der DB-Cluster-Parametergruppe aufgelistet werden, andernfalls tritt ein Fehler auf.
- **`modelName`** – (Optional) Der Modelltyp für das Training. Standardmäßig basiert das ML-Modell automatisch auf dem `modelType`, der in der Datenverarbeitung verwendet wird. Sie können hier jedoch einen anderen Modelltyp angeben.

  Typ: Zeichenfolge. Standardwert: `rgcn` für heterogene Diagramme und `kge` für Wissensdiagramme. Gültige Werte: für heterogene Graphen: `rgcn`. Für `kge`-Diagramme: `transe`, `distmult` oder `rotate`. Für eine benutzerdefinierte Modellimplementierung: `custom`.

- **`baseProcessingInstanceType`** – (Optional) Der Typ der ML-Instance, die zur Vorbereitung und Verwaltung des Trainings von ML-Modellen verwendet wird.

  Typ: Zeichenfolge. Hinweis: Diese CPU-Instance wird anhand der Speicheranforderungen für die Verarbeitung von Trainingsdaten und Trainingsmodell ausgewählt. Siehe Auswahl einer Instance für Modelltraining und Modelltransformation.

- **`trainingInstanceType`** – (Optional) Der Typ der ML-Instance, die für das Modelltraining verwendet wird. Alle Neptune-ML-Modelle unterstützen CPU-, GPU- und MultiGPU-Trainings.

  Typ: Zeichenfolge. Standardwert: `ml.p3.2xlarge`.

  Hinweis: Die Auswahl des richtigen Instance-Typs für ein Training ist von Aufgabentyp, Diagrammgröße und Budget abhängig. Siehe Auswahl einer Instance für Modelltraining und Modelltransformation.

- **`trainingInstanceVolumeSizeInGB`** – (Optional) Die Größe des Festplatten-Volumes der Trainings-Instance. Da sowohl Eingabedaten als auch das Ausgabemodell auf der Festplatte gespeichert werden, muss das Volume groß genug für beide Datensätze sein.

  Typ: Ganzzahl. Standardwert: `0`.

  Hinweis: Wenn nicht angegeben oder 0, wählt Neptune ML die Größe des Festplatten-Volumes anhand der im Datenverarbeitungsschritt generierten Empfehlung aus. Siehe Auswahl einer Instance für Modelltraining und Modelltransformation.

- **`trainingTimeOutInSeconds`** – (Optional) Timeout in Sekunden für den Trainingsauftrag.

  Typ: Ganzzahl. Default: `86,400` (1 Tag).

- **`maxHPONumberOfTrainingJobs`** – Maximale Gesamtzahl der Trainingsaufträge, die für den Hyperparameter-Optimierungsauftrag gestartet werden sollen.

  Typ: Ganzzahl. Standardwert: 2.

  Hinweis: Neptune ML optimiert automatisch die Hyperparameter des Machine-Learning-Modells. Um ein Modell mit guter Leistung zu erhalten, müssen Sie mindestens 10 Aufträge verwenden (mit anderen Worten, `maxHPONumberOfTrainingJobs` auf 10 festlegen). Im Allgemeinen gilt: Je mehr Optimierungsausführungen, desto besser die Ergebnisse.

- **`maxHPOParallelTrainingJobs`** – Maximale Anzahl der parallelen Trainingsaufträge, die für den Hyperparameter-Optimierungsauftrag gestartet werden sollen.

Typ: Ganzzahl. Standardwert: 2.

Hinweis: Die Anzahl der parallelen Aufträge, die Sie ausführen können, wird durch die verfügbaren Ressourcen auf Ihrer Trainings-Instance begrenzt.

- **`subnets`**— (Optional) Die Subnetze in IDs der Neptune VPC.

  Typ: Auflistung von Zeichenfolgen. Standardwert: keiner.

- **`securityGroupIds`**— (Optional) Die VPC-Sicherheitsgruppe IDs.

  Typ: Auflistung von Zeichenfolgen. Standardwert: keiner.

- **`volumeEncryptionKMSKey`**— (Optional) Der Schlüssel AWS Key Management Service (AWS KMS), den SageMaker KI verwendet, um Daten auf dem Speichervolume zu verschlüsseln, das den ML-Compute-Instances zugeordnet ist, die den Trainingsjob ausführen.

  Typ: Zeichenfolge. Standardwert: keiner.

- **`s3OutputEncryptionKMSKey`**— (Optional) Der Schlüssel AWS Key Management Service (AWS KMS), den SageMaker AI verwendet, um die Ausgabe des Verarbeitungsjobs zu verschlüsseln.

  Typ: Zeichenfolge. Standardwert: keiner.

- **`enableInterContainerTrafficEncryption`** – (Optional) Aktiviert oder deaktiviert die Verschlüsselung des Datenverkehrs zwischen Containern bei Trainings- oder Hyperparameter-Optimierungsaufträgen.

  Typ: boolescher Wert. Standardwert: True.

  > **Note**
  >
  > Der Parameter `enableInterContainerTrafficEncryption` ist nur in Engine-Version 1.2.0.2.R3 verfügbar.

- **`enableManagedSpotTraining`**— (Optional) Optimiert die Kosten für das Training von Machine-Learning-Modellen mit Amazon-Elastic-Compute-Cloud-Spot-Instances. Weitere Informationen finden Sie unter Managed Spot Training bei Amazon SageMaker.

  Typ: boolescher Wert. Standardwert: False.

- **`customModelTrainingParameters`** – (Optional) Die Konfiguration für das Training benutzerdefinierter Modelle. Dies ist ein JSON-Objekt mit den folgenden Feldern:

- **`sourceS3DirectoryPath`** – (Erforderlich) Der Pfad zum Amazon-S3-Speicherort des Python-Moduls, das Ihr Modell implementiert. Dieser Pfad muss auf einen gültigen, vorhandenen Amazon-S3-Speicherort verweisen, der mindestens ein Trainingsskript, ein Transformationsskript und die Datei `model-hpo-configuration.json` enthält.
- **`trainingEntryPointScript`** – (Optional) Der Name des Einstiegspunkts in Ihrem Modul für ein Skript, das Modelltrainings durchführt und Hyperparameter als Befehlszeilenargumente verwendet, einschließlich fester Hyperparameter.

  Standardwert: `training.py`.
- **`transformEntryPointScript`** – (Optional) Der Name des Einstiegspunkts in Ihrem Modul für ein Skript, das ausgeführt werden soll, nachdem das beste Modell aus der Hyperparametersuche identifiziert wurde, um die für die Modellbereitstellung notwendigen Modellartefakte zu berechnen. Es sollte ohne Befehlszeilenargumente ausgeführt werden können.

  Standardwert: `transform.py`.
- **`maxWaitTime`** – (Optional) Die maximale Wartezeit in Sekunden, wenn ein Modelltraining mit Spot-Instances durchgeführt wird. Der Wert sollte größer als `trainingTimeOutInSeconds` sein.

  Typ: Ganzzahl.

Abrufen des Status eines Modelltrainingsauftrags mit dem Neptune-ML-Befehl `modeltraining`

Dies ist ein Beispiel für den Neptune-ML-Befehl `modeltraining` für den Abruf des Status eines Auftrags:

AWS CLI

```
aws neptunedata get-ml-model-training-job \
  --endpoint-url https://your-neptune-endpoint:port \
  --id "(the job ID)"
```

Weitere Informationen finden Sie unter get-ml-model-training-job in der AWS CLI Befehlsreferenz.

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**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

curl

```
curl -s \
  "https://your-neptune-endpoint:port/ml/modeltraining/(the job ID)" \
  | python -m json.tool
```

Parameter für den Abruf des Status des **`modeltraining`**-Auftrags

- **`id`** – (Erforderlich) Die eindeutige ID des Modelltrainingsauftrags.

  Typ: Zeichenfolge.
- **`neptuneIamRoleArn`**— (Optional) Der ARN einer IAM-Rolle, die Neptune Zugriff auf SageMaker KI- und Amazon S3 S3-Ressourcen bietet.

Typ: Zeichenfolge. Hinweis: Er muss in der DB-Cluster-Parametergruppe aufgelistet werden, andernfalls tritt ein Fehler auf.

Stoppen eines Modelltrainingsauftrags mit dem Neptune-ML-Befehl **`modeltraining`**

Dies ist ein Beispiel für den Neptune-ML-Befehl `modeltraining` für das Stoppen eines Auftrags:

AWS CLI

```
aws neptunedata cancel-ml-model-training-job \
  --endpoint-url https://your-neptune-endpoint:port \
  --id "(the job ID)"
```

Um auch Amazon S3 S3-Artefakte zu bereinigen:

```
aws neptunedata cancel-ml-model-training-job \
  --endpoint-url https://your-neptune-endpoint:port \
  --id "(the job ID)" \
  --clean
```

Weitere Informationen finden Sie unter cancel-ml-model-training-job in der AWS CLI Befehlsreferenz.

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
```

Um auch Amazon S3 S3-Artefakte zu bereinigen:

```
awscurl "https://your-neptune-endpoint:port/ml/modeltraining/(the job ID)?
clean=true" \
  --region us-east-1 \
  --service neptune-db \
  -X DELETE
```

> **Note**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

curl

```
curl -s \
  -X DELETE "https://your-neptune-endpoint:port/ml/modeltraining/(the job ID)"
```

Oder:

```
curl -s \
  -X DELETE "https://your-neptune-endpoint:port/ml/modeltraining/(the job ID)?
clean=true"
```

Parameter für das Stoppen eines **`modeltraining`**-Auftrags

- **`id`** – (Erforderlich) Die eindeutige ID des Modelltrainingsauftrags.

  Typ: Zeichenfolge.

- **`neptuneIamRoleArn`**— (Optional) Der ARN einer IAM-Rolle, die Neptune Zugriff auf SageMaker KI- und Amazon S3 S3-Ressourcen bietet.

  Typ: Zeichenfolge. Hinweis: Er muss in der DB-Cluster-Parametergruppe aufgelistet werden, andernfalls tritt ein Fehler auf.
- **`clean`** – (Optional) Dieses Flag gibt an, dass alle Amazon-S3-Artefakte gelöscht werden sollen, wenn der Auftrag gestoppt wird.

  Typ: boolescher Wert. Standardwert: `FALSE`.

Auflisten aktiver Modelltrainingsaufträge mit dem Neptune-ML-Befehl **`modeltraining`**

Dies ist ein Beispiel für den Neptune-ML-Befehl `modeltraining` für das Auflisten aktiver Aufträge:

AWS CLI

```
aws neptunedata list-ml-model-training-jobs \
  --endpoint-url https://your-neptune-endpoint:port
```

Um die Anzahl der Ergebnisse zu begrenzen:

```
aws neptunedata list-ml-model-training-jobs \
  --endpoint-url https://your-neptune-endpoint:port \
  --max-items 3
```

Weitere Informationen finden Sie unter list-ml-model-training-jobs in der AWS CLI Befehlsreferenz.

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
```

Um die Anzahl der Ergebnisse zu begrenzen:

```
awscurl "https://your-neptune-endpoint:port/ml/modeltraining?maxItems=3" \
  --region us-east-1 \
  --service neptune-db \
  -X GET
```

> **Note**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

curl

```
curl -s "https://your-neptune-endpoint:port/ml/modeltraining" | python -m json.tool
```

Oder:

```
curl -s "https://your-neptune-endpoint:port/ml/modeltraining?maxItems=3" | python -m
 json.tool
```

Parameter für das Auflisten von **`modeltraining`**-Aufträgen

- **`maxItems`** – (Optional) Die maximale Anzahl der Elemente, die zurückgegeben werden sollen.

Typ: Ganzzahl. Standardwert: `10`. Maximal zulässiger Wert: `1024`.

- **`neptuneIamRoleArn`**— (Optional) Der ARN einer IAM-Rolle, die Neptune Zugriff auf SageMaker KI- und Amazon S3 S3-Ressourcen bietet.

  Typ: Zeichenfolge. Hinweis: Er muss in der DB-Cluster-Parametergruppe aufgelistet werden, andernfalls tritt ein Fehler auf.

Modelltransformation mit dem Befehl **`modeltransform`**

Mit dem Neptune-ML-Befehl `modeltransform` können Sie einen Modelltransformationsauftrag erstellen, seinen Status überprüfen, ihn beenden oder alle aktiven Modelltransformationsaufträge auflisten.

Erstellen eines Modelltransformationsauftrags mit dem Neptune-ML-Befehl **`modeltransform`**

Ein Neptune-ML-Befehl `modeltransform` zum Erstellen eines inkrementellen Transformationsauftrags ohne erneutes Modelltraining sieht wie folgt aus:

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"
```

Weitere Informationen finden Sie unter start-ml-model-transform-job in der AWS CLI Befehlsreferenz.

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**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

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"
      }'
```

Ein Neptune `modeltransform` ML-Befehl zum Erstellen eines Jobs aus einem abgeschlossenen SageMaker AI-Trainingsjob sieht so aus:

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"
```

Weitere Informationen finden Sie unter start-ml-model-transform-job in der AWS CLI Befehlsreferenz.

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**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

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"
      }'
```

Ein Neptune-ML-Befehl `modeltransform` zum Erstellen eines Auftrags mit einer benutzerdefinierten Modellimplementierung sieht wie folgt aus:

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)"
  }'
```

Weitere Informationen finden Sie unter start-ml-model-transform-job in der AWS CLI Befehlsreferenz.

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**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

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)"
        }
      }'
```

Parameter für die Erstellung eines **`modeltransform`**-Auftrags

- **`id`** – (Optional) Eine eindeutige ID für den neuen Auftrag.

  Typ: Zeichenfolge. Standardwert: eine automatisch generierte UUID.

- **`dataProcessingJobId`** – Die Auftrags-ID eines abgeschlossenen Datenverarbeitungsauftrags.

  Typ: Zeichenfolge.

Hinweis: Sie müssen entweder `dataProcessingJobId` und `mlModelTrainingJobId` oder `trainingJobName` angeben.

- **`mlModelTrainingJobId`** – Die Auftrags-ID eines abgeschlossenen Modelltrainingsauftrags.

  Typ: Zeichenfolge.

  Hinweis: Sie müssen entweder `dataProcessingJobId` und `mlModelTrainingJobId` oder `trainingJobName` angeben.

- **`trainingJobName`**— Der Name eines abgeschlossenen SageMaker KI-Schulungsjobs.

  Typ: Zeichenfolge.

  Hinweis: Sie müssen entweder die Parameter `dataProcessingJobId` und `mlModelTrainingJobId` oder den Parameter `trainingJobName` angeben.

- **`sagemakerIamRoleArn`**— (Optional) Der ARN einer IAM-Rolle für die SageMaker KI-Ausführung.

  Typ: Zeichenfolge. Hinweis: Er muss in der DB-Cluster-Parametergruppe aufgelistet werden, andernfalls tritt ein Fehler auf.

- **`neptuneIamRoleArn`**— (Optional) Der ARN einer IAM-Rolle, die Neptune Zugriff auf SageMaker KI- und Amazon S3 S3-Ressourcen bietet.

  Typ: Zeichenfolge. Hinweis: Er muss in der DB-Cluster-Parametergruppe aufgelistet werden, andernfalls tritt ein Fehler auf.

- **`customModelTransformParameters`** – (Optional) Informationen zur Konfiguration einer Modelltransformation anhand eines benutzerdefinierten Modells. Das `customModelTransformParameters`-Objekt enthält die folgenden Felder, deren Werte mit den gespeicherten Modellparametern aus dem Trainingsauftrag kompatibel sein müssen:

  - **`sourceS3DirectoryPath`** – (Erforderlich) Der Pfad zum Amazon-S3-Speicherort des Python-Moduls, das Ihr Modell implementiert. Dieser Pfad muss auf einen gültigen, vorhandenen Amazon-S3-Speicherort verweisen, der mindestens ein Trainingsskript, ein Transformationsskript und die Datei `model-hpo-configuration.json` enthält.
  - **`transformEntryPointScript`** – (Optional) Der Name des Einstiegspunkts in Ihrem Modul für ein Skript, das ausgeführt werden soll, nachdem das beste Modell aus der Hyperparametersuche identifiziert wurde, um die für die Modellbereitstellung notwendigen Modellartefakte zu berechnen. Es sollte ohne Befehlszeilenargumente ausgeführt werden können.

Standardwert: `transform.py`.

- **`baseProcessingInstanceType`** – (Optional) Der Typ der ML-Instance, die zur Vorbereitung und Verwaltung des Trainings von ML-Modellen verwendet wird.

  Typ: Zeichenfolge. Hinweis: Diese CPU-Instance wird anhand der Speicheranforderungen für die Verarbeitung von Transformationsdaten und Transformationsmodell ausgewählt. Siehe Auswahl einer Instance für Modelltraining und Modelltransformation.
- **`baseProcessingInstanceVolumeSizeInGB`** – (Optional) Die Größe des Festplatten-Volumes der Trainings-Instance. Da sowohl Eingabedaten als auch das Ausgabemodell auf der Festplatte gespeichert werden, muss das Volume groß genug für beide Datensätze sein.

  Typ: Ganzzahl. Standardwert: `0`.

  Hinweis: Wenn nicht angegeben oder 0, wählt Neptune ML die Größe des Festplatten-Volumes anhand der im Datenverarbeitungsschritt generierten Empfehlung aus. Siehe Auswahl einer Instance für Modelltraining und Modelltransformation.
- **`subnets`**— (Optional) Die Subnetze in IDs der Neptune VPC.

  Typ: Auflistung von Zeichenfolgen. Standardwert: keiner.
- **`securityGroupIds`**— (Optional) Die VPC-Sicherheitsgruppe IDs.

  Typ: Auflistung von Zeichenfolgen. Standardwert: keiner.
- **`volumeEncryptionKMSKey`**— (Optional) Der Schlüssel AWS Key Management Service (AWS KMS), den SageMaker KI verwendet, um Daten auf dem Speichervolume zu verschlüsseln, das den ML-Compute-Instances zugeordnet ist, die den Transformationsjob ausführen.

  Typ: Zeichenfolge. Standardwert: keiner.
- **`enableInterContainerTrafficEncryption`** – (Optional) Aktiviert oder deaktiviert die Verschlüsselung des Datenverkehrs zwischen Containern bei Trainings- oder Hyperparameter-Optimierungsaufträgen.

  Typ: boolescher Wert. Standardwert: True.

> **Note**
>
> Der Parameter `enableInterContainerTrafficEncryption` ist nur in Engine-Version 1.2.0.2.R3 verfügbar.

- **`s3OutputEncryptionKMSKey`**— (Optional) Der Schlüssel AWS Key Management Service (AWS KMS), den SageMaker AI verwendet, um die Ausgabe des Verarbeitungsjobs zu verschlüsseln.

  Typ: Zeichenfolge. Standardwert: keiner.

Abrufen des Status eines Modelltransformationsauftrags mit dem Neptune-ML-Befehl `modeltransform`

Dies ist ein Beispiel für den Neptune-ML-Befehl `modeltransform` für den Abruf des Status eines Auftrags:

AWS CLI

```
aws neptunedata get-ml-model-transform-job \
  --endpoint-url https://your-neptune-endpoint:port \
  --id "(the job ID)"
```

Weitere Informationen finden Sie unter get-ml-model-transform-job in der AWS CLI Befehlsreferenz.

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**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

curl

```
curl -s \
  "https://your-neptune-endpoint:port/ml/modeltransform/(the job ID)" \
  | python -m json.tool
```

Parameter für den Abruf des Status des **`modeltransform`**-Auftrags

- **`id`** – (Erforderlich) Die eindeutige ID des Modelltransformationsauftrags.

  Typ: Zeichenfolge.
- **`neptuneIamRoleArn`**— (Optional) Der ARN einer IAM-Rolle, die Neptune Zugriff auf SageMaker KI- und Amazon S3 S3-Ressourcen bietet.

  Typ: Zeichenfolge. Hinweis: Er muss in der DB-Cluster-Parametergruppe aufgelistet werden, andernfalls tritt ein Fehler auf.

Stoppen eines Modelltransformationsauftrags mit dem Neptune-ML-Befehl **`modeltransform`**

Dies ist ein Beispiel für den Neptune-ML-Befehl `modeltransform` für das Stoppen eines Auftrags:

AWS CLI

```
aws neptunedata cancel-ml-model-transform-job \
  --endpoint-url https://your-neptune-endpoint:port \
  --id "(the job ID)"
```

Um auch Amazon S3 S3-Artefakte zu bereinigen:

```
aws neptunedata cancel-ml-model-transform-job \
  --endpoint-url https://your-neptune-endpoint:port \
  --id "(the job ID)" \
  --clean
```

Weitere Informationen finden Sie unter cancel-ml-model-transform-job in der AWS CLI Befehlsreferenz.

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
```

Um auch Amazon S3 S3-Artefakte zu bereinigen:

```
awscurl "https://your-neptune-endpoint:port/ml/modeltransform/(the job ID)?
clean=true" \
  --region us-east-1 \
  --service neptune-db \
  -X DELETE
```

> **Note**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

curl

```
curl -s \
  -X DELETE "https://your-neptune-endpoint:port/ml/modeltransform/(the job ID)"
```

Oder:

```
curl -s \
  -X DELETE "https://your-neptune-endpoint:port/ml/modeltransform/(the job ID)?
clean=true"
```

Parameter für das Stoppen eines **`modeltransform`**-Auftrags

- **`id`** – (Erforderlich) Die eindeutige ID des Modelltransformationsauftrags.

  Typ: Zeichenfolge.
- **`neptuneIamRoleArn`**— (Optional) Der ARN einer IAM-Rolle, die Neptune Zugriff auf SageMaker KI- und Amazon S3 S3-Ressourcen bietet.

  Typ: Zeichenfolge. Hinweis: Er muss in der DB-Cluster-Parametergruppe aufgelistet werden, andernfalls tritt ein Fehler auf.
- **`clean`** – (Optional) Dieses Flag gibt an, dass alle Amazon-S3-Artefakte gelöscht werden sollen, wenn der Auftrag gestoppt wird.

  Typ: boolescher Wert. Standardwert: `FALSE`.

Auflisten aktiver Modelltransformationsaufträge mit dem Neptune-ML-Befehl **`modeltransform`**

Dies ist ein Beispiel für den Neptune-ML-Befehl `modeltransform` für das Auflisten aktiver Aufträge:

AWS CLI

```
aws neptunedata list-ml-model-transform-jobs \
  --endpoint-url https://your-neptune-endpoint:port
```

Um die Anzahl der Ergebnisse zu begrenzen:

```
aws neptunedata list-ml-model-transform-jobs \
  --endpoint-url https://your-neptune-endpoint:port \
  --max-items 3
```

Weitere Informationen finden Sie unter list-ml-model-transform-jobs in der AWS CLI Befehlsreferenz.

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
```

Um die Anzahl der Ergebnisse zu begrenzen:

```
awscurl "https://your-neptune-endpoint:port/ml/modeltransform?maxItems=3" \
```

```
  --region us-east-1 \
  --service neptune-db \
  -X GET
```

> **Note**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

curl

```
curl -s "https://your-neptune-endpoint:port/ml/modeltransform" | python -m json.tool
```

Oder:

```
curl -s "https://your-neptune-endpoint:port/ml/modeltransform?maxItems=3" | python -
m json.tool
```

Parameter für das Auflisten von **`modeltransform`**-Aufträgen

- **`maxItems`** – (Optional) Die maximale Anzahl der Elemente, die zurückgegeben werden sollen.

  Typ: Ganzzahl. Standardwert: `10`. Maximal zulässiger Wert: `1024`.
- **`neptuneIamRoleArn`**— (Optional) Der ARN einer IAM-Rolle, die Neptune Zugriff auf SageMaker KI- und Amazon S3 S3-Ressourcen bietet.

  Typ: Zeichenfolge. Hinweis: Er muss in der DB-Cluster-Parametergruppe aufgelistet werden, andernfalls tritt ein Fehler auf.

Verwalten von Inferenzendpunkten mit dem Befehl **endpoints**

Mit dem Neptune-ML-Befehl `endpoints` können Sie einen Inferenzendpunkt erstellen, seinen Status überprüfen, ihn löschen oder vorhandene Inferenzendpunkte auflisten.

Erstellen eines Inferenzendpunkts mit dem Neptune ML-Befehl **endpoints**

Ein Neptune-ML-Befehl `endpoints` zum Erstellen eines Inferenzendpunkts aus einem Modell, das durch einen Trainingsauftrag erstellt wurde, sieht wie folgt aus:

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)"
```

Weitere Informationen finden Sie create-ml-endpointin der AWS CLI Befehlsreferenz.

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**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

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)"
      }'
```

Ein Neptune-ML-Befehl `endpoints` zum Aktualisieren eines Inferenzendpunkts aus einem Modell, das durch einen Trainingsauftrag erstellt wurde, sieht wie folgt aus:

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)"
```

Weitere Informationen finden Sie create-ml-endpointin der AWS CLI Befehlsreferenz.

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**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

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)"
      }'
```

Ein Neptune-ML-Befehl `endpoints` zum Erstellen eines Inferenzendpunkts aus einem Modell, das durch einen Modelltransformationsauftrag erstellt wurde, sieht wie folgt aus:

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)"
```

Weitere Informationen finden Sie create-ml-endpointin der AWS CLI Befehlsreferenz.

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**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

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)"
      }'
```

Ein Neptune-ML-Befehl `endpoints` zum Aktualisieren eines Inferenzendpunkts aus einem Modell, das durch einen Modelltransformationsauftrag erstellt wurde, sieht wie folgt aus:

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)"
```

Weitere Informationen finden Sie create-ml-endpointin der AWS CLI Befehlsreferenz.

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**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

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)"
}'
```

Parameter für die Erstellung von Inferenzendpunkten mit **`endpoints`**

- **`id`** – (Optional) Eine eindeutige ID für den neuen Inferenzendpunkt.

  Typ: Zeichenfolge. Standard: Ein automatisch generierter Name mit Zeitstempel.

- **`mlModelTrainingJobId`** – Die Auftrags-ID des abgeschlossenen Modelltrainingsauftrags, der das Modell erstellt hat, auf das der Inferenzendpunkt verweisen wird.

  Typ: Zeichenfolge.

  Hinweis: Sie müssen entweder die `mlModelTrainingJobId` oder die `mlModelTransformJobId` angeben.

- **`mlModelTransformJobId`** – Die Auftrags-ID eines abgeschlossenen Modelltransformationsauftrags.

  Typ: Zeichenfolge.

  Hinweis: Sie müssen entweder die `mlModelTrainingJobId` oder die `mlModelTransformJobId` angeben.

- **`update`** – (Optional) Wenn vorhanden, gibt dieser Parameter an, dass es sich um eine Aktualisierungsanforderung handelt.

  Typ: boolesch. Standardwert: `false`

  Hinweis: Sie müssen entweder die `mlModelTrainingJobId` oder die `mlModelTransformJobId` angeben.

- **`neptuneIamRoleArn`**— (Optional) Der ARN einer IAM-Rolle, die Neptune Zugriff auf SageMaker KI- und Amazon S3 S3-Ressourcen bietet.

  Typ: Zeichenfolge. Hinweis: Er muss in der DB-Cluster-Parametergruppe aufgelistet werden, andernfalls tritt ein Fehler auf.

- **`modelName`** – (Optional) Modelltyp für das Training. Standardmäßig basiert das ML-Modell automatisch auf dem `modelType`, der in der Datenverarbeitung verwendet wird. Sie können hier jedoch einen anderen Modelltyp angeben.

Typ: Zeichenfolge. Standardwert: `rgcn` für heterogene Diagramme und `kge` für Wissensdiagramme. Gültige Werte: für heterogene Diagramme: `rgcn`. Für Wissensdiagramme: `kge`, `transe`, `distmult` oder `rotate`.

- **`instanceType`** – (Optional) Der Typ der ML-Instance, die für die Online-Bereitstellung verwendet wird.

  Typ: Zeichenfolge. Standardwert: `ml.m5.xlarge`.

  Hinweis: Die Auswahl der ML-Instance für einen Inferenzendpunkt ist von Aufgabentyp, Diagrammgröße und Budget abhängig. Siehe Auswahl einer Instance für einen Inferenzendpunkt.
- **`instanceCount`** – (Optional) Die Mindestanzahl von Amazon-EC2-Instances, die auf einem Endpunkt bereitgestellt werden sollen, um Vorhersagen zu erstellen.

  Typ: Ganzzahl. Standardwert: 1.
- **`volumeEncryptionKMSKey`**— (Optional) Der Schlüssel AWS Key Management Service (AWS KMS), den SageMaker KI verwendet, um Daten auf dem Speichervolume zu verschlüsseln, das an die ML-Compute-Instances angehängt ist, auf denen die Endgeräte ausgeführt werden.

  Typ: Zeichenfolge. Standardwert: keiner.

Abrufen des Status eines Inferenzendpunkts mit dem Neptune-ML-Befehl **`endpoints`**

Dies ist ein Beispiel für den Neptune-ML-Befehl `endpoints` für den Abruf des Status eines Instance-Endpunkts:

AWS CLI

```
aws neptunedata get-ml-endpoint \
  --endpoint-url https://your-neptune-endpoint:port \
  --id "(the inference endpoint ID)"
```

Weitere Informationen finden Sie get-ml-endpointin der AWS CLI Befehlsreferenz.

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**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

curl

```
curl -s \
  "https://your-neptune-endpoint:port/ml/endpoints/(the inference endpoint ID)" \
  | python -m json.tool
```

Parameter für den Abruf des Instance-Endpunktstatus mit **`endpoints`**

- **`id`** – (Erforderlich) Die eindeutige ID des Inferenzendpunkts.

  Typ: Zeichenfolge.
- **`neptuneIamRoleArn`**— (Optional) Der ARN einer IAM-Rolle, die Neptune Zugriff auf SageMaker KI- und Amazon S3 S3-Ressourcen bietet.

Typ: Zeichenfolge. Hinweis: Er muss in der DB-Cluster-Parametergruppe aufgelistet werden, andernfalls tritt ein Fehler auf.

Löschen eines Instance-Endpunkts mit dem Neptune-ML-Befehl **endpoints**

Dies ist ein Beispiel für den Neptune-ML-Befehl `endpoints` für das Löschen des Status eines Instance-Endpunkts:

AWS CLI

```
aws neptunedata delete-ml-endpoint \
  --endpoint-url https://your-neptune-endpoint:port \
  --id "(the inference endpoint ID)"
```

Um auch verwandte Artefakte zu bereinigen:

```
aws neptunedata delete-ml-endpoint \
  --endpoint-url https://your-neptune-endpoint:port \
  --id "(the inference endpoint ID)" \
  --clean
```

Weitere Informationen finden Sie delete-ml-endpointin der AWS CLI Befehlsreferenz.

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
```

Um auch verwandte Artefakte zu bereinigen:

```
awscurl "https://your-neptune-endpoint:port/ml/endpoints/(the inference endpoint
 ID)?clean=true" \
  --region us-east-1 \
  --service neptune-db \
  -X DELETE
```

> **Note**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

curl

```
curl -s \
  -X DELETE "https://your-neptune-endpoint:port/ml/endpoints/(the inference endpoint
 ID)"
```

Oder:

```
curl -s \
  -X DELETE "https://your-neptune-endpoint:port/ml/endpoints/(the inference endpoint
 ID)?clean=true"
```

Parameter zum Löschen eines Inferenzendpunkts mit **`endpoints`**

- **`id`** – (Erforderlich) Die eindeutige ID des Inferenzendpunkts.

Typ: Zeichenfolge.

- **`neptuneIamRoleArn`**— (Optional) Der ARN einer IAM-Rolle, die Neptune Zugriff auf SageMaker KI- und Amazon S3 S3-Ressourcen bietet.

  Typ: Zeichenfolge. Hinweis: Er muss in der DB-Cluster-Parametergruppe aufgelistet werden, andernfalls tritt ein Fehler auf.

- **`clean`** – (Optional) Gibt an, dass alle Artefakte, die sich auf diesen Endpunkt beziehen, ebenfalls gelöscht werden sollen.

  Typ: boolescher Wert. Standardwert: `FALSE`.

Auflisten von Inferenzendpunkten mit dem Neptune ML-Befehl **`endpoints`**

Dies ist ein Neptune-ML-Befehl `endpoints` zum Auflisten von Inferenzendpunkten:

AWS CLI

```
aws neptunedata list-ml-endpoints \
  --endpoint-url https://your-neptune-endpoint:port
```

Um die Anzahl der Ergebnisse zu begrenzen:

```
aws neptunedata list-ml-endpoints \
  --endpoint-url https://your-neptune-endpoint:port \
  --max-items 3
```

Weitere Informationen finden Sie list-ml-endpointsin der AWS CLI Befehlsreferenz.

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
```

Um die Anzahl der Ergebnisse zu begrenzen:

```
awscurl "https://your-neptune-endpoint:port/ml/endpoints?maxItems=3" \
  --region us-east-1 \
  --service neptune-db \
  -X GET
```

> **Note**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

curl

```
curl -s "https://your-neptune-endpoint:port/ml/endpoints" \
  | python -m json.tool
```

Oder:

```
curl -s "https://your-neptune-endpoint:port/ml/endpoints?maxItems=3" \
  | python -m json.tool
```

Parameter zum Auflisten von Inferenzendpunkten mit **`dataprocessing`**

- **`maxItems`** – (Optional) Die maximale Anzahl der Elemente, die zurückgegeben werden sollen.

Typ: Ganzzahl. Standardwert: `10`. Maximal zulässiger Wert: `1024`.

- **`neptuneIamRoleArn`**— (Optional) Der ARN einer IAM-Rolle, die Neptune Zugriff auf SageMaker KI- und Amazon S3 S3-Ressourcen bietet.

  Typ: Zeichenfolge. Hinweis: Er muss in der DB-Cluster-Parametergruppe aufgelistet werden, andernfalls tritt ein Fehler auf.

Neptune-ML-Management-API – Fehler und Ausnahmen

Alle Neptune-ML-Management-API-Ausnahmen geben einen HTTP-400-Code zurück. Nach Auslösung einer dieser Ausnahmen sollte der Befehl, der die Ausnahme generiert hat, nicht erneut ausgeführt werden.

- **`MissingParameterException`** – Fehlermeldung

  ```
  Required credentials are missing. Please add IAM role to the cluster or pass as a parameter to this request.
  ```

- **`InvalidParameterException`** – Fehlermeldungen:
  - `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`** – Fehlermeldungen:
  - `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.`

Neptune-ML-Einschränkungen

- Die zurzeit unterstützten Inferenztypen sind Knotenklassifikation, Knotenregression, Kantenklassifizierung, Kantenregression und Linkvorhersage (siehe Neptune-ML-Fähigkeiten).
- Die maximale Diagrammgröße, die Neptune ML unterstützen kann, ist von der Menge an Arbeitsspeicher und Speicherplatz abhängig, die für Datenvorbereitung, Modelltraining und Inferenz benötigt werden.
  - Die maximale Speichergröße einer SageMaker KI-Datenverarbeitungsinstanz beträgt 768 GB. Daher schlägt die Datenverarbeitungsphase fehl, wenn sie mehr als 768 GB Arbeitsspeicher benötigt.
  - Die maximale Speichergröße einer SageMaker KI-Trainingsinstanz beträgt 732 GB. Daher schlägt die Trainingsphase fehl, wenn sie mehr als 732 GB Arbeitsspeicher benötigt.
- Die maximale Größe einer Inferenz-Payload für einen SageMaker KI-Endpunkt beträgt 6 MiB. Daher schlägt eine induktive Inferenz fehl, wenn die Nutzlast des Unterdiagramms diese Größe überschreitet.
- Neptune ML ist derzeit nur in Regionen verfügbar, in denen Neptune und die anderen Dienste, von denen es abhängig ist (wie Amazon API Gateway und Amazon SageMaker AI) AWS Lambda, alle unterstützt werden.

  Es gibt Unterschiede in China (Peking) und China (Ningxia) im Zusammenhang mit der standardmäßigen Verwendung der IAM-Authentifizierung, wie hier beschrieben, zusammen mit anderen Unterschieden.
- Die von Neptune ML gestarteten Linkvorhersage-Inferenzendpunkte können derzeit nur mögliche Links mit Knoten vorhersagen, die während des Trainings im Diagramm vorhanden waren.

  Betrachten Sie beispielsweise ein Diagramm mit den Eckpunkten `User` und `Movie` sowie der Kante `Rated`. Mit einem entsprechenden Neptune-ML-Linkvorhersage-Empfehlungsmodell können Sie einen neuen Benutzer zum Diagramm hinzufügen und das Modell Filme für diesen vorhersagen lassen. Das Modell kann jedoch nur Filme empfehlen, die während des Modelltrainings vorhanden waren. Auch wenn die `User`-Knoteneinbettung in Echtzeit anhand des lokalen Unterdiagramms und des GNN-Modells berechnet wird und sich daher mit der Zeit ändern kann, wenn Benutzer Filme bewerten, wird sie für die endgültige Empfehlung mit den statischen, vorberechneten Filmeinbettungen verglichen.
- Die von Neptune ML unterstützten KGE-Modelle funktionieren nur für Linkvorhersageaufgaben und die Darstellungen sind für die Eckpunkte und Kantentypen spezifisch, die während des Trainings im Diagramm vorhanden sind. Daher müssen alle Eckpunkte und Kantentypen, auf die in einer

Inferenzabfrage verwiesen wird, während des Trainings im Diagramm vorhanden gewesen sein. Es können keine Vorhersagen für neue Kantentypen oder Eckpunkte erstellt werden, ohne das Modell erneut zu trainieren.

SageMaker Einschränkungen der KI-Ressourcen

Abhängig von Ihren Aktivitäten und der Ressourcennutzung im Laufe der Zeit können Ihnen Fehlermeldungen angezeigt werden, die besagen, dass Sie Ihr Kontingent überschritten haben (ResourceLimitExceeded). Und Sie müssen Ihre SageMaker KI-Ressourcen skalieren. Folgen Sie den Schritten im Verfahren Beantragen einer Erhöhung des Servicekontingents für SageMaker Ressourcen auf dieser Seite, um eine Kontingenterhöhung beim AWS Support zu beantragen.

SageMaker Die Namen der KI-Ressourcen entsprechen den Neptune ML-Stufen wie folgt:

- Die SageMaker KI `ProcessingJob` wird bei Neptun-Datenverarbeitungs-, Modelltrainings- und Modelltransformationsjobs verwendet.
- Die SageMaker KI `HyperParameterTuningJob` wird bei Trainingsjobs mit Neptun-Modellen verwendet.
- Die SageMaker KI `TrainingJob` wird bei Trainingsjobs mit Neptun-Modellen verwendet.
- Die SageMaker KI `Endpoint` wird von Neptun-Inferenzendpunkten verwendet.

Überwachung der Amazon Neptune Neptune-Ressourcen

Amazon Neptune unterstützt verschiedene Methoden für die Überwachung der Leistung und Nutzung von Datenbanken:

- Instance-Status – Überprüfen Sie den Zustand der Graphdatenbank-Engine eines Neptune-Clusters, ermitteln Sie die installierte Version der Engine und rufen Sie über die Instance-Status-API weitere Informationen zur Instance ab.
- Diagrammübersicht-API – Mit der Diagrammübersicht-API können Sie schnell eine Übersicht über Größe und Inhalt Ihrer Diagrammdaten erhalten.

  > **Note**
  >
  > Da die Diagrammübersicht-API DFE-Statistiken nutzt, ist sie nur verfügbar, wenn Statistiken aktiviert sind, was bei den Instance-Typen T3 und T4g nicht der Fall ist.

- Amazon CloudWatch — Neptune sendet automatisch Messwerte an Alarme CloudWatch und unterstützt CloudWatch diese auch. Weitere Informationen finden Sie unter the section called “Benutzen CloudWatch”.
- Audit-Protokolldateien – Sie können Datenbankprotokolldateien über die Neptune-Konsole anzeigen, herunterladen oder beobachten. Weitere Informationen finden Sie unter the section called “Prüfprotokolle mit Neptune”.
- Protokolle in Amazon CloudWatch Logs veröffentlichen — Sie können einen Neptune-DB-Cluster so konfigurieren, dass Audit-Protokolldaten in einer Protokollgruppe in Amazon CloudWatch Logs veröffentlicht werden. Mit CloudWatch Logs können Sie die Protokolldaten in Echtzeit analysieren, Alarme erstellen und Metriken anzeigen und CloudWatch Logs verwenden, um Ihre Protokolldatensätze auf einem äußerst dauerhaften Speicher zu speichern. CloudWatch Siehe Neptun-Logs CloudWatch .
- AWS CloudTrail— Neptune unterstützt die API-Protokollierung mithilfe von. CloudTrail Weitere Informationen finden Sie unter the section called “Neptune-API-Aufrufe protokollieren mit AWS CloudTrail”.
- Ereignisse — Sehen Sie sich Neptune-Ereignisse in der Konsole oder API an und abonnieren Sie Ereignisbenachrichtigungen mit Amazon SNS. AWS CLI Weitere Informationen finden Sie unter the section called “Ereignisse”.

- Tagging – Sie können Tags verwenden, um Ihren Neptune-Ressourcen Metadaten hinzuzufügen und die Nutzung mit Tags nachzuverfolgen. Weitere Informationen finden Sie unter the section called “Neptun-Ressourcen taggen”.

Themen

- Prüfen des Zustands einer Neptune-Instance
- Überwachung von Neptune mit Amazon CloudWatch
- Verwenden von Prüfprotokollen mit Amazon-Neptune-Clustern
- Neptune Logs auf Amazon Logs veröffentlichen CloudWatch
- Amazon CloudWatch Logs für ein Neptune-Notizbuch aktivieren
- Verwenden der Amazon-Neptune-Protokollierung für langsame Abfragen
- Protokollieren Amazon Neptune Neptune-API-Aufrufen mit AWS CloudTrail
- Amazon Neptune Neptune-Ereignisse
- Taggen von Amazon Neptune Neptune-Ressourcen

Prüfen des Zustands einer Neptune-Instance

Amazon Neptune stellt einen Mechanismus für die Überprüfung des Status der Graphsatenbank auf dem Host bereit. Auf diesem Wege können Sie auch prüfen, ob Sie sich mit einer Instance verbinden können.

Um den Zustand einer Instance zu überprüfen und den DB-Cluster-Status abzurufen, verwenden Sie `curl`:

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

Ab Engine-Version 1.2.1.0.R6 können Sie stattdessen auch den folgenden CLI-Befehl verwenden:

```
aws neptunedata get-engine-status
```

Wenn die Instance fehlerfrei ist, gibt der `status`-Befehl ein JSON-Objekt mit den folgenden Feldern zurück:

- **`status`** – Auf `"healthy"` festgelegt, wenn die Instance keine Probleme hat.

Wenn die Instance nach einem Ausfall oder Neustart wiederhergestellt wird und weiter aktive Transaktionen vom letzten Herunterfahren des Servers ausgeführt werden, ist `status` auf `"recovery"` festgelegt.

- **`startTime`** – Auf die UTC-Zeit festgelegt, zu der der aktuelle Serverprozess gestartet wurde.
- **`dbEngineVersion`** – Auf die Version der Neptune-Engine festgelegt, die im DB-Cluster ausgeführt wird.

  Wenn diese Engine-Version nach der Veröffentlichung manuell gepatcht wurde, wird der Versionsnummer `"Patch-"` vorangestellt.
- **`role`** – Auf `"reader"` festgelegt, wenn die Instance eine Read Replica ist, oder auf `"writer"`, wenn die Instance die primäre Instance ist.
- **`dfeQueryEngine`** – Auf `"enabled"` festgelegt, wenn die DFE-Engine vollständig aktiviert ist, oder auf `viaQueryHint` (Standardwert), wenn die DFE-Engine nur für Abfragen verwendet wird, bei denen der Abfragehinweis `useDFE` auf `true` festgelegt ist (Standardwert ist `viaQueryHint`).
- **`gremlin`** – Enthält Informationen zur Gremlin-Abfragesprache im Cluster. Insbesondere enthält es ein `version` Feld, das die aktuelle TinkerPop Version angibt, die von der Engine verwendet wird.
- **`sparql`** – Enthält Informationen zur SPARQL-Abfragesprache im Cluster. Insbesondere ist das Feld `version` enthalten, das die die aktuelle SPARQL-Version angibt, die von der Engine verwendet wird.
- **`opencypher`** – Enthält Informationen zur openCypher-Abfragesprache im Cluster. Insbesondere ist das Feld `version` enthalten, das die openCypher-Version angibt, die von der Engine verwendet wird.
- **`labMode`** – Enthält die Labor-Modus-Einstellungen, die von der Engine verwendet werden. Dies ist eine selektive Liste von Labormodus-Einstellungen und nicht der vollständige Satz. Den gesamten verwendeten Satz finden Sie in der Cluster-Parametergruppe.
- **`rollingBackTrxCount`** – Wenn Transaktionen rückgängig gemacht werden, ist dieses Feld auf die Anzahl dieser Transaktionen festgelegt Wenn keine Transaktionen rückgängig gemacht werden, wird das Feld nicht angezeigt.
- **`rollingBackTrxEarliestStartTime`** – Auf die Startzeit der frühesten Transaktion festgelegt, die rückgängig gemacht wird wird. Wenn keine Transaktionen rückgängig gemacht werden, wird das Feld nicht angezeigt.
- **`features`** – Enthält Statusinformationen zu den im DB-Cluster aktivierten Funktionen:

- **`lookupCache`** – Der aktuelle Status für Nachschlage-Cache. Dieses Feld wird nur für R5d-Instance-Typen angezeigt, da dies die einzigen Instances sind, in denen ein Lookup-Cache vorhanden sein kann. Das Feld ist ein JSON-Objekt mit dem Format:

  ```
  "lookupCache": {
    "status": "current lookup cache status"
  }
  ```

  Für eine R5d-Instance:

  - Wenn der Lookup-Cache aktiviert ist, wird der Status als `"Available"` aufgelistet.
  - Wenn der Lookup-Cache deaktiviert ist, wird der Status als `"Disabled"` aufgelistet.
  - Wenn das Festplattenlimit für die Instance erreicht wurde, wird der Status als `"Read Only Mode - Storage Limit Reached"` aufgelistet.
- **`ResultCache`** – Der aktuelle Status für Zwischenspeichern von Abfrageergebnissen. Das Feld ist ein JSON-Objekt mit dem Format:

  ```
  "ResultCache": {
    "status": "current results cache status"
  }
  ```

  - Wenn der Ergebnis-Cache aktiviert wurde, wird der Status als `"Available"` aufgelistet.
  - Wenn der Cache deaktiviert ist, wird der Status als `"Disabled"` aufgelistet.
- **`IAMAuthentication`**— Gibt an, ob die AWS Identity and Access Management (IAM-) Authentifizierung in Ihrem DB-Cluster aktiviert wurde oder nicht:
  - Wenn die IAM-Authentifizierung aktiviert ist, wird der Status als `"enabled"` aufgelistet.
  - Wenn die IAM-Authentifizierung deaktiviert ist, wird der Status als `"disabled"` aufgelistet.
- **`Streams`** – Gibt an, ob für den DB-Cluster Neptune-Streams aktiviert sind oder nicht:
  - Wenn Streams aktiviert sind, wird der Status als `"enabled"` aufgelistet.
  - Wenn Streams deaktiviert sind, wird der Status als `"disabled"` aufgelistet.
- **`AuditLog`** – Gleich `enabled`, wenn Audit-Logs aktiviert sind, andernfalls `disabled`.
- **`SlowQueryLogs`** – Gleich `info` oder `debug`, wenn die Protokollierung für langsame Abfragen aktiviert ist, andernfalls `disabled`.
- **`QueryTimeout`** – Der Wert des Abfrage-Timeouts in Millisekunden.
- **`settings`** – Auf die Instance angewendete Einstellungen:

- **`clusterQueryTimeoutInMs`** – Der Wert des Abfrage-Timeouts in Millisekunden, der für den gesamten Cluster festgelegt wurde.
- **`SlowQueryLogsThreshold`** – Der Wert des Abfrage-Timeouts in Millisekunden, der für den gesamten Cluster festgelegt wurde.

- **`serverlessConfiguration`** – Serverless-Einstellungen für einen Cluster, wenn er als Serverless-Cluster ausgeführt wird:
  - **`minCapacity`** – Die kleinste Größe in Neptune Capacity Units (NCUs), auf die eine Serverless-Instance im DB-Cluster verkleinert werden kann.
  - **`maxCapacity`** – Die größte Größe in Neptune Capacity Units (NCUs), auf die eine Serverless-Instance im DB-Cluster vergrößert werden kann.

Beispielausgabe des Befehls Instanzstatus

Dies ist ein Beispiel für die Ausgabe des Instance-Status-Befehls (in diesem Fall für eine R5d-Instance):

```
{
  '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'
  }
}
```

Gibt es ein Problem mit der Instance, gibt der Statusbefehl den Fehlercode `HTTP 500` zurück. Wenn der Host nicht erreichbar ist, kommt es zu einer Zeitüberschreitung der Anforderung. Stellen Sie sicher, dass Sie über die Virtual Private Cloud (VPC) auf die Instance zugreifen und dass Ihre Sicherheitsgruppen Ihnen Zugriff darauf gewähren.

Überwachung von Neptune mit Amazon CloudWatch

Amazon Neptune und Amazon CloudWatch sind integriert, sodass Sie Leistungskennzahlen sammeln und analysieren können. Sie können diese Metriken mithilfe der CloudWatch Konsole, der AWS Command Line Interface (AWS CLI) oder der CloudWatch API überwachen.

CloudWatch ermöglicht es Ihnen auch, Alarme einzustellen, sodass Sie benachrichtigt werden können, wenn ein Metrikwert einen von Ihnen angegebenen Schwellenwert überschreitet. Sie können sogar CloudWatch Ereignisse einrichten, um im Falle eines Verstoßes Abhilfemaßnahmen zu ergreifen. Weitere Informationen zur Verwendung CloudWatch und zu Alarmen finden Sie in der CloudWatch Dokumentation.

Themen

- CloudWatch Daten anzeigen (Konsole)
- CloudWatch Daten anzeigen (AWS CLI)
- CloudWatch Daten anzeigen (API)
- Wird CloudWatch zur Überwachung der DB-Instance-Leistung in Neptune verwendet
- Neptun-Metriken CloudWatch
- CloudWatch Neptun-Dimensionen

CloudWatch Daten anzeigen (Konsole)

So zeigen Sie CloudWatch Daten für einen Neptun-Cluster an (Konsole)

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die CloudWatch Konsole unter. https://console.aws.amazon.com/cloudwatch/
2. Wählen Sie im Navigationsbereich Metriken aus.
3. Wählen Sie im Bereich Alle Metriken die Option Neptune und dann aus. DBClusterIdentifier
4. Führen Sie im oberen Bereich einen Bildlauf nach unten durch, um die vollständige Liste der Metriken für Ihren Cluster anzuzeigen. Die verfügbaren Neptune-Metrikoptionen erscheinen in der Liste Angezeigt.

Um eine einzelne Metrik im Ergebnisbereich aus- oder abzuwählen, aktivieren Sie das Kontrollkästchen neben dem Ressourcennamen und der Metrik. Am unteren Rand der Konsole werden Diagramme mit den Metriken für die ausgewählten Elemente angezeigt. Weitere Informationen zu CloudWatch Diagrammen finden Sie unter Graph Metrics im CloudWatch Amazon-Benutzerhandbuch.

CloudWatch Daten anzeigen (AWS CLI)

Um CloudWatch Daten für einen Neptun-Cluster anzuzeigen (AWS CLI)

1. Installieren Sie das. AWS CLI Anweisungen finden Sie im AWS Command Line Interface - Benutzerhandbuch.
2. Verwenden Sie den AWS CLI , um Informationen abzurufen. Die relevanten CloudWatch Parameter für Neptune sind unter aufgeführt. Neptun-Metriken CloudWatch

   Im folgenden Beispiel werden CloudWatch Metriken für die Anzahl der Gremlin-Anfragen pro Sekunde für den Cluster abgerufen. `gremlin-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
   ]]>
   ```

CloudWatch Daten anzeigen (API)

CloudWatch unterstützt auch eine `Query` Aktion, sodass Sie Informationen programmgesteuert anfordern können. Weitere Informationen finden Sie in der CloudWatch Query API-Dokumentation und in der Amazon CloudWatch API-Referenz.

Wenn für eine CloudWatch Aktion ein Parameter erforderlich ist, der spezifisch für die Neptun-Überwachung ist`MetricName`, verwenden Sie z. B. die unter aufgeführten Werte. Neptun-Metriken CloudWatch

Das folgende Beispiel zeigt eine CloudWatch Anfrage auf niedriger Ebene, bei der die folgenden Parameter verwendet werden:

- `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`

So sieht die CloudWatch Anfrage aus. Dies ist jedoch nur eine Veranschaulichung der Form der Anforderung. Sie müssen Ihre eigene Anforderung basierend auf Ihren Metriken und Ihrem Zeitrahmen erstellen.

```
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
```

Wird CloudWatch zur Überwachung der DB-Instance-Leistung in Neptune verwendet

Sie können CloudWatch Metriken in Neptune verwenden, um zu überwachen, was auf Ihren DB-Instances passiert, und die von der Datenbank beobachtete Länge der Abfragewarteschlange verfolgen. Die folgenden Metriken sind besonders nützlich:

- **`CPUUtilization`** – Zeigt den Prozentsatz der CPU-Auslastung an.
- **`VolumeWriteIOPs`**— Zeigt die durchschnittliche Anzahl von I/O Schreibvorgängen auf das Cluster-Volume auf der Festplatte an, die in Intervallen von 5 Minuten gemeldet wird.
- **`MainRequestQueuePendingRequests`** – Zeigt die Anzahl der Anforderungen in der Eingabewarteschlange an, deren Ausführung aussteht.

Sie können auch feststellen, wie viele Anforderungen auf dem Server ausstehen, indem Sie die Gremlin-Endpunkt-Statusabfrage mit dem Parameter `includeWaiting` verwenden. Dadurch erhalten Sie den Status aller wartenden Abfragen.

Die folgenden Indikatoren können Ihnen helfen, die Bereitstellungs- und Abfragestrategien für Neptune anzupassen, um Effizienz und Leistung zu verbessern:

- Konsistente Latenz, hoch`CPUUtilization`, hoch `VolumeWriteIOPs` und niedrig `MainRequestQueuePendingRequests` zusammen zeigen, dass der Server aktiv damit beschäftigt ist, gleichzeitige Schreibanforderungen mit einer nachhaltigen Geschwindigkeit und mit geringer I/O Wartezeit zu verarbeiten.
- Eine konsistente Latenz, niedrige Werte für `CPUUtilization`, niedrige Werte für `VolumeWriteIOPs` und keine `MainRequestQueuePendingRequests` zeigen überschüssige Kapazitäten für die Verarbeitung von Schreibanforderungen auf der primären DB-Instance an.
- Eine variable Latenz, hohe Werte für `CPUUtilization` und `VolumeWriteIOPs` und keine `MainRequestQueuePendingRequests` zeigen, dass mehr Aufträge gesendet werden, als der Server in einem bestimmten Intervall verarbeiten kann. Erwägen Sie, Batch-Anfragen zu erstellen oder ihre Größe zu ändern, um dieselbe Menge an Arbeit mit weniger Transaktionsaufwand zu erledigen, indem Sie die primäre Instanz and/or skalieren, um die Anzahl der Abfrage-Threads zu erhöhen, die Schreibanforderungen gleichzeitig verarbeiten können.

- Niedrig `CPUUtilization` und hoch `VolumeWriteIOPs` bedeuten, dass Abfrage-Threads darauf warten, dass die I/O Operationen auf der Speicherebene abgeschlossen sind. Wenn Sie variable Latenzen und einen Anstieg für `MainRequestQueuePendingRequests` feststellen, sollten Sie die Erstellung oder Änderung von Batch-Anforderungen in Betracht ziehen, um die gleiche Anzahl von Aufträgen mit einem geringeren Transaktionsaufwand auszuführen.

Neptun-Metriken CloudWatch

> **Note**
>
> Amazon Neptune sendet CloudWatch nur dann Messwerte an, wenn sie einen Wert ungleich Null haben.
> Die Aggregationsgranularität für alle Neptune-Metriken beträgt 5 Minuten.

Themen

- Neptun-Metriken CloudWatch
- CloudWatch Metriken, die jetzt in Neptune veraltet sind

Neptun-Metriken CloudWatch

In der folgenden Tabelle sind die CloudWatch Metriken aufgeführt, die Neptune unterstützt.

> **Note**
>
> Alle kumulativen Metriken werden bei jedem Serverneustart auf null zurückgesetzt, ob für Wartungsarbeiten, Neustart oder Wiederherstellung nach einem Absturz.

Neptun-Metriken CloudWatch

Metrik	Description	Time Interval	Instanzstatistik
`BackupRetentionPeriodStorageUsed`	Der gesamte Sicherungsspeicher (in Byte), der zur Unterstützung aus		

Metrik	Description	Time Interval	Instanzstatistik
	dem Sicherungs-Aufbewahrungszeitraum des Neptune-DB-Clusters verwendet wird. Ist in dem von der `TotalBackupStorageBilled` -Metrik gemeldeten Gesamtwert enthalten.		
`BufferCacheHitRatio`	Der Prozentsatz der vom Buffer-Cache bedienten Anfragen. Diese Metrik kann bei der Diagnose der Abfragelatenz nützlich sein, da Cache-Fehler zu einer erheblichen Latenz führen. Wenn die Cache-Trefferquote unter 99,9 % liegt sollten Sie den Instance-Typ heraufstufen um mehr Daten im Arbeitsspeicher zwischenspeichern zu können.	1 Minute	Durchschnitt

Metrik	Description	Time Interval	Instanzstatistik
`ClusterReplicaLag`	Der Verzögerungszeitraum in Millisekunden für ein Read Replica, wenn Updates aus einer primären Instance repliziert werden.	1 Minute	Durchschnitt
`ClusterReplicaLagMaximum`	Der größte Verzögerungszeitraum in Millisekunden zwischen der primären Instance und jeder Neptune-DB-Instance im DB-Cluster.	1 Minute	max/min
`ClusterReplicaLagMinimum`	Der kleinste Verzögerungszeitraum in Millisekunden zwischen der primären Instance und jeder Neptune-DB-Instance im DB-Cluster.	1 Minute	max/min

Metrik	Description	Time Interval	Instanzstatistik
`CPUCredit Balance`	Die Anzahl der in 5-Minuten-Interval len gemeldeten CPU-Guthaben, die eine Instance angesammelt hat. Sie können diese Metrik verwenden, um zu bestimmen, wie lange eine DB-Instance das normale Leistungslevel bei gegebener Leistungs rate übersteigen kann.	5 Minuten	Durchschnitt
`CPUCreditUsage`	Die Anzahl der während des angegebenen Zeitraums verwendet en CPU-Guthaben (Meldung in 5-Minuten -Intervallen). Diese Metrik misst den Zeitraum, für den physikalische CPUs für die Verarbeitung von Instruktionen von den der DB-Instance zugeteilten virtuelle n CPUs verwendet wurden.	5 Minuten	Durchschnitt

Metrik	Description	Time Interval	Instanzstatistik
`CPUSurplusCreditBalance`	Die Anzahl überzähliger Guthaben, die von einer Unlimited-Instance verbraucht wurden, wenn ihr `CPUCreditBalance`-Wert null ist. Der `CPUSurplusCreditBalance`-Wert wird durch erworbene CPU-Guthaben abgezahlt. Wenn die Anzahl überzähliger Guthaben die Höchstzahl der Guthaben überschreitet, die die Instance in einem 24-Stunden-Zeitraum verdienen kann, fallen für die verbrauchten überzähligen Guthaben zusätzliche Gebühren an. Die Metriken für CPU-Guthaben sind nur mit einer fünfminütigen Frequenz verfügbar.	5 Minuten	Durchschnitt

Metrik	Description	Time Interval	Instanzstatistik
`CPUSurplusCreditsCharged`	Die Anzahl der verbrauchten überschüssigen Credits, die nicht durch verdiente CPU-Guthaben zurückgezahlt werden und für die eine zusätzliche Gebühr anfällt.	5 Minuten	Durchschnitt
`CPUUtilization`	Prozentsatz der CPU-Auslastung.	1 Minute	average/P99
`EngineUptime`	Der gesamte Zeitraum der Laufzeit der Instance in Sekunden.	1 Minute	Durchschnitt
`FreeableMemory`	Die Menge des verfügbaren Arbeitsspeichers (RAM-Speicher) in Bytes.	1 Minute	Durchschnitt
`GlobalDbDataTransferBytes`	Die Anzahl der Byte an Redo-Log-Daten, die AWS-Region in einer globalen Neptune-Datenbank von der primären AWS-Region zur sekundären Datenbank übertragen wurden.	1 Minute	Durchschnitt

Metrik	Description	Time Interval	Instanzstatistik
`GlobalDbReplicatedWriteIO`	Die Anzahl der I/O Schreibvorgänge, die von der Primärdatenbank in der globalen Datenbank auf das Cluster-Volume AWS-Region in einer Sekundärdatenbank repliziert wurden. AWS-Region Die Abrechnungsberechnungen für jeden DB-Cluster in einer globalen Neptune-Datenbank verwenden die `VolumeWriteIOPS` -Metrik, um die im Cluster ausgeführten Schreibvorgänge zu berücksichtigen. Für den primären DB-Cluster verwenden die Abrechnungsberechnungen `GlobalDbReplicatedWriteIO` zur Berücksichtigung der regionsübergreifenden Replikation zu sekundären DB-Clustern.	5 Minuten	sum

Metrik	Description	Time Interval	Instanzstatistik
`GlobalDbProgressLag`	Die Anzahl von Millisekunden, die ein sekundärer Cluster bei Benutzertransaktionen und Systemtransaktionen hinter dem primären Cluster zurückliegt.	1 Minute	Durchschnitt
`GremlinClientErrorsPerSec`	Anzahl der clientseitigen Fehler pro Sekunde in Gremlin-Transversalen.	1 Minute	Durchschnitt
`GremlinServerErrorsPerSec`	Anzahl der serverseitigen Fehler pro Sekunde in Gremlin-Transversalen.	1 Minute	Durchschnitt
`GremlinRequestsPerSec`	Anzahl der Anforderungen pro Sekunde an die Gremlin-Engine.	1 Minute	Durchschnitt
`GremlinWebSocketOpenConnections`	Die Anzahl der offenen WebSocket Verbindungen zu Neptune.	1 Minute	sum
`LoaderClientErrorsPerSec`	Anzahl der clientseitigen Fehler pro Sekunde von Loader-Anforderungen.	1 Minute	Durchschnitt
`LoaderRequestsPerSec`	Anzahl der Loader-Anforderungen pro Sekunde.	1 Minute	Durchschnitt

Metrik	Description	Time Interval	Instanzstatistik
`LoaderServerErrorsPerSec`	Anzahl der serverseitigen Loader-Fehler pro Sekunde.	1 Minute	Durchschnitt
`MainRequestQueuePendingRequests`	Die Anzahl der Anforderungen in der Eingabewarteschlange, deren Ausführung aussteht. Neptune beginnt mit der Drosselung von Anforderugnen, wenn sie die maximale Warteschlangenkapazität überschreiten.	1 Minute	sum

Metrik	Description	Time Interval	Instanzstatistik
`NCUUtilization`	Gilt nur für Neptune-Serverless-DB-Instances oder -DB-Cluster. Meldet auf Instance-Ebene einen Prozentsatz, berechnet als die Anzahl der Neptune Capacity Units (NCUs), die zurzeit von der betreffenden Instance verwendet werden, geteilt durch die Einstellung für die maximale NCU-Kapazität für den Cluster. Eine Neptune Capacity Unit (NCU) besteht aus einem Arbeitsspeicher (RAM) mit 2 GiB (Gibibyte) sowie der entsprechenden Kapazität des virtuellen Prozessors (vCPU) und des Netzwerks. `NCUUtilization` meldet auf Cluster-Ebene den Prozentsatz der maximalen Kapazität an, der vom gesamten Cluster verwendet wird.		

Metrik	Description	Time Interval	Instanzstatistik
`NetworkReceiveThroughput`	Die Menge des eingehenden Netzwerkdurchsatzes, den jede Instance im Neptune-DB-Cluster von Clients empfängt, in Byte pro Sekunde. Dieser Durchsatz enthält nicht den Netzwerkdatenverkehr zwischen Instances im DB-Cluster und dem Cluster-Volume.	1 Minute	Durchschnitt
`NetworkThroughput`	Die Menge des Netzwerkdurchsatzes in Byte pro Sekunde, der von jeder Instance im DB-Cluster von Clients empfangen und an Clients gesendet wird. Dieser Durchsatz enthält nicht den Netzwerkdatenverkehr zwischen Instances im DB-Cluster und dem Cluster-Volume.	1 Minute	Durchschnitt

Metrik	Description	Time Interval	Instanzstatistik
`NetworkTransmitThroughput`	Die Menge des ausgehenden Netzwerkdurchsatzes in Byte pro Sekunde, der von jeder Instance im Neptune-DB-Cluster an Clients gesendet wird. Dieser Durchsatz enthält nicht den Netzwerkdatenverkehr zwischen Instances im DB-Cluster und dem Cluster-Volume.	1 Minute	Durchschnitt
`NumIndexDeletesPerSec`	Anzahl der Löschungen aus einzelnen Indizes. Löschungen aus jedem Index werden einzeln gezählt. Dazu gehören auch die Löschungen, die rückgängig gemacht werden können, wenn bei einer Abfrage ein Fehler auftritt.	1 Minute	Durchschnitt

Metrik	Description	Time Interval	Instanzstatistik
`NumIndexInsertsPerSec`	Anzahl der Einfügungen in einzelne Indizes. Einfügungen in jeden Index werden separat gezählt. Dazu gehören auch die Einfügungen, die möglicherweise zurückgesetzt werden, wenn bei einer Abfrage ein Fehler auftritt.	1 Minute	Durchschnitt
`NumIndexReadsPerSec`	Anzahl der aus einem beliebigen Index gescannten Anweisungen. Jedes Zugriffsmuster beginnt mit einer Suche in einem Index und dem Lesen aller passenden Anweisungen. Eine Erhöhung dieser Metrik kann zu einer Erhöhung der Abfragelatenzen oder der CPU-Auslastung führen.	1 Minute	Durchschnitt
`NumQueuedRequestsPerSec`	Die Anzahl der Anfragen, die pro Sekunde in die Warteschlange gestellt werden.		

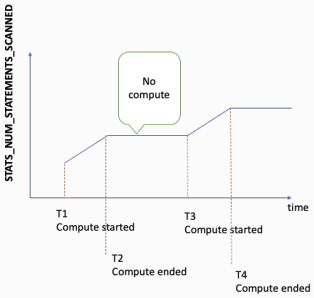
Metrik	Description	Time Interval	Instanzstatistik
NumResultCacheHit	Anzahl der Treffer im Gremlin-Ergebniscache.	1 Minute	sum
NumResultCacheMiss	Anzahl der Fehlschläge im Gremlin-Ergebniscache.	1 Minute	sum
NumTxCommitted	Die Anzahl der Transaktionen pro Sekunde, für die ein Commit durchgeführt wurde.	1 Minute	sum
NumTxOpened	Die Anzahl der pro Sekunde auf dem Server geöffneten Transaktionen.	1 Minute	sum
NumTxRolledBack	Bei Schreibabfragen die Anzahl der Transaktionen pro Sekunde, die auf dem Server aufgrund von Fehlern rückgängig gemacht wurden. Bei schreibgeschützten Abfragen entspricht diese Metrik der Anzahl der abgeschlossenen schreibgeschützten Transaktionen pro Sekunde.	1 Minute	sum

Metrik	Description	Time Interval	Instanzstatistik
NumUndoPagesPurged	Diese Metrik gibt die Anzahl der gelöschten Batches an. Diese Kennzahl ist ein Indikator für den Fortschritt beim Löschen. Der Wert gilt 0 für Reader-Instances, und die Metrik gilt nur für die Writer-Instanz.	1 Minute	sum
OpenCypherRequestsPerSec	Anzahl der Anforderungen pro Sekunde (HTTPS und Bolt) an die openCypher-Engine.	1 Minute	Durchschnitt
OpenCypherBoltOpenConnections	Die Anzahl offener Bolt-Verbindungen mit Neptune.	1 Minute	sum
ResultCacheSizeInBytes	Geschätzte Gesamtgröße (in Byte) aller zwischengespeicherten Elemente im Gremlin-Ergebniscache.	1 Minute	sum
ResultCacheItemCount	Anzahl der Elemente im Gremlin-Ergebniscache.	1 Minute	sum

Metrik	Description	Time Interval	Instanzstatistik
`ResultCacheOldestItemTimestamp`	Der Zeitstempel des ältesten Elements, das im Gremlin-Ergebniscache zwischengespeichert wurde.	1 Minute	sum
`ResultCacheNewestItemTimestamp`	Der Zeitstempel des neuesten Elements, das im Gremlin-Ergebniscache zwischengespeichert wurde.		

Metrik	Description	Time Interval	Instanzstatistik
`ServerlessDatabaseCapacity`	Als Metrik auf Instance-Ebene gibt `ServerlessDatabaseCapacity` die aktuelle Instance-Kapazität einer bestimmten Serverless-Neptune-Instance in NCUs an. Eine Neptune Capacity Unit (NCU) besteht aus einem Arbeitsspeicher (RAM) mit 2 GiB (Gibibyte) sowie der entsprechenden Kapazität des virtuellen Prozessors (vCPU) und des Netzwerks. Als Metrik auf Cluster-Ebene meldet `ServerlessDatabaseCapacity` den Durchschnitt aller `ServerlessDatabaseCapacity` -Werte der DB-Instances im Cluster an.		

Metrik	Description	Time Interval	Instanzstatistik
`SnapshotStorageUsed`	Der gesamte Sicherungsspeicher, der von allen Snapshots für einen Neptune-DB-Cluster außerhalb des Sicherungs-Aufbewahrungszeitraums verwendet wird (in Byte). Ist in dem von der `TotalBackupStorageBilled` -Metrik gemeldeten Gesamtwert enthalten.	1 Minute	sum
`SparqlClientErrorsPerSec`	Die Anzahl der clientseitigen Fehler pro Sekunde in SPARQL-Abfragen.	1 Minute	Durchschnitt
`SparqlRequestsPerSec`	Die Anzahl der Anforderungen pro Sekunde an die SPARQL-Engine.	1 Minute	Durchschnitt
`SparqlServerErrorsPerSec`	Die Anzahl der SPARQL-Serverfehler pro Sekunde.	1 Minute	Durchschnitt

Metrik	Description	Time Interval	Instanzstatistik
`StatsNumStatementsScanned`	Die Gesamtzahl der Anweisungen, die seit dem Serverstart für DFE-Statistiken gescannt wurden. Bei jeder Auslösung einer Statistikberechnung nimmt diese Zahl zu. Wenn keine Berechnung stattfindet, bleibt sie jedoch konstant. Wenn Sie ein Zeitverlaufdiagramm erstellen, können Sie daher erkennen, wann die Berechnung stattgefunden hat und wann nicht:  Wenn Sie die Steigung des Diagramms in Zeiträumen betrachten, in denen die Metrik zunimmt, können Sie auch erkennen, wie	1 Minute	sum

Metrik	Description	Time Interval	Instanzstatistik
	schnell die Berechnung durchgeführt wurde. Wenn es keine solche Metrik gibt, bedeutet dies, dass das Statistik-Feature im DB-Cluster deaktiviert ist oder dass die verwendete Engine-Version nicht über das Statistik-Feature verfügt. Wenn der Metrikwert null ist, bedeutet dies, dass keine Statistikberechnung stattgefunden hat.		
`StorageNetworkReceiveThroughput`	Die Menge des Netzwerkdurchsatzes, den jede Instance im Neptune-DB-Cluster vom Speichersubsystem erhält.	1 Minute	Durchschnitt

Metrik	Description	Time Interval	Instanzstatistik
StorageNetworkThroughput	Die Menge des Netzwerkdurchsatzes, der von jeder Instance im Neptune-DB-Cluster vom Speichersubsystem empfangen und an das Speichersubsystem gesendet wird.	1 Minute	Durchschnitt
StorageNetworkTransmitThroughput	Die Menge des Netzwerkdurchsatzes, der von jeder Instance im Neptune DB-Cluster an das Speichersubsystem gesendet wird.	1 Minute	Durchschnitt
SwapUsage	Die Größe des verwendeten Auslagerungsbereichs.	1 Minute	sum
TempStorageIOPS	Die Anzahl der IOPS für Lese- und Schreibvorgänge auf dem lokalen Speicher, der an die Neptune-DB-Instance angeschlossen ist. Diese Metrik stellt eine Zählung dar und wird einmal pro Sekunde gemessen.		

Metrik	Description	Time Interval	Instanzstatistik
TempStorageThroughput	Die Datenmenge, die zum und vom lokalen Speicher übertragen wird, der der Neptune-DB-Instance zugeordnet ist. Diese Metrik wird in Byte angegeben und einmal pro Sekunde gemessen.		
TotalBackupStorageBilled	Der gesamte Sicherungsspeicher, der Ihnen für einen bestimmten Neptune-DB-Cluster berechnet wird (in Byte). Umfasst den Sicherungsspeicher gemessen an den Metriken SnapshotStorageUsed und BackupRetentionPeriodStorageUsed .	1 Tag	sum
TotalRequestsPerSec	Die Gesamtzahl der Anfragen pro Sekunde an den Server aus allen Quellen.	1 Minute	Durchschnitt

Metrik	Description	Time Interval	Instanzstatistik
`TotalClientErrorsPerSec`	Die Gesamtzahl der Anfragen pro Sekunde, die aufgrund von clientseitigen Problemen zu Fehlern führten.	1 Minute	Durchschnitt
`TotalServerErrorsPerSec`	Die Gesamtzahl der Anfragen pro Sekunde, die aufgrund interner Ausfälle auf dem Server zu Fehlern führten.	1 Minute	Durchschnitt

Metrik	Description	Time Interval	Instanzstatistik
`UndoLogListSize`	Die Anzahl der Undo-Protokolle in der Liste der Undo-Protokolle. Undo-Logs enthalten Datensätze zu Commit-Transaktionen, die ablaufen, wenn alle aktiven Transaktionen jünger als die Commit-Zeit sind. Die abgelaufenen Datensätze werden regelmäßig gelöscht. Das Löschen von Datensätzen für Löschvorgänge kann länger dauern als das Löschen von Datensätzen für andere Transaktionsarten. Das Löschen erfolgt ausschließlich über die Writer-Instance des DB-Clusters, sodass die Geschwindigkeit des Löschvorgangs vom Typ der Writer-Instance abhängig ist. Wenn der Wert für `UndoLogListSize`	1 Minute	sum

Metrik	Description	Time Interval	Instanzstatistik
	hoch ist und im DB-Cluster steigt, sollten Sie die Writer-In stance heraufstufen, um die Löschrate zu erhöhen. Wenn Sie von einer älteren Version auf Engine-Version `1.2.0.0` oder höher aktualisi eren, stellen Sie außerdem zunächst sicher`1.2.0.0`, dass der `UndoLogLi stSize` Wert unter einem bestimmten Schwellenwert liegt. Andernfalls wird der Patch zurückges etzt und schlägt fehl. Die Schwellenwerte hängen vom Instance-Typ ab: Das Standardl imit liegt bei 40.000 für 4xlarge- oder größere Instances und 10.000 für Instances, die kleiner als 4xlarge sind. Wenn Sie versuchen , einen Cluster zu aktualisieren, dessen `UndoLogListSize`		

Metrik	Description	Time Interval	Instanzstatistik
	Metrik über dem Limit liegt, wird der Patch-Vorgang zurückgesetzt, das Upgrade wird abgebrochen und ein Ereignis mit dem Grund wird auf der Cluster-Ereignisseite angezeigt. Diese Grenzwerte können sich aus betrieblichen Gründen ohne vorherige Warnung ändern. Da Engine-Versionen `1.2.0.0` und höher ein anderes Format für Undo-Logs verwenden, kann das Upgrade erst beginnen, nachdem Ihre vorherigen Undo-Logs vollständig gelöscht wurden und den geltenden Schwellenwert unterschritten haben. Weitere Informationen finden Sie unter Upgrade zu 1.2.0.0 oder höher.		

Metrik	Description	Time Interval	Instanzstatistik
`VolumeBytesLeftTotal`	Der verbleibende verfügbare Speicherplatz für das Cluster-Volume in Byte. Während das Cluster-Volume wächst, verringert sich dieser Wert. Wenn er Null erreicht, meldet der Cluster den Fehler „Nicht genügend Speicherplatz“. Wenn Sie feststellen möchten, ob sich Ihr Neptune-DB-Cluster seiner Größenbeschränkung nähert, ist dieser Wert einfacher und zuverlässiger zu überwachen als. `VolumeBytesUsed` `VolumeBytesLeftTotal` berücksichtigt den Speicherplatz, der für die interne Verwaltung und andere Zuweisungen verwendet wird, die sich nicht auf Ihre Speicherabrechnung auswirken.	1 Minute	Durchschnitt

Metrik	Description	Time Interval	Instanzstatistik
	Diese Metrik wird nur von der Writer-Instanz gemeldet. Verwenden Sie die Dimension auf Clusterebene (), um Unterbrechungen bei Failovers zu vermeiden . `DBCluster Identifier`		
`VolumeBytesUsed`	Die Gesamtmenge des Speichers, der Ihrem Neptune-DB-Cluster zugeteilt ist (in Byte). Dies ist die Speichermenge, die Ihnen in Rechnung gestellt wird. Dies ist die maximale Speichermenge, die Ihrem DB-Cluster zu einem beliebigen Zeitpunkt seines Bestehens zugewiesen ist, und nicht die Menge, die Sie derzeit verwenden (siehe Neptune-Speicher-Fakturierung).	5 Minuten	sum

Metrik	Description	Time Interval	Instanzstatistik
`VolumeReadIOPs`	Die Gesamtzahl der in Rechnung gestellten I/O Lesevorgänge von einem Cluster-Volume, angegeben in Intervallen von 5 Minuten. Berechnete Lesevorgänge werden auf Cluster-Volume-Ebene berechnet, aggregiert aus allen Instances im Neptune-DB-Cluster und in Intervallen von 5 Minuten gemeldet.	5 Minuten	sum
`VolumeWriteIOPs`	Die Gesamtzahl der I/O Festplattenschreibvorgänge auf das Cluster-Volume, die in Intervallen von 5 Minuten gemeldet wurden.	5 Minuten	sum

CloudWatch Metriken, die jetzt in Neptune veraltet sind

Die Verwendung dieser Neptune-Metriken ist jetzt nicht mehr möglich. Sie werden weiterhin unterstützt, können aber in Zukunft eliminiert werden, wenn neue und bessere Metriken verfügbar werden.

Metrik	Description
`GremlinHttp1xx`	Anzahl der HTTP 1xx-Antworten für den Gremlin-Endpunkt pro Sekunde.

Metrik	Description
	Wir empfehlen stattdessen die Verwendung der neuen kombinierten `Http1xx`-Metrik.
`GremlinHttp2xx`	Anzahl der HTTP 2xx-Antworten für den Gremlin-Endpunkt pro Sekunde. Wir empfehlen stattdessen die Verwendung der neuen kombinierten `Http2xx`-Metrik.
`GremlinHttp4xx`	Anzahl der HTTP 4xx-Fehler für den Gremlin-Endpunkt pro Sekunde. Wir empfehlen stattdessen die Verwendung der neuen kombinierten `Http4xx`-Metrik.
`GremlinHttp5xx`	Anzahl der HTTP 5xx-Fehler für den Gremlin-Endpunkt pro Sekunde. Wir empfehlen stattdessen die Verwendung der neuen kombinierten `Http5xx`-Metrik.
`GremlinErrors`	Anzahl der Fehler in Gremlin-Traversals.
`GremlinRequests`	Anzahl der Anforderungen an die Gremlin-Engine.
`GremlinWebSocketSuccess`	Anzahl erfolgreicher WebSocket Verbindungen zum Gremlin-Endpunkt pro Sekunde.
`GremlinWebSocketClientErrors`	Anzahl der WebSocket Client-Fehler auf dem Gremlin-Endpunkt pro Sekunde.
`GremlinWebSocketServerErrors`	Anzahl der WebSocket Serverfehler auf dem Gremlin-Endpunkt pro Sekunde.
`GremlinWebSocketAvailableConnections`	Anzahl der derzeit verfügbaren potenziellen WebSocket Verbindungen.

Metrik	Description
`Http100`	Anzahl der HTTP 100-Antworten für den Endpunkt pro Sekunde. Wir empfehlen stattdessen die Verwendung der neuen kombinierten `Http1xx`-Metrik.
`Http101`	Anzahl der HTTP 101-Antworten für den Endpunkt pro Sekunde. Wir empfehlen stattdessen die Verwendung der neuen kombinierten `Http1xx`-Metrik.
`Http1xx`	Anzahl der HTTP 1xx-Antworten für den Endpunkt pro Sekunde.
`Http200`	Anzahl der HTTP 200-Antworten für den Endpunkt pro Sekunde. Wir empfehlen stattdessen die Verwendung der neuen kombinierten `Http2xx`-Metrik.
`Http2xx`	Anzahl der HTTP 2xx-Antworten für den Endpunkt pro Sekunde.
`Http400`	Anzahl der HTTP 400-Fehler für den Endpunkt pro Sekunde. Wir empfehlen stattdessen die Verwendung der neuen kombinierten `Http4xx`-Metrik.
`Http403`	Anzahl der HTTP 403-Fehler für den Endpunkt pro Sekunde. Wir empfehlen stattdessen die Verwendung der neuen kombinierten `Http4xx`-Metrik.

Metrik	Description
`Http405`	Anzahl der HTTP 405-Fehler für den Endpunkt pro Sekunde. Wir empfehlen stattdessen die Verwendung der neuen kombinierten `Http4xx`-Metrik.
`Http413`	Anzahl der HTTP 413-Fehler für den Endpunkt pro Sekunde. Wir empfehlen stattdessen die Verwendung der neuen kombinierten `Http4xx`-Metrik.
`Http429`	Anzahl der HTTP 429-Fehler für den Endpunkt pro Sekunde. Wir empfehlen stattdessen die Verwendung der neuen kombinierten `Http4xx`-Metrik.
`Http4xx`	Anzahl der HTTP 4xx-Fehler für den Endpunkt pro Sekunde.
`Http500`	Anzahl der HTTP 500-Fehler für den Endpunkt pro Sekunde. Wir empfehlen stattdessen die Verwendung der neuen kombinierten `Http5xx`-Metrik.
`Http501`	Anzahl der HTTP 501-Fehler für den Endpunkt pro Sekunde. Wir empfehlen stattdessen die Verwendung der neuen kombinierten `Http5xx`-Metrik.
`Http5xx`	Anzahl der HTTP 5xx-Fehler für den Endpunkt pro Sekunde.
`LoaderErrors`	Anzahl der Fehler von Loader-Anforderungen.

Metrik	Description
`LoaderRequests`	Anzahl der Loader-Anforderungen.
`SparqlHttp1xx`	Anzahl der HTTP 1xx-Antworten für den SPARQL-Endpunkt pro Sekunde. Wir empfehlen stattdessen die Verwendung der neuen kombinierten `Http1xx`-Metrik.
`SparqlHttp2xx`	Anzahl der HTTP 2xx-Antworten für den SPARQL-Endpunkt pro Sekunde. Wir empfehlen stattdessen die Verwendung der neuen kombinierten `Http2xx`-Metrik.
`SparqlHttp4xx`	Anzahl der HTTP 4xx-Fehler für den SPARQL-Endpunkt pro Sekunde. Wir empfehlen stattdessen die Verwendung der neuen kombinierten `Http4xx`-Metrik.
`SparqlHttp5xx`	Anzahl der HTTP 5xx-Fehler für den SPARQL-Endpunkt pro Sekunde. Wir empfehlen stattdessen die Verwendung der neuen kombinierten `Http5xx`-Metrik.
`SparqlErrors`	Anzahl der Fehler in den SPARQL-Abfragen.
`SparqlRequests`	Anzahl der Anforderungen an die SPARQL-Engine.
`StatusErrors`	Anzahl der Fehler vom Statusendpunkt.
`StatusRequests`	Anzahl der Anforderungen an den Statusendpunkt.

CloudWatch Neptun-Dimensionen

Die Metriken für Amazon Neptune werden durch die Werte für das Konto, den Diagrammnamen oder die Operation qualifiziert. Sie können die CloudWatch Amazon-Konsole verwenden, um Neptun-Daten zusammen mit allen Dimensionen in der folgenden Tabelle abzurufen.

Dimension	Description
`DBInstanceIdentifier`	Filtert die angeforderten Daten für eine bestimmte Datenbank-Instance innerhalb eines Clusters.
`DBClusterIdentifier`	Filtert die angeforderten Daten für einen spezifischen Neptune-DB-Cluster.
`DBClusterIdentifier` , `EngineName`	Filtert die Daten nach dem Cluster. Der Name der Engine für alle Neptune-Instances ist `neptune`.
`DBClusterIdentifier` , `Role`	Filtert die von Ihnen angeforderten Daten für einen bestimmten Neptune-DB-Cluster und aggregiert die Metrik nach Instance-Rolle (). WRITER/READER Sie können beispielsweise Metriken für alle READER-Instances eines Clusters zusammenfassen.
`DBClusterIdentifier` , `SourceRegion`	Filtert die Daten nach dem primären Cluster in einer primären Region der globalen Datenbank.
`DatabaseClass`	Filtert die angeforderten Daten für alle Instances einer Datenbankklasse. Sie können beispielsweise Metriken für alle Instances der Datenbankklasse db.r4.large zusammenfassen.
`EngineName`	Der Name der Engine für alle Neptune-I nstances ist `neptune`.

Dimension	Description
`GlobalDbDBClusterIdentifier` , `SecondaryRegion`	Filtert die Daten nach dem sekundären Cluster einer angegebenen globalen Datenbank in einer sekundären Region.

Verwenden von Prüfprotokollen mit Amazon-Neptune-Clustern

Um die Aktivität in Amazon-Neptune-DB-Clustern zu überwachen, aktivieren Sie die Erfassung von Prüfprotokollen durch Festlegen eines DB-Cluster-Parameters. Wenn Prüfprotokolle aktiviert sind, können Sie sie verwenden, um eine beliebige Kombination von unterstützten Ereignissen zu protokollieren. Sie können die Prüfprotokolle ansehen oder herunterladen, um sie zu überprüfen.

Aktivieren von Neptune-Prüfprotokollen

Verwenden Sie den Parameter `neptune_enable_audit_log` zum Aktivieren (`1`) oder Deaktivieren (`0`) von Prüfprotokollen.

Legen Sie diesen Parameter in der Parametergruppe fest, die von Ihrem DB-Cluster verwendet wird. Sie können das unter beschriebene Verfahren verwenden, um den Parameter mithilfe von Bearbeiten einer DB-Cluster-Parametergruppe oder DB-Parametergruppe zu ändern AWS-Managementkonsole, oder den Befehl AWS CLI modify-db-cluster-parameter-group oder den API-Befehl verwenden, um den Parameter programmatisch zu ändern. ModifyDBClusterParameterGroup

Sie müssen nach der Änderung dieses Parameters die DB-Instances neu starten, um die Änderung anzuwenden.

Anzeigen von Neptune-Prüfprotokollen über die Konsole

Sie können Prüfprotokolle mithilfe der AWS-Managementkonsole-Konsole anzeigen und herunterladen. Wählen Sie auf der Seite Instances die DB-Instance aus, um ihre Details anzuzeigen, und scrollen Sie dann in den Abschnitt Logs (Protokolle).

Suchen Sie diese Datei im AbschnittLogs (Protokolle) und wählen Sie anschließend Download (Herunterladen) aus, um die Protokolldatei herunterzuladen.

Einzelheiten zu Neptune-Prüfprotokollen

UTF-8 Die Protokolldateien haben ein Format. Protokolle sind in verschiedene Dateien geschrieben, deren Anzahl je nach Größe der Instance variiert. Sie müssen eventuell alle Prüfprotokolle überprüfen, um die aktuellen Ereignisse zu sehen.

Protokolleinträge folgen keiner sequenziellen Reihenfolge. Sie können den `timestamp`-Wert verwenden, um sie zu sortieren.

Protokolldateien werden rotiert, wenn Sie eine Größe von 100 MB erreichen. Dieses Limit ist nicht konfigurierbar.

Die Prüfprotokolldateien enthalten die folgenden durch Komma getrennten Informationen in Zeilen, in der folgenden Reihenfolge:

Feld	Beschreibung
Zeitstempel	Der auf die Mikrosekunde genaue Unix-Zeitstempel für das protokollierte Ereignis.
ClientHost	Der Hostname oder die IP, von dem sich der Benutzer verbunden hat.
ServerHost	Der Hostname oder die IP der Instance, für die das Ereignis protokolliert wird.
ConnectionType	Der Verbindungstyp. Kann `Websocket` , `HTTP_POST` , `HTTP_GET` oder `Bolt` sein.

IAM-ARN der aufrufenden Entität	Der ARN des IAM-Benutzers oder der IAM-Rolle, mit dem die Anforderung signiert wurde. Leer, wenn die IAM-Authentifizierung deaktiviert ist. Das Format ist:
	`arn:`*`partition`*` :`*`service`*`:`*`region`*`:`*`account`*`:`*`resource`*
	Beispiel:
	`arn:aws:iam::123456789012:user/Anna`
	`arn:aws:sts::123456789012:assumed-role/AWSNeptuneNotebookRole/SageMaker`

Feld	Beschreibung
Auth Context	Enthält ein serialisiertes JSON-Objekt mit Authentifizierungsinformationen. Das Feld `authenticationSucceeded` ist `True`, wenn der Benutzer authentifiziert wurde. Leer, wenn die IAM-Authentifizierung deaktiviert ist.
HttpHeader	Die HTTP-Header-Informationen. Kann eine Abfrage enthalten. Leere Verbindungen für WebSocket und Bolt.
Nutzlast	Die Gremlin-, SPARQL- oder openCypher-Abfrage.

Neptune Logs auf Amazon Logs veröffentlichen CloudWatch

Sie können einen Neptune-DB-Cluster so konfigurieren, dass er Protokolldaten and/or mit langsamer Abfrage in einer Protokollgruppe in Amazon Logs veröffentlicht. CloudWatch Mit CloudWatch Logs können Sie die Protokolldaten in Echtzeit analysieren und sie zum Erstellen von Alarmen und CloudWatch zum Anzeigen von Metriken verwenden. Sie können CloudWatch Logs verwenden, um Ihre Protokolldatensätze in einem äußerst langlebigen Speicher zu speichern.

Um Audit-Logs in CloudWatch Logs zu veröffentlichen, müssen Audit-Logs explizit aktiviert werden (sieheAktivieren von Prüfprotokollen). Ebenso müssen Slow-Query-Logs explizit aktiviert werden, um CloudWatch Slow-Query-Logs in Logs zu veröffentlichen (siehe). Verwenden der Amazon-Neptune-Protokollierung für langsame Abfragen

> **Note**
>
> Achten Sie auf Folgendes:
>
> - Zusätzliche Gebühren fallen an, wenn Sie Protokolle in veröffentlichen. CloudWatch Einzelheiten finden Sie auf der CloudWatch Preisseite.
> - Sie können keine Logs für die Regionen China (Peking) oder China (Ningxia) in CloudWatch Logs veröffentlichen.
> - Wenn der Export von Protokolldaten deaktiviert ist, löscht Neptune keine vorhandenen Protokollgruppen oder Protokoll-Streams. Wenn der Export von Protokolldaten deaktiviert ist, bleiben vorhandene Protokolldaten je nach Aufbewahrung der CloudWatch Protokolle in Logs verfügbar, und es fallen weiterhin Gebühren für gespeicherte Audit-Protokolldaten an.

Sie können Protokollstreams und Protokollgruppen mithilfe der CloudWatch Logs-Konsole AWS CLI, der oder der CloudWatch Logs-API löschen.

Verwenden der Konsole zum Veröffentlichen von Neptune-Protokollen in Logs CloudWatch

Um Neptune-Logs von der Konsole aus in CloudWatch Logs zu veröffentlichen

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die Amazon Neptune Neptune-Konsole unter. https://console.aws.amazon.com/neptune/home
2. Wählen Sie im Navigationsbereich Datenbanken aus.
3. Wählen Sie den Neptune-DB-Cluster aus, für den die Protokolldaten veröffentlicht werden sollen.
4. Wählen Sie für Actions (Aktionen) die Option Modify (Ändern) aus.
5. Wählen Sie im Abschnitt Protokollexporte die Protokolle aus, die Sie in Logs veröffentlichen möchten CloudWatch .
6. Wählen Sie Weiter und dann auf der Übersichtsseite DB-Cluster ändern.

Verwenden der CLI zum Veröffentlichen von Neptune-Auditprotokollen in Logs CloudWatch

Sie können einen neuen DB-Cluster erstellen, der Audit-Logs in CloudWatch Logs veröffentlicht, indem Sie den AWS CLI `create-db-cluster` Befehl mit den folgenden Parametern verwenden:

```
aws neptune create-db-cluster \
    --region us-east-1  \
    --db-cluster-identifier my_db_cluster_id \
    --engine neptune \
    --enable-cloudwatch-logs-exports '["audit"]'
```

Sie können einen vorhandenen DB-Cluster so konfigurieren, dass er Audit-Logs in CloudWatch Logs veröffentlicht, indem Sie den AWS CLI `modify-db-cluster` Befehl mit den folgenden Parametern verwenden:

```
aws neptune modify-db-cluster \
    --region us-east-1  \
```

```
    --db-cluster-identifier my_db_cluster_id \
    --cloudwatch-logs-export-configuration '{"EnableLogTypes":["audit"]}'
```

Verwenden der CLI zum Veröffentlichen von Neptune-Protokollen für langsame Abfragen in Logs CloudWatch

Sie können auch einen neuen DB-Cluster erstellen, der Logs für langsame Abfragen in Logs veröffentlicht, CloudWatch indem Sie den AWS CLI `create-db-cluster` Befehl mit den folgenden Parametern verwenden:

```
aws neptune create-db-cluster \
    --region us-east-1  \
    --db-cluster-identifier my_db_cluster_id \
    --engine neptune \
    --enable-cloudwatch-logs-exports '["slowquery"]'
```

In ähnlicher Weise können Sie einen vorhandenen DB-Cluster so konfigurieren, dass er Logs mit langsamen Abfragen in Logs veröffentlicht, CloudWatch indem Sie den AWS CLI `modify-db-cluster` Befehl mit den folgenden Parametern verwenden:

```
aws neptune modify-db-cluster --region us-east-1  \
    --db-cluster-identifier my_db_cluster_id \
    --cloudwatch-logs-export-configuration '{"EnableLogTypes":["slowquery"]}'
```

Überwachung Neptune Neptune-Protokollereignissen in Amazon CloudWatch

Nachdem Sie Neptune Logs aktiviert haben, können Sie Protokollereignisse in Amazon CloudWatch Logs überwachen. Für den Neptune-DB-Cluster wird automatisch eine neue Protokollgruppe mit dem folgenden Präfix erstellt. Dabei repräsentiert *`cluster-name`* den Namen des DB-Clusters und *`log_type`* den Protokolltyp:

```
/aws/neptune/cluster-name/log_type
```

Wenn Sie beispielsweise die Exportfunktion so konfigurieren, dass das Prüfprotokoll für einen DB-Cluster namens `mydbcluster` eingefügt wird, werden die Protokolldaten in der Protokollgruppe `/aws/neptune/mydbcluster/audit` gespeichert.

Alle Auditereignisse aus allen DB-Instances in einem DB-Cluster werden von einer Protokollgruppe unter Verwendung verschiedener Protokollstreams erfasst.

Wenn bereits eine Protokollgruppe mit dem angegebenen Namen vorhanden ist, verwendet Neptune diese Protokollgruppe, um Protokolldaten für den Neptune-DB-Cluster zu exportieren. Sie können die automatische Konfiguration verwenden AWS CloudFormation, um beispielsweise Protokollgruppen mit vordefinierten Aufbewahrungsfristen für Protokolle, Metrikfiltern und Kundenzugriff zu erstellen. Andernfalls wird automatisch eine neue Protokollgruppe erstellt, wobei der standardmäßige Aufbewahrungszeitraum für Protokolle, „Niemals ablaufen“, in CloudWatch Logs verwendet wird.

Sie können die CloudWatch Logs-Konsole AWS CLI, die oder die CloudWatch Logs-API verwenden, um den Aufbewahrungszeitraum für Protokolle zu ändern. Weitere Informationen zur Änderung der Aufbewahrungsfristen für CloudWatch Protokolle in Logs finden Sie unter Ändern der Aufbewahrung von Protokolldaten in CloudWatch Logs.

Sie können die CloudWatch Logs-Konsole, die oder die CloudWatch Logs-API verwenden AWS CLI, um in den Protokollereignissen für einen DB-Cluster nach Informationen zu suchen. Weitere Informationen finden Sie unter Suchen und Filtern von Protokolldaten.

Amazon CloudWatch Logs für ein Neptune-Notizbuch aktivieren

CloudWatch Protokolle für Neptune-Notebooks sind standardmäßig deaktiviert. Gehen Sie zur Aktivierung wie folgt vor, beispielsweise zum Debuggen oder für andere Zwecke:

Verwendung der AWS-Managementkonsole um CloudWatch Logs für ein Neptune-Notizbuch zu aktivieren

1. Öffnen Sie die Amazon SageMaker AI-Konsole unter https://console.aws.amazon.com/sagemaker/.
2. Wählen Sie im Navigationsbereich auf der linken Seite Notebook und dann Notebook-Instances aus. Suchen Sie nach dem Namen des Neptune-Notebooks, für das Sie Protokolle aktivieren möchten.
3. Gehen Sie zur Detailseite, indem Sie den Namen der Notebook-Instance auswählen, der im Schritt oben genannt wurde.
4. Wenn die Notebook-Instance ausgeführt wird, klicken Sie oben rechts auf der Notebook-Detailseite auf die Schaltfläche Stoppen.
5. Unter Berechtigungen und Verschlüsselung gibt es ein Feld für den IAM-Rollen-ARN. Wählen Sie den Link in diesem Feld aus, um zur IAM-Rolle für diesen Notebook zu wechseln.

6. Erstellen Sie die folgende Richtlinie:

   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. Speichern Sie diese neue Richtlinie und fügen Sie sie an die IAM-Rolle aus Schritt 4 an.
8. Wählen Sie oben rechts auf der Detailseite der SageMaker AI-Notebook-Instanz die Option Start aus.
9. Sobald Protokolle erstellt werden, sollte der Link Protokolle anzeigen unterhalb des Felds Lifecycle-Konfiguration unten links auf der Detailseite im Abschnitt Notebook-Instance-Einstellungen angezeigt werden.

Wenn Ihr Notebook nicht gestartet werden kann, wird auf der Seite mit den Notebook-Details auf der SageMaker AI-Konsole eine Meldung angezeigt, dass der Start der Notebook-Instanz mehr als 5 Minuten gedauert hat. Die für dieses Problem relevanten CloudWatch Protokolle finden Sie unter dem Namen:*(your-notebook-name)*`/LifecycleConfigOnStart`.

Weitere Informationen finden Sie bei Bedarf unter SageMaker CloudWatch Amazon-Ereignisse bei Amazon protokollieren.

Verwenden der Amazon-Neptune-Protokollierung für langsame Abfragen

Das Identifizieren, Debuggen und Optimieren einer langsam ausgeführten Abfrage kann schwierig sein. Wenn die Neptune-Protokollierung für langsame Abfragen aktiviert ist, werden die Attribute aller Abfragen mit langer Laufzeit automatisch protokolliert, um diesen Vorgang zu vereinfachen.

> **Note**
>
> Slow-query Die Protokollierung wurde in der Neptune-Engine-Version 1.2.1.0 eingeführt.

Sie aktivieren die Protokollierung langsamer Abfragen mit dem DB-Cluster-Parameter neptune_enable_slow_query_log. Dieser Parameter ist standardmäßig auf `disabled` festgelegt. Wenn Sie ihn auf `info` oder `debug` festlegen, wird die Protokollierung für langsame Abfragen aktiviert. Die Einstellung `info` protokolliert einige nützliche Attribute für jede langsam ausgeführte Abfrage. Die Einstellung `debug` protokolliert hingegen alle verfügbaren Attribute.

Um den Schwellenwert für eine als langsam ausgeführte Abfrage betrachtete Abfrage festzulegen, verwenden Sie den DB-Cluster-Parameter neptune_slow_query_log_threshold, um die Anzahl der Millisekunden anzugeben, nach der eine Abfrage als langsam betrachtet und protokolliert wird, wenn die Protokollierung langsamer Abfragen aktiviert ist. Der Standardwert ist 5.000 Millisekunden (5 Sekunden).

Sie können diese DB-Cluster-Parameter im oder mithilfe des AWS-Managementkonsole Befehls AWS CLI modify-db-cluster-parameter-group oder der Verwaltungsfunktion festlegen. ModifyDBClusterParameterGroup

> **Note**
>
> Die Protokollierungsparameter für langsame Abfragen sind dynamisch. Das bedeutet, dass das Ändern ihrer Werte keinen Neustart des DB-Clusters erfordert oder verursacht.

Um Logs für langsame Abfragen anzuzeigen, finden Sie im AWS-Managementkonsole

Sie können Protokolle für langsame Abfragen wie folgt im anzeigen und herunterladen: AWS-Managementkonsole

Wählen Sie auf der Seite Instances eine DB-Instance aus und scrollen Sie dann zum Abschnitt Protokolle. Sie können dort eine Protokolldatei auswählen und dann Herunterladen auswählen, um sie herunterzuladen.

Dateien, die von der Neptune-Protokollierung für langsame Abfragen generiert werden

Die durch die Protokollierung für langsame Abfragen generierten Protokolldateien besitzen die folgenden Eigenschaften:

- Die Dateien sind kodiert als UTF-8.
- Abfragen und ihre Attribute werden im JSON-Format protokolliert.
- Null- und leere Attribute werden nicht protokolliert, außer `queryTime`-Daten.
- Protokolle umfassen mehrere Dateien, deren Anzahl von der Größe der Instance abhängig ist.
- Protokolleinträge folgen keiner sequenziellen Reihenfolge. Sie können ihre `timestamp`-Werte verwenden, um sie anzuordnen.
- Sie müssen möglicherweise alle Protokolldateien für langsame Abfragen überprüfen, um die neuesten Ereignisse anzuzeigen.
- Protokolldateien werden rotiert, wenn sie zusammen die Größe von 100 MiB erreichen. Dieses Limit ist nicht konfigurierbar.

Im **`Informationsmodus`** protokollierte Attribute abfragen

Die folgenden Attribute werden für langsame Abfragen protokolliert, wenn der DB-Cluster-Parameter `neptune_enable_slow_query_log` auf `info` festgelegt wurde:

Group (Gruppieren)	Attribut	Description
Anfrage ResponseMetadata	`requestId`	Anforderungs-ID der Abfrage.

Group (Gruppieren)	Attribut	Description
	`requestType`	Art der Anfrage, wie HTTP oder WebSocket.
	`responseStatusCode`	Statuscode der Abfragean twort, z. B. 200.
	`exceptionClass`	Ausnahmeklasse des Fehlers, der nach Ausführung der Abfrage zurückgegeben wurde.
queryStats	`query`	Abfragezeichenfolge.
	`queryFingerprint`	Fingerabdruck der Abfrage.
	`queryLanguage`	Abfragesprache wie Gremlin, SPARQL oder openCypher.
memoryStats	`allocatedPermits`	Berechtigungen, die der Abfrage zugeteilt sind.
	`approximateUsedMem oryBytes`	Ungefährer Arbeitsspeichernut zung durch die Abfrage während der Ausführung.
queryTimeStats	`startTime`	Startzeit der Abfrage (UTC).
	`overallRunTimeMs`	Gesamtlaufzeit der Abfrage in Millisekunden.
	`parsingTimeMs`	Analysezeit der Abfrage in Millisekunden.
	`waitingTimeMs`	Wartezeit für die Gremlin/ SPARQL/openCypher Abfragewarteschlange, in Millisekunden

Group (Gruppieren)	Attribut	Description
	`executionTimeMs`	Ausführungszeit der Abfrage in Millisekunden.
	`serializationTimeMs`	Serialisierungszeit der Abfrage in Millisekunden.
statementCounters	`read`	Anzahl der gelesenen Anweisungen.
	`written`	Anzahl der geschriebenen Anweisungen.
	`deleted`	Anzahl der gelöschten Anweisungen.
transactionCounters	`committed`	Anzahl der übergebenen Transaktionen.
	`rolledBack`	Anzahl der rückgängig gemachten Transaktionen.
vertexCounters	`added`	Anzahl der hinzugefügten Eckpunkte.
	`removed`	Anzahl der entfernten Eckpunkte.
	`propertiesAdded`	Anzahl der hinzugefügten Eckpunkteigenschaften.
	`propertiesRemoved`	Anzahl der entfernten Eckpunkteigenschaften.
edgeCounters	`added`	Anzahl der hinzugefügten Kanten.
	`removed`	Anzahl der entfernten Kanten.

Group (Gruppieren)	Attribut	Description
	`propertiesAdded`	Anzahl der hinzugefügten Kanteneigenschaften.
	`propertiesRemoved`	Anzahl der entfernten Kanteneigenschaften.
resultCache	`hitCount`	Anzahl der Ergebnis-Cache-Treffer.
	`missCount`	Anzahl der Ergebnis-Cache-Fehler.
	`putCount`	Anzahl der Ergebnis-Cache-Puts.
gleichzeitig ExecutionStats	`acceptedQueryCountAtStart`	Parallele Abfragen, die mit der aktuellen Abfrageausführung beim Start akzeptiert wurden.
	`runningQueryCountAtStart`	Parallele Abfragen, die mit der aktuellen Abfrageausführung beim Start ausgeführt wurden.
	`acceptedQueryCountAtEnd`	Parallele Abfragen, die mit der aktuellen Abfrageausführung am Ende akzeptiert wurden.
	`runningQueryCountAtEnd`	Parallele Abfragen, die mit der aktuellen Abfrageausführung am Ende ausgeführt wurden.
queryBatch	`queryProcessingBatchSize`	Batchgröße während der Abfrageverarbeitung.
	`querySerialisationBatchSize`	Batchgröße während der Abfrageserialisierung.

Im Debug-Modus protokollierte Attribute **`abfragen`**

Wenn der `neptune_enable_slow_query_log`-DB-Cluster-Parameter auf `debug` festgelegt wurde, werden zusätzlich zu den Attributen, die im `info`-Modus protokolliert werden, die folgenden Speicherzähler-Attribute protokolliert:

Attribut	Description
`statementsScannedInAllIndexes`	Zahl der in allen Indizes gescannten Anweisungen.
`statementsScannedSPOGIndex`	Zahl der im SPOG-Index gescannten Anweisungen.
`statementsScannedPOGSIndex`	Zahl der im POGS-Index gescannten Anweisungen.
`statementsScannedGPSOIndex`	Zahl der im GPSO-Index gescannten Anweisungen.
`statementsScannedOSGPIndex`	Zahl der im OSGP-Index gescannten Anweisungen.
`statementsScannedInChunk`	Zahl der zusammen im Block gescannten Anweisungen.
`postFilteredStatementScans`	Zahl der Anweisungen, die nach dem Filtern nach dem Scannen übrig sind.
`distinctStatementScans`	Zahl der eindeutigen gescannten Anweisungen.
`statementsReadInAllIndexes`	Zahl der gelesenen Anweisungen nach dem Scannen nach dem Filtern in allen Indizes.
`statementsReadSPOGIndex`	Zahl der gelesenen Anweisungen nach dem Scannen nach dem Filtern im SPOG-Index.
`statementsReadPOGSIndex`	Zahl der gelesenen Anweisungen nach dem Scannen nach dem Filtern im POGS-Index.

Attribut	Description
`statementsReadGPSOIndex`	Zahl der gelesenen Anweisungen nach dem Scannen nach dem Filtern im GPSO-Index.
`statementsReadOSGPIndex`	Zahl der gelesenen Anweisungen nach dem Scannen nach dem Filtern im OSGP-Index.
`accessPathSearches`	Anzahl der Zugriffspfadsuchen.
`fullyBoundedAccessPathSearches`	Anzahl der Suchen nach vollständig begrenzten Schlüsselzugriffspfaden.
`accessPathSearchedByPrefix`	Anzahl der Zugriffspfadsuchen nach Präfix.
`searchesWhereRecordsWereFound`	Anzahl der Suchen, bei denen 1 oder mehrere Datensätze ausgegeben wurden.
`searchesWhereRecordsWereNotFound`	Anzahl der Suchen, bei denen keine Datensätze ausgegeben wurden.
`totalRecordsFoundInSearches`	Gesamtzahl der bei allen Suchen gefundenen Datensätze.
`statementsInsertedInAllIndexes`	Anzahl der in allen Indizes eingefügten Anweisungen.
`statementsUpdatedInAllIndexes`	Anzahl der in allen Indizes aktualisierten Anweisungen.
`statementsDeletedInAllIndexes`	Anzahl der in allen Indizes gelöschten Anweisungen.
`predicateCount`	Anzahl der Prädikate.
`dictionaryReadsFromValueToIdTable`	Anzahl der Verzeichnislesevorgänge von der Wert- in die ID-Tabelle.
`dictionaryReadsFromIdToValueTable`	Anzahl der Verzeichnislesevorgänge von der ID- in die Wert-Tabelle.

Attribut	Description
`dictionaryWritesToValueToIdTable`	Anzahl der Verzeichnisschreibvorgänge von der Wert- in die ID-Tabelle.
`dictionaryWritesToIdToValueTable`	Anzahl der Verzeichnisschreibvorgänge von der ID- in die Wert-Tabelle.
`rangeCountsInAllIndexes`	Anzahl der Bereichszählungen in allen Indizes.
`deadlockCount`	Anzahl der Deadlocks in der Abfrage.
`singleCardinalityInserts`	Anzahl der ausgeführten Einfügungen mit einfacher Kardinalität.
`singleCardinalityInsertDeletions`	Anzahl der während Einfügungen mit einfacher Kardinalität gelöschten Anweisungen.
`sharedLocksWaitTimeMillis`	Anzahl der Millisekunden, die für das Warten auf gemeinsame Sperren aufgewendet wurden.
`exclusiveLocksWaitTimeMillis`	Anzahl der Millisekunden, die damit verbracht wurden, auf exklusive Sperren zu warten.

Beispiel für die Debug-Protokollierung einer langsamen Abfrage

Die Ausführung der folgenden Gremlin-Abfrage könnte länger dauern als der für langsame Abfragen festgelegte Schwellenwert:

```
gremlin=g.V().has('code','AUS').repeat(out().simplePath()).until(has('code','AGR')).path().by('
```

Wenn die Protokollierung langsamer Abfragen im Debug-Modus aktiviert wäre, würden die folgenden Attribute für die Abfrage in einem Format wie dem folgenden protokolliert:

```
{
  "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
  }
}
```

Protokollieren Amazon Neptune Neptune-API-Aufrufen mit AWS CloudTrail

Amazon Neptune ist in einen Service integriert AWS CloudTrail, der eine Aufzeichnung der Aktionen bereitstellt, die von einem Benutzer, einer Rolle oder einem AWS Service in Neptune ausgeführt wurden. CloudTrail erfasst API-Aufrufe für Neptune als Ereignisse, einschließlich Aufrufe von der Neptune-Konsole und von Codeaufrufen an die Neptune-APIs.

CloudTrail protokolliert nur Ereignisse für Neptune Management API-Aufrufe, z. B. das Erstellen einer Instanz oder eines Clusters. Wenn Sie Änderungen an Ihrem Graph prüfen möchten, können Sie dazu Audit-Protokolle verwenden. Weitere Informationen finden Sie unter Verwenden von Prüfprotokollen mit Amazon-Neptune-Clustern.

> **Important**
>
> Amazon Neptune Neptune-Konsolen AWS CLI- und API-Aufrufe werden als Aufrufe an die Amazon Relational Database Service (Amazon RDS) -API protokolliert.

Wenn Sie einen Trail erstellen, können Sie die kontinuierliche Übermittlung von CloudTrail Ereignissen an einen Amazon S3 S3-Bucket aktivieren, einschließlich Ereignissen für Neptune. Wenn Sie keinen Trail konfigurieren, können Sie die neuesten Ereignisse trotzdem in der CloudTrail Konsole im Ereignisverlauf anzeigen. Anhand der von gesammelten Informationen können Sie die Anfrage CloudTrail, die an Neptune gestellt wurde, die IP-Adresse, von der aus die Anfrage gestellt wurde, wer die Anfrage gestellt hat, wann sie gestellt wurde, und weitere Details ermitteln.

Weitere Informationen CloudTrail dazu finden Sie im AWS CloudTrail Benutzerhandbuch.

Neptun-Informationen in CloudTrail

CloudTrail ist in Ihrem AWS Konto aktiviert, wenn Sie das Konto erstellen. Wenn in Amazon Neptune Aktivitäten auftreten, wird diese Aktivität zusammen mit anderen AWS Serviceereignissen in der

CloudTrail Ereignishistorie in einem Ereignis aufgezeichnet. Sie können aktuelle Ereignisse in Ihrem AWS Konto ansehen, suchen und herunterladen. Weitere Informationen finden Sie unter Ereignisse mit CloudTrail Ereignisverlauf anzeigen.

Für eine fortlaufende Aufzeichnung der Ereignisse in Ihrem AWS Konto, einschließlich der Ereignisse für Neptune, erstellen Sie einen Trail. Ein Trail ermöglicht CloudTrail die Übermittlung von Protokolldateien an einen Amazon S3 S3-Bucket. Wenn Sie ein Trail in der Konsole anlegen, gilt dieser für alle Regionen. Der Trail protokolliert Ereignisse aus allen Regionen der AWS Partition und übermittelt die Protokolldateien an den von Ihnen angegebenen Amazon S3 S3-Bucket. Darüber hinaus können Sie andere AWS Dienste konfigurieren, um die in den CloudTrail Protokollen gesammelten Ereignisdaten weiter zu analysieren und darauf zu reagieren. Weitere Informationen finden Sie unter:

- Übersicht zum Erstellen eines Trails
- CloudTrail Unterstützte Dienste und Integrationen
- Konfiguration von Amazon SNS SNS-Benachrichtigungen für CloudTrail
- Empfangen von CloudTrail Protokolldateien aus mehreren Regionen und Empfangen von CloudTrail Protokolldateien von mehreren Konten

Wenn im Namen Ihres AWS Kontos über die Neptune-Konsole, die Neptune-Befehlszeilenschnittstelle oder die Neptune SDK-APIs eine Aktion ausgeführt wird, wird die Aktion als Aufrufe an die AWS CloudTrail Amazon RDS-API protokolliert. Wenn Sie beispielsweise die Neptune-Konsole verwenden, um eine DB-Instance zu ändern oder den Befehl AWS CLImodify-db-instance aufrufen, zeigt das AWS CloudTrail Protokoll einen Aufruf der Amazon RDS-API-Aktion ModifyDBInstance an. Eine Liste der Neptune-API-Aktionen, die protokolliert werden AWS CloudTrail, finden Sie in der Neptune-API-Referenz.

> **Note**
>
> AWS CloudTrail protokolliert nur Ereignisse für Neptune Management API-Aufrufe, z. B. das Erstellen einer Instanz oder eines Clusters. Wenn Sie Änderungen an Ihrem Graph prüfen möchten, können Sie dazu Audit-Protokolle verwenden. Weitere Informationen finden Sie unter Verwenden von Prüfprotokollen mit Amazon-Neptune-Clustern.

Jeder Ereignis- oder Protokolleintrag enthält Informationen zu dem Benutzer, der die Anforderung generiert hat. Die Identitätsinformationen unterstützen Sie bei der Ermittlung der folgenden Punkte:

- Gibt an, ob die Anforderung mit Root- oder IAM-Benutzer-Anmeldeinformationen ausgeführt wurde.
- Gibt an, ob die Anforderung mit temporären Sicherheitsanmeldeinformationen für eine Rolle oder einen Verbundbenutzer gesendet wurde.
- Ob die Anfrage von einem anderen AWS Dienst gestellt wurde.

Weitere Informationen finden Sie unter CloudTrail userIdentity-Element.

Neptune-Protokolldateieinträge verstehen

Ein Trail ist eine Konfiguration, die die Übertragung von Ereignissen als Protokolldateien an einen von Ihnen angegebenen Amazon S3 S3-Bucket ermöglicht. CloudTrail Protokolldateien enthalten einen oder mehrere Protokolleinträge. Ein Ereignis stellt eine einzelne Anforderung aus einer beliebigen Quelle dar und enthält Informationen über die angeforderte Aktion, Datum und Uhrzeit der Aktion, Anforderungsparameter usw. CloudTrail Protokolldateien sind kein geordneter Stack-Trace der öffentlichen API-Aufrufe, sodass sie nicht in einer bestimmten Reihenfolge angezeigt werden.

Das folgende Beispiel zeigt ein CloudTrail Protokoll für einen Benutzer, der einen Snapshot einer DB-Instance erstellt und diese Instance dann mit der Neptune-Konsole gelöscht hat. Die Konsole wird durch das Element `userAgent` identifiziert. Die angeforderten API-Aufrufe, die von der Konsole gemacht wurden (`CreateDBSnapshot` und `DeleteDBInstance`) sind im Element `eventName` in jedem Eintrag zu finden. Informationen zum Benutzer (`Alice`) finden Sie im `userIdentity`-Element.

```
{
  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"
    }
  }
  ]
}
```

Amazon Neptune Neptune-Ereignisse

Themen

- Ereigniskategorien und Ereignismeldungen in Amazon Neptune
- Neptun-Ereignisse anzeigen
- Abonnieren von Neptune-Ereignisbenachrichtigungen
- Verwalten von Abonnements für Neptune-Ereignisbenachrichtigungen

Amazon Neptune nutzt Amazon Simple Notification Service (Amazon SNS), um bei Neptune-Ereignissen Benachrichtigungen zu senden. Diese Benachrichtigungen können in jeder Form erfolgen, die von Amazon SNS für eine AWS Region unterstützt wird, z. B. eine E-Mail, eine Textnachricht oder ein Anruf an einen HTTP-Endpunkt.

Neptune gruppiert diese Ereignisse nach Kategorien, die Sie abonnieren können, um bei einem Ereignis in diesen Kategorien benachrichtigt zu werden. Sie können eine Ereigniskategorie für eine DB-Instance, einen DB-Cluster, einen DB-Snapshot, einen DB-Cluster-Snapshot oder eine DB-Parametergruppe abonnieren. Wenn Sie beispielsweise die Kategorie „Backup“ für eine bestimmte DB-Instance abonnieren, werden Sie benachrichtigt, sobald ein sicherungsbezogenes Ereignis auftritt, das sich auf Ihre DB-Instance auswirkt. Außerdem erhalten Sie eine Benachrichtigung, wenn ein Abonnement für Ereignisbenachrichtigungen geändert wird.

Ereignisse treten auf DB-Cluster- und DB-Instance-Ebene auf. Daher erhalten Sie Ereignisse, wenn Sie einen DB-Cluster oder eine DB-Instance abonnieren.

Ereignisbenachrichtigungen werden an die Adressen gesendet, die Sie bei Erstellung des Abonnements angegeben haben. Sie sollten mehrere verschiedene Abonnements erstellen, beispielsweise ein Abonnement für alle Ereignisbenachrichtigungen und ein anderes Abonnement für kritische Ereignisse bei Produktions-DB-Instances. Sie können die Benachrichtigungen ganz einfach deaktivieren, ohne ein Abonnement zu löschen. Hierzu legen Sie in der Neptune-Konsole das Optionsfeld Aktiviert auf Nein fest.

> **Important**
>
> Amazon Neptune garantiert nicht die Reihenfolge der Ereignisse, die in einem Ereignis-Stream gesendet werden. Die Reihenfolge der Ereignisse kann sich ändern.

Neptune verwendet den Amazon-Ressourcennamen (ARN) eines Amazon-SNS-Themas, um die einzelnen Abonnements zu ermitteln. Die Neptune-Konsole erstellt einen ARN für Sie, wenn Sie ein Abonnement erstellen.

Die Abrechnung Neptune-Ereignisbenachrichtigungen erfolgt über Amazon SNS. Bei Verwendung von Ereignisbenachrichtigungen fallen Amazon-SNS-Gebühren an. Weitere Informationen finden Sie unter Preise für den Amazon Simple Notification Service (SNS).

Ereigniskategorien und Ereignismeldungen in Amazon Neptune

Neptune generiert eine erhebliche Anzahl von Ereignissen in Kategorien, die Sie über die Neptune-Konsole abonnieren können. Jede Kategorie gilt für einen Quelltyp, der eine DB-Instance, ein DB-Snapshot oder eine DB-Parametergruppe sein kann.

> **Note**
>
> Neptune verwendet vorhandene Amazon-RDS-Ereignisdefinitionen und -IDs.

Neptune-Ereignisse aus DB-Instances

Die folgende Tabelle zeigt eine Liste von Ereignissen nach Ereigniskategorie, wenn der Quelltyp eine DB-Instance ist.

Kategorie	Amazon RDS-Ereignis-ID	Beschreibung
Verfügbarkeit	RDS-EVENT-0006	Die DB-Instance wurde neu gestartet.
	RDS-EVENT-0004	DB-Instance-Shutdown.
	RDS-EVENT-0022	Während des Neustarts der Neptune-Engine ist ein Fehler aufgetreten.

Kategorie	Amazon RDS-Ereignis-ID	Beschreibung
backup	RDS-EVENT-0001	Sichern einer DB-Instance
	RDS-EVENT-0002	DB-Instance-Sicherung wurde beendet.
Konfigurationsänderung	RDS-EVENT-0009	Die DB-Instance wurde einer Sicherheitsgruppe hinzugefügt.
	RDS-EVENT-0024	Die DB-Instance wird in eine Multi-AZ DB-Instance konvertiert.
	RDS-EVENT-0030	Die DB-Instance wird in eine Single-AZ DB-Instance konvertiert.
	RDS-EVENT-0012	Die Änderung wird auf die Datenbank-Instance-Klasse angewendet.
	RDS-EVENT-0018	Die derzeitigen Speichereinstellungen für diese DB-Instance werden zurzeit geändert.
	RDS-EVENT-0011	Eine Parametergruppe für diese DB-Instance wurde geändert.

Kategorie	Amazon RDS-Ereignis-ID	Beschreibung
	RDS-EVENT-0092	Eine Parameter gruppe für diese DB-Instance wurde aktualisiert.
	RDS-EVENT-0028	Automatische Backups für diese DB-Instance wurden deaktiviert.
	RDS-EVENT-0032	Automatische Backups für diese DB-Instance wurden aktiviert.
	RDS-EVENT-0025	Die DB-Instance wurde in eine Multi-AZ DB-Instance konvertie rt.
	RDS-EVENT-0029	Die DB-Instance wurde in eine Single-AZ DB-Instance konvertiert.
	RDS-EVENT-0014	Die DB-Instance-Klasse für diese DB-Instance wurde geändert.
	RDS-EVENT-0017	Die Speichereinstellun gen für diese DB-Instance wurden geändert.

Kategorie	Amazon RDS-Ereignis-ID	Beschreibung
	RDS-EVENT-0010	Die DB-Instance wurde aus einer Sicherheitsgruppe entfernt.
Erstellung	RDS-EVENT-0005	Die DB-Instance wurde erstellt.
Löschung	RDS-EVENT-0003	Die DB-Instance wurde gelöscht.
Failover	RDS-EVENT-0034	Neptune führt das angeforderte Failover nicht aus, da erst vor Kurzem ein Failover auf dieser DB-Instanz erfolgt ist.
	RDS-EVENT-0013	Ein Multi-AZ Failover, das zur Heraufstufung einer Standby-Instance führte, wurde gestartet.
	RDS-EVENT-0015	Ein Multi-AZ Failover, das zur Heraufstufung einer Standby-Instance geführt hat, ist abgeschlossen. Es kann einige Minuten dauern, bis die DNS-Übertragung auf die neue primäre DB-Instance abgeschlossen ist.

Kategorie	Amazon RDS-Ereignis-ID	Beschreibung
	RDS-EVENT-0065	Die Instance wurde nach einem partielle n Failover wiederher gestellt.
	RDS-EVENT-0049	Ein Multi-AZ Failover wurde abgeschlossen.
	RDS-EVENT-0050	Nach einer erfolgrei chen Instanzwi ederherstellung wurde eine Multi-AZ Aktivierung gestartet.
	RDS-EVENT-0051	Eine Multi-AZ Aktivierung ist abgeschlossen. Sie sollten nun Zugriff auf Ihre Datenbank haben.
	RDS-EVENT-0031	Die DB-Instance ist aufgrund einer inkompatiblen Konfiguration oder eines zugrunde liegenden Speicherp roblems ausgefall en. Führen Sie eine zeitpunktbezogene Wiederherstellung für die DB-Instance aus.

Kategorie	Amazon RDS-Ereignis-ID	Beschreibung
	RDS-EVENT-0036	Die DB-Instance befindet sich in einem inkompatiblen Netzwerk. Einige der angegebenen Subnetz-IDs sind ungültig oder nicht vorhanden.
	RDS-EVENT-0035	Einige Parameter der DB-Instance sind ungültig. Wenn beispielsweise die DB-Instance nicht gestartet werden konnte, da der Wert eines speicherb ezogenen Parameter s zu hoch für diese Instance-Klasse ist, sollte der Kunde den Speicherparameter ändern und die DB-Instance neu starten.
	RDS-EVENT-0082	Neptune konnte die Sicherungsdaten aus einem Amazon-S3-Bucket nicht kopieren. Wahrscheinlich sind die Berechtig ungen für Neptune zum Zugriff auf den Amazon-S3-Bucket falsch konfiguriert.

Kategorie	Amazon RDS-Ereignis-ID	Beschreibung
Wenig Speicherplatz	RDS-EVENT-0089	Die DB-Instance hat mehr als 90% ihres zugewiesenen Speichers verbrauch t. Sie können den Speicherplatz für eine DB-Instan ce mit der Metrik Freier Speicherplatz überwachen.
	RDS-EVENT-0007	Der Speicherplatz, der für die DB-Instance zugewiesen wurde, ist aufgebraucht. Sie sollten der DB-Instan ce weiteren Speicher zuordnen, um dieses Problem zu lösen.
Wartung	RDS-EVENT-0026	Die Offline-Wartung der DB-Instance wird gerade durchgeführt. Die DB-Instance ist zurzeit nicht verfügbar .
	RDS-EVENT-0027	Die Offline-Wartung der DB-Instance ist abgeschlossen. Die DB-Instance ist nun verfügbar.

Kategorie	Amazon RDS-Ereignis-ID	Beschreibung
	RDS-EVENT-0047	Patchen der DB-Instance wurde abgeschlossen.
Benachrichtigung	RDS-EVENT-0044	Operator-issued Benachrichtigung. Weitere Informationen finden Sie in der Ereignismeldung.
	RDS-EVENT-0048	Patchen der DB-Instance wurde verzögert.
	RDS-EVENT-0087	Die DB-Instance wurde gestoppt.
	RDS-EVENT-0088	Die DB-Instance wurde gestartet.
	RDS-EVENT-0154	Die DB-Instance wird gestartet, da sie die maximal zulässige Anhaltezeit überschritten hat.
	RDS-EVENT-0158	Die DB-Instance befindet sich in einem Zustand, der nicht aktualisiert werden kann.
	RDS-EVENT-0173	Die DB-Instance wurde gepatcht.

Kategorie	Amazon RDS-Ereignis-ID	Beschreibung
Read Replica	RDS-EVENT-0045	Bei der Read-Replikation ist ein Fehler aufgetreten. Weitere Informationen finden Sie in der Ereignismeldung.
	RDS-EVENT-0046	Das Lesereplikat hat die Replikation fortgesetzt. Diese Meldung wird beim ersten Erstellen eines Lesereplikats sowie als Überwachungsmeldung zur Bestätigung der ordnungsgemäßen Replikationsausführung angezeigt. Falls diese Meldung auf die Benachrichtigung RDS-EVENT-0045 folgt, wurde die Replikation (nach einem Fehler oder nachdem sie gestoppt wurde) fortgesetzt.
	RDS-EVENT-0057	Die Replikation auf dem Lesereplikat wurde beendet.

Kategorie	Amazon RDS-Ereignis-ID	Beschreibung
	RDS-EVENT-0062	Die Replikation auf dem Leserepli kat wurde manuell gestoppt.
	RDS-EVENT-0063	Die Replikation auf dem Lesereplikat wurde zurückgesetzt.
Wiederherstellung	RDS-EVENT-0020	Wiederherstellung der DB-Instance wurde gestartet. Die Wiederherstellungs dauer variiert je nach zu wiederherstellende r Datenmenge.
	RDS-EVENT-0021	Wiederherstellung der DB-Instance ist abgeschlossen.
	RDS-EVENT-0023	Eine manuelle Sicherung wurde angefordert, jedoch erstellt Neptune gerade einen DB-Snapshot. Senden Sie die Anforderu ng erneut, nachdem Neptune den DB-Snapshot abgeschlo ssen hat.

Kategorie	Amazon RDS-Ereignis-ID	Beschreibung
	RDS-EVENT-0052	Die Wiederherstellung der Multi-AZ Instanz wurde gestartet. Die Wiederherstellungs dauer variiert je nach zu wiederherstellende r Datenmenge.
	RDS-EVENT-0053	Die Wiederherstellung der Multi-AZ Instanz ist abgeschlossen.
Wiederherstellung	RDS-EVENT-0008	Die DB-Instance wurde aus einem DB-Snapshot wiederher gestellt.
	RDS-EVENT-0019	Die DB-Instance wurde aus einem zeitpunktbezogenen Backup wiederher gestellt.

Neptune-Ereignisse aus einem DB-Cluster

Die folgende Tabelle zeigt eine Liste von Ereignissen nach Ereigniskategorie, wenn der Quelltyp ein DB-Cluster ist.

Kategorie	RDS-Ereignis-ID	Description
Failover	RDS-EVENT-0069	Ein Failover für das DB-Cluster ist fehlgeschlagen.

Kategorie	RDS-Ereignis-ID	Description
	RDS-EVENT-0070	Ein Failover für das DB-Cluster wurde neu gestartet.
	RDS-EVENT-0071	Ein Failover für das DB-Cluster wurde abgeschlossen.
	RDS-EVENT-0072	Ein Failover für das DB-Cluster wurde in derselben Availability Zone gestartet.
	RDS-EVENT-0073	Ein Failover für das DB-Cluster wurde in mehreren Availability Zones gestartet.
	RDS-EVENT-0083	Neptune konnte die Sicherungsdaten aus einem Amazon-S3-Bucket nicht kopieren. Wahrscheinlich sind die Berechtigungen für Neptune zum Zugriff auf den Amazon-S3-Bucket falsch konfiguriert.
Wartung	RDS-EVENT-0156	Der DB-Cluster verfügt über ein Upgrade der DB-Engine für kleinere Versionen.

Kategorie	RDS-Ereignis-ID	Description
Benachrichtigung	RDS-EVENT-0076	Migrieren zu einem Neptune-DB-Cluster fehlgeschlagen.
	RDS-EVENT-0077	Während der Migration zu einem Neptune-DB-Cluster ist der Versuch fehlgeschlagen, eine Tabelle aus der Quelldatenbank in ein Datenbankformular zu konvertieren.
	RDS-EVENT-0150	Der DB-Cluster wird angehalten.
	RDS-EVENT-0151	Der DB-Cluster wird gestartet.
	RDS-EVENT-0152	Das Anhalten des DB-Clusters ist fehlgesch lagen.
	RDS-EVENT-0153	Der DB-Cluster wird gestartet, da er die maximal zulässige Anhaltezeit überschri tten hat.

Neptune-Ereignisse aus einem DB-Cluster-Snapshot

Die folgende Tabelle zeigt die Ereigniskategorie und eine Liste von Ereignissen, wenn der Quelltyp ein Neptune-DB-Cluster-Snapshot ist.

Kategorie	RDS-Ereignis-ID	Beschreibung
backup	RDS-EVENT-0074	Die Erstellung eines manuellen DB-Cluster-Snapshots hat begonnen.
backup	RDS-EVENT-0075	Ein manueller DB-Cluster-Snapshot wurde erstellt.
Benachrichtigung	RDS-EVENT-0162	Fehler beim Exportieren von DB-Cluster-Snapshots.
Benachrichtigung	RDS-EVENT-0163	DB-Cluster-Snapshot-Exportaufgabe abgebrochen.
Benachrichtigung	RDS-EVENT-0164	DB-Cluster-Snapshot-Exportaufgabe abgeschlossen.
backup	RDS-EVENT-0168	Erstellen eines automatisierten Cluster-Snapshots.
backup	RDS-EVENT-0169	Automatisierter Cluster-Snapshot erstellt.
Erstellung	RDS-EVENT-0170	DB-Cluster erstellt.
Löschung	RDS-EVENT-0171	DB-Cluster gelöscht.
Benachrichtigung	RDS-EVENT-0172	DB-Cluster von [alter DB-Cluster-Name] in [neuer DB-Cluster-Name] umbenannt.

Neptune-Ereignisse aus einer DB-Cluster-Parametergruppe

Die folgende Tabelle zeigt die Ereigniskategorie und eine Liste von Ereignissen, wenn der Quelltyp eine DB-Cluster-Parametergruppe ist.

Kategorie	RDS-Ereignis-ID	Beschreibung
Konfigurationsänderung	RDS-EVENT-0037	Die Parametergruppe wurde geändert.

Neptune-Ereignisse aus einer Sicherheitsgruppe

Die folgende Tabelle zeigt die Ereigniskategorie und eine Liste von Ereignisse, wenn der Quelltyp eine Sicherheitsgruppe ist.

Kategorie	RDS-Ereignis-ID	Beschreibung
Konfigurationsänderung	RDS-EVENT-0038	Die Sicherheitsgruppe wurde geändert.
Ausfall	RDS-EVENT-0039	Die Sicherheitsgruppe, deren Besitzer [Benutzer] ist, existiert nicht. Die Autorisierung für diese Sicherheitsgruppe wurde widerrufen.

Neptun-Ereignisse anzeigen

Sie können die folgenden Informationen für jedes Neptun-Ereignis einsehen:

- Ressourcenname
- Ressourcentyp
- Ereigniskategorie

- Zeitpunkt des Ereignisses
- Eine zusammenfassende Meldung, die das Ereignis beschreibt

Sie können in den folgenden Bereichen der AWS Management Console auf Ereignisse zugreifen:

- Die Seite „Ereignisse“ im Navigationsbereich der Neptune-Konsole, auf der Ereignisse der letzten 24 Stunden angezeigt werden.
- Die Tabelle mit den letzten Ereignissen auf der Registerkarte Protokolle und Ereignisse auf der Detailseite eines DB-Clusters oder einer DB-Instance, in der Ereignisse der letzten 14 Tage angezeigt werden können.

Sie können Ereignisse auch mithilfe des AWS CLI Befehls describe-events oder der DescribeEventsAPI-Operation abrufen. Mit der API AWS CLI oder können Sie Ereignisse der letzten 14 Tage abrufen.

Ereignisse anzeigen (Konsole)

So zeigen Sie Neptun-Ereignisse in der Konsole an

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die Neptune-Konsole unter. https://console.aws.amazon.com/neptune/home
2. Wählen Sie im Navigationsbereich die Option Events.

   Auf der Seite werden Ereignisse der letzten 24 Stunden angezeigt. Sie können Ereignisse nach Quelltyp und Ereigniskategorie filtern.

Ereignisse anzeigen (AWS CLI)

Um alle Neptun-Ereignisse anzuzeigen, die in der letzten Stunde generiert wurden, führen Sie den Befehl `describe-events` ohne Parameter aus:

```
aws neptune describe-events
```

Verwenden Sie den folgenden Parameter, um Ereignisse der letzten 7 Tage (10.080 Minuten) anzuzeigen: `--duration`

```
aws neptune describe-events --duration 10080
```

Um Ereignisse für einen bestimmten DB-Cluster innerhalb eines Zeitraums anzuzeigen, verwenden Sie die `--source-identifier` Parameter, `--source-type--start-time`, und`--end-time`:

```
aws neptune describe-events \
    --source-identifier my-cluster \
    --source-type db-cluster \
    --start-time 2026-04-01T00:00Z \
    --end-time 2026-04-01T23:59Z
```

Weitere Informationen finden Sie unter describe-events in der AWS CLI Befehlsreferenz. Informationen zum programmgesteuerten Abrufen von Ereignissen finden Sie DescribeEventsin der Neptune API-Referenz.

Abonnieren von Neptune-Ereignisbenachrichtigungen

Sie können die Neptune-Konsole verwenden, um Ereignisbenachrichtigungen wie folgt zu abonnieren:

So abonnieren Sie Neptune-Ereignisbenachrichtigungen

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die Amazon Neptune Neptune-Konsole unter. https://console.aws.amazon.com/neptune/home
2. Wählen Sie im Navigationsbereich Ereignisabonnements aus.
3. Wählen Sie im Bereich Ereignisabonnements Ereignisabonnement erstellen aus.
4. Gehen Sie im Dialogfeld Ereignisabonnement erstellen wie folgt vor:
    a. Geben Sie unter Name einen Namen für das Abonnement für Ereignisbenachrichtigungen ein.
    b. Wählen Sie unter Send notifications to (Benachrichtigungen senden an) einen vorhandenen Amazon SNS-ARN für ein Amazon SNS-Thema oder wählen Sie create topic (Thema erstellen), um den Namen für ein Thema und eine Liste von Empfängern anzugeben.
    c. Wählen Sie unter Quelltyp einen Quelltyp aus.
    d. Wählen Sie Ja aus, um das Abonnement zu aktivieren. Wenn Sie das Abonnement erstellen möchten, jedoch noch keine Benachrichtigungen gesendet haben, wählen Sie Nein aus.
    e. Wählen Sie abhängig vom ausgewähltem Quelltyp die Ereigniskategorien und Quellen aus, für die Sie Ereignisbenachrichtigungen erhalten möchten.
    f. Wählen Sie Erstellen aus.

Verwalten von Abonnements für Neptune-Ereignisbenachrichtigungen

Wenn Sie im Navigationsbereich der Neptune-Konsole Ereignisbenachrichtigungen auswählen, können Sie Abonnementkategorien und die Liste aktueller Abonnements anzeigen.

Sie können auch ein bestimmtes Abonnement ändern oder löschen.

Ändern von Abonnements für Neptune-Ereignisbenachrichtigungen

Ändern aktueller Abonnements für Neptune-Ereignisbenachrichtigungen

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die Amazon Neptune Neptune-Konsole unter. https://console.aws.amazon.com/neptune/home
2. Wählen Sie im Navigationsbereich Ereignisabonnements aus. Im Bereich Ereignisabonnements werden all Ihre Abonnements für Ereignisbenachrichtigungen angezeigt.
3. Wählen Sie im Bereich Ereignisabonnements das Abonnement, das Sie modifizieren möchten, und klicken Sie auf Bearbeiten.
4. Nehmen Sie Ihre Änderungen am Abonnement im Bereich Ziel oder Quelle vor. Sie können Quell-IDs hinzufügen oder entfernen, indem Sie diese im Bereich Quelle auswählen oder ihre Auswahl aufheben.
5. Wählen Sie Bearbeiten aus. Die Neptune-Konsole zeigt an, dass das Abonnement geändert wurde.

Löschen von Abonnements für Neptune-Ereignisbenachrichtigungen

Sie können ein Abonnement löschen, wenn Sie es nicht mehr benötigen. Alle Abonnenten des Themas erhalten dann keine weiteren Ereignisbenachrichtigungen, die über dieses Abonnement ausgegeben wurden.

Löschen eines Abonnements für Neptune-Ereignisbenachrichtigungen

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die Amazon Neptune Neptune-Konsole unter. https://console.aws.amazon.com/neptune/home
2. Wählen Sie im Navigationsbereich Ereignisabonnements aus.
3. Wählen Sie im Bereich Ereignisabonnements das Abonnement aus, das Sie löschen möchten.
4. Wählen Sie Löschen aus.

5. Die Neptune-Konsole zeigt an, dass das Abonnement gelöscht wurde.

Taggen von Amazon Neptune Neptune-Ressourcen

Sie können mit Neptune-Tags Metadaten zu Neptune-Ressourcen hinzufügen. Darüber hinaus können Sie Tags mit AWS Identity and Access Management (IAM-) Richtlinien verwenden, um den Zugriff auf Neptune-Ressourcen zu verwalten und zu kontrollieren, welche Aktionen auf diese Ressourcen angewendet werden können. Außerdem können Sie mit Tags auch Kosten verfolgen, indem Ausgaben für ähnlich getaggte Ressourcen gruppiert werden.

Alle Neptune-Verwaltungsressourcen können getaggt werden, einschließlich der folgenden:

- DB-Instances
- DB-Cluster
- Read Replicas
- DB-Snapshots
- DB-Cluster-Snapshots
- Ereignisabonnements
- DB-Parametergruppen
- DB-Cluster-Parametergruppen
- DB-Subnetzgruppen

Überblick über Neptun-Ressourcen-Tags

Ein Amazon-Neptune-Tag ist ein Name-Wert-Paar, das Sie definieren und mit einer Neptune-Ressource verknüpfen. Der Name wird als der Schlüssel bezeichnet. Die Angabe eines Wertes für den Schlüssel ist optional. Sie können mit Tags einer Neptune-Ressource Informationen zuweisen. Sie können einen Tag-Schlüssel z. B. dazu verwenden, um eine Kategorie zu definieren, und ein Tag-Wert könnte ein Element in dieser Kategorie sein. Sie könnten beispielsweise den Tag-Schlüssel „Projekt“ und den Tag-Wert „Salix“ definieren. Das bedeutet, dass die Neptune-Ressource dem Projekt „Salix“ zugewiesen ist. Sie können mit Tags Neptune-Ressourcen auch als Test- oder Produktionsressourcen bezeichnen, indem Sie einen Schlüssel wie `environment=test` oder `environment=production` verwenden. Wir empfehlen die Verwendung eines konsistenten Satzes von Tag-Schlüsseln, um die mit Neptune-Ressourcen verknüpften Metadaten einfacher verfolgen zu können.

Verwenden Sie Tags, um Ihre AWS Rechnung so zu organisieren, dass sie Ihrer eigenen Kostenstruktur entspricht. Melden Sie sich dazu an, um Ihre AWS-Konto Rechnung mit den Tag-Schlüsselwerten zu erhalten. Um dann die Kosten kombinierter Ressourcen anzuzeigen, organisieren Sie Ihre Fakturierungsinformationen nach Ressourcen mit gleichen Tag-Schlüsselwerten. Beispielsweise können Sie mehrere Ressourcen mit einem bestimmten Anwendungsnamen markieren und dann Ihre Fakturierungsinformationen so organisieren, dass Sie die Gesamtkosten dieser Anwendung über mehrere Services hinweg sehen können. Weitere Informationen finden Sie unter Verwendung von Kostenzuordnungs-Tags im AWS Billing -Benutzerhandbuch.

Jede Neptune-Ressource besitzt einen Tag-Satz, der alle Tags enthält, die dieser Neptune-Ressource zugewiesen sind. Ein Tag-Satz kann bis zu zehn Tags enthalten oder leer sein. Wenn Sie einer Neptune-Ressource ein Tag hinzufügen, das denselben Schlüssel wie ein bereits vorhandenes Tag der Ressource hat, überschreibt der neue Wert den alten Wert.

AWS weist Ihren Tags keine semantische Bedeutung zu; Tags werden ausschließlich als Zeichenketten interpretiert. Neptune kann Tags für eine DB-Instance oder andere Neptune-Ressourcen festlegen, abhängig von den Einstellungen, die Sie beim Erstellen der Ressource verwenden. Beispielsweise kann Neptune ein Tag hinzufügen, das eine DB-Instance als Produktions- oder Testressource kennzeichnet.

- Der Tag-Schlüssel ist der erforderliche Name des Tags. Der Zeichenfolgenwert kann aus 1 bis 128 Unicode-Zeichen bestehen. Ihm darf kein "`aws:`" oder "`rds:`" als Präfix vorangestellt werden. Die Zeichenfolge darf nur Unicode-Zeichen, Ziffern, Leerzeichen sowie '_', '.', '/', '=', '+', '-' enthalten (Java-Regex: "`^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$`").
- Der Tag-Wert ist ein optionaler Zeichenfolgenwert des Tags. Der Zeichenfolgenwert kann aus 1 bis 256 Unicode-Zeichen bestehen. Ihm darf kein "`aws:`" als Präfix vorangestellt werden. Die Zeichenfolge darf nur Unicode-Zeichen, Ziffern, Leerzeichen sowie '_', '.', '/', '=', '+', '-' enthalten (Java-Regex: "`^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$`").

  Die Werte innerhalb eines Tag-Satzes müssen nicht eindeutig und können null sein. Es ist z. B. ein Schlüssel-Wert-Paar in einem Tag-Satz `project/Trinity` und `cost-center/Trinity` möglich.

> **Note**
>
> Sie können einem Snapshot ein Tag hinzufügen. Diese Gruppierung erscheint jedoch nicht in Ihrer Rechnung.

Sie können die AWS-Managementkonsole, oder die Neptune-API verwenden AWS CLI, um Tags zu Neptune-Ressourcen hinzuzufügen, aufzulisten und zu löschen. Wenn Sie die AWS CLI oder die Neptune-API verwenden, müssen Sie den Amazon-Ressourcennamen (ARN) für die Neptune-Ressource angeben, mit der Sie arbeiten möchten. Weitere Informationen zum Konstruieren eines ARN finden Sie unter Konstruieren eines ARN für Neptune.

Tags werden für Autorisierungszwecke im Cache gespeichert. Daher kann es einige Minuten dauern, bis Ergänzungen und Aktualisierungen für Tags von Neptune-Ressourcen verfügbar sind.

Tags in Neptune kopieren

Wenn Sie eine DB-Instance erstellen oder wiederherstellen, können Sie festlegen, dass die Tags aus der DB-Instance in Snapshots der DB-Instance kopiert werden. Das Kopieren von Tags stellt sicher, dass die Metadaten für die DB-Snapshots mit denen der Quell-DB-Instance übereinstimmen und alle vordefinierten Zugriffsrichtlinien für die DB-Instance denen der Quell-DB-Instance entsprechen. Tags werden nicht standardmäßig kopiert.

Sie können für die folgenden Aktionen festlegen, dass Tags in DB-Snapshots kopiert werden:

- Erstellen einer DB-Instance
- Wiederherstellen einer DB-Instance
- Erstellen eines Read Replicas
- Kopieren eines DB-Snapshots

> **Note**
>
> Wenn Sie einen Wert für den `--tag-key` Parameter des AWS CLI Befehls create-db-cluster-snapshot angeben (oder mindestens ein Tag für die DBClusterSnapshot erstellen API-Aktion angeben), kopiert Neptune keine Tags von der Quell-DB-Instance in den neuen DB-Snapshot. Dies gilt auch dann, wenn in der Quell-DB-Instance die Option `--copy-tags-to-snapshot` (`CopyTagsToSnapshot`) aktiviert ist.
> Sie können also eine Kopie einer DB-Instance aus einen DB-Snapshot erstellen und vermeiden, Tags hinzuzufügen, die nicht für die neue DB-Instance gelten. Nachdem Sie Ihren DB-Snapshot mit dem AWS CLI `create-db-cluster-snapshot` Befehl (oder der `CreateDBClusterSnapshot` Neptune-API-Aktion) erstellt haben, können Sie Tags hinzufügen, wie später in diesem Thema beschrieben.

Taggen in Neptune mit dem AWS-Managementkonsole

Amazon-Neptune-Ressourcen werden alle auf ähnliche Weise markiert. Das folgende Verfahrezn zeigt das Markieren einer Neptune-DB-Instance.

So fügen Sie ein Tag zu einer DB-Instance hinzu

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die Amazon Neptune Neptune-Konsole unter. https://console.aws.amazon.com/neptune/home
2. Wählen Sie im Navigationsbereich Instances aus.

   > **Note**
   >
   > Geben Sie im Bereich Instances im Feld Filter instances (Instances filtern) eine Textzeichenfolge ein, um die Liste der DB-Instances zu filtern. Es werden nur DB-Instances angezeigt, welche die Zeichenfolge enthalten.

3. Wählen Sie die DB-Instance aus, die getaggt werden soll.
4. Wählen Sie Instance actions und dann See details aus.
5. Scrollen Sie im Detailbereich nach unten zum Bereich Tags.
6. Wählen Sie Add aus. Das Fenster Add tags (Tags hinzufügen) wird angezeigt.
7. Geben Sie einen Wert für Tag key (Tag-Schlüssel) und Value (Wert) ein.
8. Wenn Sie ein weiteres Tag hinzufügen möchten, klicken Sie auf Add another Tag (Weiteres Tag hinzufügen) und geben Sie einen Wert für Tag key (Tag-Schlüssel) und Value (Wert) ein.

   Wiederholen Sie diesen Schritt, bis Sie alle Tags hinzugefügt haben.
9. Wählen Sie Add aus.

So löschen Sie ein Tag aus einer DB-Instance

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die Amazon Neptune Neptune-Konsole unter. https://console.aws.amazon.com/neptune/home
2. Wählen Sie im Navigationsbereich Instances aus.

> **Note**
>
> Geben Sie im Bereich Instances im Feld Filter instances (Instances filtern) eine Textzeichenfolge ein, um die Liste der DB-Instances zu filtern. Es werden nur DB-Instances angezeigt, welche die Zeichenfolge enthalten.

3. Wählen Sie die DB-Instance aus, die getaggt werden soll.
4. Wählen Sie Instance actions und dann See details aus.
5. Scrollen Sie im Detailbereich nach unten zum Bereich Tags.
6. Wählen Sie das Tag aus, das Sie löschen möchten.
7. Klicken Sie auf Remove (Entfernen) und dann im Fenster Remove tags (Tags entfernen) auf Remove (Entfernen).

Taggen in Neptune mit dem AWS CLI

Über die AWS CLI können Sie Tags für eine DB-Instance in Neptune hinzufügen, auflisten oder entfernen.

- Verwenden Sie den Befehl, um einer Neptun-Ressource ein oder mehrere Tags hinzuzufügen. AWS CLI `add-tags-to-resource`
- Verwenden Sie den Befehl, um die Tags einer Neptun-Ressource aufzulisten. AWS CLI `list-tags-for-resource`
- Verwenden Sie den Befehl, um ein oder mehrere Tags aus einer Neptun-Ressource zu entfernen. AWS CLI `remove-tags-from-resource`

Weitere Informationen zum Erstellen des erforderlichen Amazon-Ressourcennamens (ARN) finden Sie unter Konstruieren eines ARN für Neptune.

Taggen in Neptune mithilfe der API

Über die Neptune-API können Sie Tags für eine DB-Instance hinzufügen, auflisten oder entfernen.

- Um einen Tag zu einer Neptune-Ressource hinzuzufügen, verwenden Sie die Operation `AddTagsToResource`.

- Um die Tags für eine Neptune-Ressource aufzulisten, verwenden Sie die Operation `ListTagsForResource`.
- Um Tags aus einer Neptune-Ressource zu entfernen, verwenden Sie die Operation `RemoveTagsFromResource`.

Weitere Informationen zum Konstruieren des erforderlichen ARN finden Sie unter Konstruieren eines ARN für Neptune.

Beim Arbeiten mit XML über die Neptune-API verwenden Tags das folgende Schema:

```
<Tagging>
    <TagSet>
        <Tag>
            <Key>Project</Key>
            <Value>Trinity</Value>
        </Tag>
        <Tag>
            <Key>User</Key>
            <Value>Jones</Value>
        </Tag>
    </TagSet>
</Tagging>
```

Die folgende Tabelle enthält eine Liste der zulässigen XML-Tags und deren Eigenschaften. Bei den Werten für `Key` und `Value` muss zwischen Klein- und Großbuchstaben unterschieden werden. Beispiel: `project=Trinity` und `PROJECT=Trinity` sind zwei verschiedene Tags.

Markieren von Elementen	Description
`TagSet`	Ein Tag-Satz ist ein Container für alle Tags, die einer Neptune-R essource zugewiesen sind. Es ist nur ein Tag-Satz pro Ressource zulässig. Das Arbeiten mit einem `TagSet` ist nur über die Neptune-API möglich.
`Tag`	Ein Tag ist ein benutzerdefiniertes Schlüssel-Wert-Paar. Ein Tag-Satz kann 1 bis 50 Tags enthalten.

Markieren von Elementen	Description
Key	Ein Schlüssel ist der erforderliche Name des Tags. Der Zeichenfolgenwert kann aus 1 bis 128 Unicode-Zeichen bestehen. Ihm darf kein "rds:" oder "aws:" als Präfix vorangestellt werden. Die Zeichenfolge darf nur Unicode-Zeichen, Ziffern, Leerzeichen sowie '_', '.', '/', '=', '+', '-' enthalten (Java-Regex: "^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$"). Schlüssel müssen in einem Tag-Satz eindeutig sein. Sie können z. B. in einem Tag-Satz kein Schlüsselpaar mit gleichem Schlüssel, aber unterschiedlichen Werten verwenden, wie project/Trinity und project/Xanadu .
Wert	Ein Wert ist der optionale Wert des Tags. Der Zeichenfolgenwert kann aus 1 bis 256 Unicode-Zeichen bestehen. Ihm darf kein "rds:" oder "aws:" als Präfix vorangestellt werden. Die Zeichenfolge darf nur Unicode-Zeichen, Ziffern, Leerzeichen sowie '_', '.', '/', '=', '+', '-' enthalten (Java-Regex: "^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$ "). Die Werte innerhalb eines Tag-Satzes müssen nicht eindeutig und können null sein. Es ist z. B. ein Schlüssel-Wert-Paar in einem Tag-Satz project/Trinity und cost-center/Trinity möglich.

Arbeiten mit administrativen ARNs in Amazon Neptune

In Amazon Web Services erstellte Ressourcen werden anhand eines Amazon-Ressourcennamens (ARN) eindeutig identifiziert. Für bestimmte Amazon-Neptune-Operationen müssen Sie Neptune-Ressourcen eindeutig identifizieren, indem Sie ihren ARN angeben.

> **Important**
>
> Amazon Neptune verwendet das Format von Amazon-RDS-ARNs für administrative Aktionen über die Management-API-Referenz. Administrative Neptune-ARNs enthalten rds und nicht neptune-db. Informationen zu ARNs auf Datenebene, die Neptune-Datenressourcen kennzeichnen, finden Sie unter Angeben von Datenressourcen.

Themen

- Konstruieren eines ARN für Neptune
- Einen vorhandenen ARN in Amazon Neptune abrufen

Konstruieren eines ARN für Neptune

Sie können mithilfe der folgenden Syntax einen ARN für eine Amazon-Neptune-Ressource konstruieren. Beachten Sie, dass Neptune das Format von Amazon-RDS-ARNs verwendet.

`arn:aws:rds:<region>:<account number>:<resourcetype>:<name>`

Die folgende Tabelle zeigt das Format, das Sie beim Konstruieren eines ARN für einen bestimmten administrativen Neptune-Ressourcentyp verwenden sollten.

Ressourcentyp	ARN-Format
DB-Instance	arn:aws:rds:: *<region>* *<account>* :db: *<name>* Beispiel: `arn:aws:rds: us-east-2 :123456789012 :db:my-instance-1`
DB-Cluster	arn:aws:rds:*<region>*: *<account>* :cluster: *<name>* Beispiel: `arn:aws:rds: us-east-2 :123456789012 :cluster: my-cluster-1`
Ereignisabonnement	arn:aws:rds:*<region>*: *<account>* :es: *<name>* Beispiel: `arn:aws:rds: us-east-2 :123456789012 :es:my-subscription`
DB-Parametergruppe	arn:aws:rds:*<region>*: *<account>* :pg: *<name>* Beispiel:

Ressourcentyp	ARN-Format
	`arn:aws:rds: us-east-2 :123456789012 :pg:my-param-enable-logs`
DB-Cluster-Parametergruppe	arn:aws:rds:*<region>*: *<account>* :cluster-pg: *<name>* Beispiel: `arn:aws:rds: us-east-2 :123456789012 :cluster-pg: my-cluster-param-timezone`
DB-Cluster-Snapshot	arn:aws: rds:*<region>*: *<account>* :cluster-snapshot: *<name>* Beispiel: `arn:aws:rds: us-east-2 :123456789012 :cluster-snapshot: my-snap-20160809`
DB-Subnetzgruppe	arn:aws: rds: *<region>* *<account>* :subgrp: *<name>* Beispiel: `arn:aws:rds: us-east-2 :123456789012 :subgrp:my-subnet-10`

Einen vorhandenen ARN in Amazon Neptune abrufen

Sie können den ARN einer Neptun-Ressource mithilfe der AWS-Managementkonsole, AWS Command Line Interface (AWS CLI) oder Neptune-API abrufen.

Abrufen eines vorhandenen ARN mit dem AWS-Managementkonsole

Um einen ARN über die Konsole abzurufen, navigieren Sie zu der Ressource, für die Sie einen ARN erhalten möchten, und zeigen Sie die Details für diese Ressource an. Zum Abrufen des ARN für eine DB-Instance, wählen Sie z. B. Instances im Navigationsbereich und dann die gewünschte Instance aus der Liste aus. Der ARN befindet sich im Bereich Instance Details (Instance-Details).

Abrufen eines vorhandenen ARN mit dem AWS CLI

Um den zu verwenden AWS CLI , um einen ARN für eine bestimmte Neptun-Ressource zu erhalten, verwenden Sie den `describe` Befehl für diese Ressource. Die folgende Tabelle zeigt jeden AWS CLI Befehl und die ARN-Eigenschaft, die zusammen mit dem Befehl verwendet wird, um einen ARN abzurufen.

AWS CLI Befehl	ARN-Eigenschaft
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

Mit dem folgenden AWS CLI Befehl wird beispielsweise der ARN für eine DB-Instance abgerufen.

Example

Für Linux, OS X oder Unix:

```
aws neptune describe-db-instances \
--db-instance-identifier DBInstanceIdentifier \
--region us-west-2
```

Für Windows:

```
aws neptune describe-db-instances ^
```

```
--db-instance-identifier DBInstanceIdentifier ^
--region us-west-2
```

Einen vorhandenen ARN mithilfe der API abrufen

Um einen ARN für eine bestimmte Neptune-Ressource abzurufen, rufen Sie die folgenden API-Aktionen auf und verwenden die angezeigten ARN-Eigenschaften.

Neptune API-Aktion	ARN-Eigenschaft
DescribeEventSubscriptions	EventSubscriptionArn
DescribeCertificates	CertificateArn
DescribeDBParameterGroups	DBParameterGroupArn
DescribeDBClusterParameterGroups	DBClusterParameterGroupArn
DescribeDBInstances	DBInstanceArn
DescribeEvents	SourceArn
DescribeDBSubnetGroups	DBSubnetGroupArn
DescribeDBClusters	DBClusterArn
DescribeDBClusterSnapshots	DBClusterSnapshotArn

Sichern und Wiederherstellen eines Amazon-Neptune-DB-Clusters

Dieser Abschnitt zeigt, wie Sie Amazon-Neptune-DB-Cluster sichern und wiederherstellen.

Themen

- Neptune-Backups
- Fehlertoleranz für einen Neptune-DB-Cluster
- CloudWatch Metriken, die für die Verwaltung des Neptune-Backup-Speichers nützlich sind
- Wiederherstellen von Daten aus einer Neptune-Sicherung
- Sicherungszeitfenster in Neptune
- Erstellen eines DB-Cluster-Snapshots in Neptune
- Wiederherstellen aus einem DB-Cluster-Snapshot
- Kopieren eines DB-Cluster-Snapshots
- Freigeben eines DB-Cluster-Snapshots
- Löschen eines Neptune-Snapshots

Neptune-Backups

Neptune sichert Ihr Cluster-Volume automatisch und bewahrt die Wiederherstellungsdaten für die Länge des Aufbewahrungszeitraums für Sicherungen auf. Neptune-Sicherungen sind kontinuierlich und inkrementell, so dass Sie schnell eine Sicherung zu einem beliebigen Punkt im Aufbewahrungszeitraum für Sicherungen durchführen können. Es gibt keine Auswirkungen auf die Leistungsfähigkeit oder Unterbrechung von Datenbank-Services, während Backup-Daten geschrieben werden. Sie können einen Aufbewahrungszeitraum für Backups von 1 bis 35 Tagen festlegen, wenn Sie Ihr DB-Cluster erstellen oder ändern.

Wenn Sie Ihren Sicherungsspeicher kontrollieren möchten, können Sie das Aufbewahrungsintervall für Sicherungen verringern, alte (nicht mehr benötigte) manuelle Snapshots entfernen oder beides. Wenn Sie Ihre Kosten kontrollieren möchten, können Sie die Menge des Speichers überwachen, der nach Ablauf des Aufbewahrungszeitraums noch von kontinuierlichen Sicherungen und manuellen Snapshots belegt wird. Sie können den Aufbewahrungszeitraum für Sicherungen verringern und manuelle Snapshots entfernen, wenn sie nicht mehr benötigt werden.

Wenn Sie ein Backup nach dem eingestellten Aufbewahrungszeitraum behalten möchten, können Sie auch einen Snapshot der Daten Ihres Cluster-Volumes machen. Für das Speichern von Snapshots fallen die Standardgebühren für die Speicherplatznutzung in Neptune an. Weitere Informationen zu Neptune-Preisen finden Sie unter Amazon Neptune – Preise.

Neptune behält für den gesamten Aufbewahrungszeitraum für Backups inkrementelle Wiederherstellungsdaten bei. Daher müssen Sie nur einen Snapshot für Daten erstellen, die Sie über den Aufbewahrungszeitraum hinaus beibehalten möchten. Sie können ein neues DB-Cluster aus dem Snapshot erstellen.

Important

Wenn Sie einen DB-Cluster löschen, werden alle seine automatisierten Sicherungen gleichzeitig gelöscht und können nicht wiederhergestellt werden. Dies bedeutet, dass Sie die DB-Instance nicht zu einem späteren Zeitpunkt in den abschließenden Zustand wiederherstellen können, es sei denn, Sie erstellen manuell einen abschließenden DB-Snapshot. Manuelle Snapshots werden nicht gelöscht, wenn der Cluster gelöscht wird.

Note

- Für Amazon-Neptune-DB-Cluster beträgt der Standard-Aufbewahrungszeitraum für Backups einen Tag, unabhängig davon, wie der DB-Cluster erstellt wurde.
- Sie können automatisierte Backups auf Neptune nicht deaktivieren. Die Aufbewahrungszeitraum von Backups für Neptune wird vom DB-Cluster verwaltet.

Fehlertoleranz für einen Neptune-DB-Cluster

Ein Neptune-DB-Cluster ist darauf ausgelegt, fehlertolerant zu sein. Das Cluster-Volume erstreckt sich über mehrere Availability Zones in einer einzigen AWS Region, und jede Availability Zone enthält eine Kopie der Cluster-Volume-Daten. Diese Funktionalität bedeutet, dass Ihr DB-Cluster einen Fehler in einer Availability Zone ohne Datenverlust und mit einer nur sehr kurzen Unterbrechung des Services toleriert.

Wenn die primäre Instance in einem DB-Cluster ausfällt, führt Neptune automatisch ein Failover zu einer neuen primären Instance auf eine von zwei Arten durch:

- Durch das Hochstufen eines vorhandenen Neptune-Replikats zu einer neuen primären Instance
- Erstellen einer neuen primären Instance

Wenn ein DB-Cluster über ein oder mehrere Neptune-Replikate verfügt, wird während eines Failover-Ereignisses ein Neptune-Replikat zu einer primären Instance hochgestuft. Ein Fehlerereignis hat eine kurze Unterbrechung zufolge, während die Lese- und Schreibvorgänge mit einer Ausnahme fehlschlagen. Jedoch wird der Service im Normalfall in weniger als 120 Sekunden und oft sogar schon nach 60 Sekunden wiederhergestellt. Wir empfehlen Ihnen, mindestens ein oder mehrere Neptune-Replikate in zwei oder mehreren verschiedenen Availability Zones zu erstellen, um die Verfügbarkeit Ihres DB-Clusters zu erhöhen.

Sie können die Reihenfolge, in der Ihre Neptune-Replikate nach einem Fehler zur primären Instance hochgestuft werden, anpassen, indem Sie jedem Replikat eine Priorität zuweisen. Prioritäten liegen im Bereich zwischen 0 als höchste Priorität und 15 als niedrigste Priorität. Wenn die primäre Instance ausfällt, stuft Neptune das Neptune-Replikat mit der höchsten Priorität zur neuen primären Instance hoch. Sie können die Priorität eines Neptune-Replikats jederzeit anpassen. Das Ändern der Priorität löst kein Failover aus.

Sie können den verwenden AWS CLI , um die Failover-Priorität einer DB-Instance wie folgt festzulegen:

```
aws neptune modify-db-instance --db-instance-identifier (the instance ID) --promotion-
tier (the failover priority value)
```

Mehr als ein Neptune-Replikat kann die gleiche Priorität haben. Dies macht Hochstufungslevels möglich. Wenn zwei oder mehrere Neptune-Replikate die gleiche Priorität haben, stuft Neptune das größte Replikat hoch. Wenn zwei oder mehrere Neptune-Replikate dieselbe Priorität und Größe haben, stuft Neptune ein beliebiges Replikat auf derselben Hochstufungsebene hoch.

Wenn der DB-Cluster über keine Neptune-Replikate verfügt, wird die primäre Instance während des Ausfallereignisses neu erstellt. Ein Fehlerereignis hat eine Unterbrechung zufolge, während die Lese- und Schreibvorgänge mit einer Ausnahme fehlschlagen. Der Service wird wiederhergestellt, wenn die primäre Instance erstellt wird. Dies dauert im Normalfall weniger als 10 Minuten. Das Hochstufen eines Neptune-Replikats zur primären Instance ist viel schneller als die Erstellung einer neuen primären Instance.

CloudWatch Metriken, die für die Verwaltung des Neptune-Backup-Speichers nützlich sind

Sie können die CloudWatch Amazon-MetrikenTotalBackupStorageBilled,SnapshotStorageUsed, wie folgt verwenden, BackupRetentionPeriodStorageUsed um den Speicherverbrauch Ihrer Neptune-Backups zu überprüfen und zu überwachen:

- BackupRetentionPeriodStorageUsed stellt die Menge des Sicherungsspeichers in Bytes dar, die zurzeit zum Speichern kontinuierlicher Sicherungen verwendet wird. Dieser Wert hängt von der Größe des Cluster-Volumens und der Menge der von Ihnen vorgenommenen Änderungen während des Aufbewahrungszeitraums ab. Zu Fakturierungszwecken übersteigt dieser Wert die kumulative Cluster-Volume-Größe während des Aufbewahrungszeitraums nicht. Wenn beispielsweise die VolumeBytesUsed-Größe des Clusters 107.374.182.400 Bytes (100 GiB) und der Aufbewahrungszeitraum zwei Tage betragen, ist der maximale Wert für BackupRetentionPeriodStorageUsed 214.748.364.800 Bytes (100 GiB + 100 GiB).
- SnapshotStorageUsed stellt die Menge des Sicherungsspeichers in Bytes dar, die zum Speichern manueller Snapshots über den Aufbewahrungszeitraum für Sicherungen hinaus verwendet wird. Manuelle Snapshots werden nicht auf den Snapshot-Sicherungsspeicher angerechnet, während ihr Erstellungszeitstempel innerhalb des Aufbewahrungszeitraums liegt. Automatische Snapshots werden ebenfalls nicht auf den Snapshot-Sicherungsspeicher angerechnet. Die Größe einzelner Snapshots entspricht der Größe des Cluster-Volumes zum Zeitpunkt der Snapshot-Erstellung. Der SnapshotStorageUsed-Wert hängt von der Anzahl der Snapshots ab, die Sie aufbewahren, und von der Größe der einzelnen Snapshots. Angenommen, ein einzelner Snapshot liegt außerhalb des Aufbewahrungszeitraums und die VolumeBytesUsed-Größe des Clusters betrug 100 GiB, als der Snapshot erstellt wurde. Die Menge von SnapshotStorageUsed beträgt 107.374.182.400 Byte (100 GiB).
- TotalBackupStorageBilled stellt die Summe in Bytes aus BackupRetentionPeriodStorageUsed und SnapshotStorageUsed abzüglich eines Betrags für den freien Sicherungsspeicher dar, der der Größe des Cluster-Volumes für einen einzelnen Tag entspricht. Der freie Sicherungsspeicher entspricht der aktuellen Volume-Größe. Wenn beispielsweise die VolumeBytesUsed-Größe des Clusters 100 GiB beträgt, der Aufbewahrungszeitraum zwei Tage ist und es einen einzelnen manuellen Snapshot außerhalb des Aufbewahrungszeitraums gibt, beträgt TotalBackupStorageBilled 214.748.364.800 Bytes (200 GiB + 100 GiB - 100 GiB).

Sie können einen Neptune-Cluster überwachen und Berichte mithilfe von CloudWatch Metriken über die CloudWatch Konsole erstellen. Weitere Informationen zur Verwendung von CloudWatch Metriken finden Sie unter Überwachen von Neptune und in der Metriktabelle unter. Neptun-Metriken CloudWatch

Wiederherstellen von Daten aus einer Neptune-Sicherung

Sie können Ihre Daten wiederherstellen, indem Sie einen neuen DB-Cluster aus den von Neptune aufbewahrten Sicherungsdaten oder aus einem von Ihnen gespeicherten DB-Cluster-Snapshot erstellen. Sie können schnell eine neue Kopie eines DB-Clusters wiederherstellen, die aus den Sicherungsdaten von irgendeinem Zeitpunkt der Aufbewahrungsfrist für Backups erstellt wurde. Das Prinzip der fortlaufenden und ansteigenden Neptune-Backups während des Aufbewahrungszeitraums für Backups bedeutet, dass Sie keine häufigen Snapshots Ihrer Daten machen müssen, um die Wiederherstellungszeiten zu optimieren.

Um den spätesten oder frühesten wiederherstellbaren Zeitpunkt für eine DB-Instance zu ermitteln, suchen Sie in der Neptune-Konsole die Werte für `Latest Restorable Time` oder `Earliest Restorable Time`. Der späteste wiederherstellbare Zeitpunkt für ein DB-Cluster ist der aktuelle Punkt, an dem Sie Ihr DB-Cluster wiederherstellen können. Dieser liegt im Normalfall 5 Minuten vor dem aktuellen Zeitpunkt. Der frühste wiederherstellbare Zeitpunkt gibt an, wie weit Sie zeitlich innerhalb des Aufbewahrungszeitraums für Backups zurückgehen können, um Ihr Cluster-Volume an einem vergangen Zeitpunkt wiederherstellen zu können.

Sie können sehen, wann ein Wiederherstellungsvorgang eines DB-Clusters abgeschlossen ist, indem Sie `Latest Restorable Time` und `Earliest Restorable Time`-Werte überprüfen. Die Werte `Latest Restorable Time` und `Earliest Restorable Time` geben NULL zurück, bis der Wiederherstellungsvorgang abgeschlossen ist. Sie können kein Backup oder einen Wiederherstellungsvorgang anfordern, wenn `Latest Restorable Time` oder `Earliest Restorable Time` NULL zurückgibt.

Um eine DB-Instance auf einen bestimmten Zeitpunkt zurückzusetzen, verwenden Sie den AWS-Managementkonsole

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die Amazon Neptune Neptune-Konsole zu Hausehttps://console.aws.amazon.com/neptune/.
2. Wählen Sie im Navigationsbereich Instances aus. Wählen Sie die primäre Instance für das DB-Cluster aus, das Sie wiederherstellen möchten.

3. Wählen Sie Instance-Aktionen und anschließend Zu einem bestimmten Zeitpunkt wiederherstellen.

   Wählen Sie im Fenster DB-Instance starten unter Wiederherstellungszeit die Option Benutzerdefiniert aus.
4. Geben Sie in Custom (Benutzerdefiniert) das Datum und die Uhrzeit ein, zu dem/der Sie die Wiederherstellung ausführen möchten.
5. Geben Sie in DB instance identifier (DB-Instance-ID) unter Settings (Einstellungen) einen Namen für die neue, wiederhergestellte DB-Instance ein.
6. Wählen Sie Launch DB Instance (DB-Instance starten) aus, um die wiederhergestellte DB-Instance zu starten.

   Mit dem von Ihnen angegebenen Namen wird eine neue DB-Instance erstellt, und es wird ein neuer DB-Cluster erstellt. Der Name des DB-Clusters ist der Name der neuen DB-Instance, gefolgt von `-cluster`. Ist der neue DB-Instance-Name beispielsweise `myrestoreddb`, ist der Name des neuen DB-Clusters `myrestoreddb-cluster`.

Sicherungszeitfenster in Neptune

Automatisierte Backups werden täglich während des bevorzugten Zeitfensters für das Backup durchgeführt. Wenn das Backup mehr Zeit als im Zeitfenster angegeben benötigt, wird es nach dem Ende des Fensters weiter ausgeführt, bis es abgeschlossen ist. Das Zeitfenster für das Backup darf sich nicht mit dem wöchentlichen Wartungsfenster für die DB-Instance überschneiden.

Während des automatischen Backup-Fensters wird der Speicher I/O möglicherweise kurzzeitig unterbrochen, während der Backup-Vorgang initialisiert wird (in der Regel weniger als ein paar Sekunden). Bei Multi-AZ-Bereitstellungen können für einen Zeitraum von wenigen Minuten längere Latenzzeiten während eines Backups auftreten.

Das Sicherungszeitfenster wird normalerweise zufällig innerhalb eines achtstündigen Zeitblocks pro Region durch die Neptune zugrunde liegende Amazon-RDS-Steuerebene ausgewählt. Die Zeitblöcke für jede Region, innerhalb derer die Standardsicherungszeiträume zugewiesen werden, sind im Abschnitt Sicherungszeitfenster im Amazon-RDS-Benutzerhandbuch dokumentiert.

Erstellen eines DB-Cluster-Snapshots in Neptune

Neptune erstellt einen Snapshot für das Volume mit Ihrem DB-Cluster, damit der gesamte DB-Cluster gesichert wird und nicht nur einzelne Datenbanken. Wenn Sie einen DB-Cluster-Snapshot erstellen, müssen Sie bestimmen, welcher DB-Cluster gesichert werden soll. Geben Sie dann dem DB-Cluster-Snapshot einen Namen, sodass über diesen eine Wiederherstellung zu einem späteren Zeitpunkt möglich ist. Die Zeit, die für die Erstellung eines DB-Cluster-Snapshots benötigt wird, hängt von der Größe Ihrer Datenbanken ab. Der Snapshot umfasst das gesamte Speichervolumen. Die Größe von Dateien (z. B. temporäre Dateien) wirkt sich also auch auf die Zeitdauer aus, die für die Erstellung des Snapshots benötigt wird.

Sie können einen DB-Cluster-Snapshot mit der AWS-Managementkonsole AWS CLI, der oder der Neptune-API erstellen.

Verwenden der Konsole zum Erstellen eines DB-Cluster-Snapshots

So erstellen Sie einen DB-Cluster-Snapshot

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die Amazon Neptune Neptune-Konsole zu Hausehttps://console.aws.amazon.com/neptune/.
2. Wählen Sie im Navigationsbereich Datenbanken aus.
3. Wählen Sie in der Liste der DB-Instances die primäre Instance für den DB-Cluster aus.
4. Wählen Sie Instance actions (Instance-Aktionen) und anschließend Take snapshot (Snapshot erstellen) aus.

   Das Fenster Take DB Snapshot (DB-Snapshot erstellen) wird angezeigt.
5. Geben Sie den Namen des DB-Cluster-Snapshots in das Feld Snapshot name (Snapshot-Name) ein.
6. Wählen Sie Take Snapshot (Snapshot erstellen) aus.

Wiederherstellen aus einem DB-Cluster-Snapshot

Wenn Sie einen Amazon-Neptune-Snapshot für einen DB-Cluster erstellen, erstellt Neptune einen Snapshot für das Volume mit dem Cluster, damit alle seine Daten gesichert und nicht nur einzelne Instances werden. Sie können einen DB-Cluster erstellen, indem Sie eine Wiederherstellung aus diesem DB-Cluster-Snapshot durchführen. Wenn Sie den DB-Cluster wiederherstellen, geben Sie den Namen des DB-Cluster-Snapshots an, aus dem die Wiederherstellung gestartet werden soll, und anschließend einen Namen für den neuen DB-Cluster, der bei dieser Wiederherstellung erstellt wird.

Inhalt

- Dinge, die Sie beim Wiederherstellen eines Neptune-DB-Clusters aus einem Snapshot beachten sollten
  - Eine Wiederherstellung auf einen bestehenden DB-Cluster ist nicht möglich.
  - Es werden keine Instances wiederhergestellt
  - Es wird keine benutzerdefinierte Parametergruppe wiederhergestellt
  - Es werden keine benutzerdefinierten Sicherheitsgruppen wiederhergestellt
  - Sie können einen Cluster nicht aus einem Snapshot wiederherstellen, der sowohl freigegeben als auch verschlüsselt ist.
  - Ein wiederhergestellter DB-Cluster nutzt genauso viel Speicherplatz wie zuvor
- Sie führen Sie eine Wiederherstellung aus einem Snapshot durch
  - Verwenden der Konsole zum Wiederherstellen aus einem Snapshot

Dinge, die Sie beim Wiederherstellen eines Neptune-DB-Clusters aus einem Snapshot beachten sollten

Eine Wiederherstellung auf einen bestehenden DB-Cluster ist nicht möglich.

Bei der Wiederherstellung wird immer ein neuer DB-Cluster erstellt, so dass Sie die Wiederherstellung nicht in einem bereits vorhandenen DB-Cluster durchführen können.

Es werden keine Instances wiederhergestellt

Einem neuen DB-Cluster, der durch eine Wiederherstellung erstellt wurde, sind keine Instances zugeordnet.

Sobald die Wiederherstellung abgeschlossen ist und Ihr neuer DB-Cluster verfügbar ist, erstellen Sie explizit die Instances, die Sie benötigen. Hierzu können Sie die Neptune-Konsole oder die Erstellen DBInstance-API verwenden.

Es wird keine benutzerdefinierte Parametergruppe wiederhergestellt

Einem neuen DB-Cluster, der durch eine Wiederherstellung erstellt wurde, wird automatisch die Standard-DB-Parametergruppe zugeordnet.

Sobald die Wiederherstellung abgeschlossen und Ihr neuer DB-Cluster verfügbar ist, müssen Sie eine benutzerdefinierte DB-Parametergruppe zuordnen, die von der Instance verwendet wird, aus der Sie die Wiederherstellung gestartet haben. Verwenden Sie dazu den Befehl Ändern auf der Neptune-Konsole oder der Modifizieren DBInstance-API.

> **Important**
>
> Wir empfehlen, dass Sie eine benutzerdefinierte Parametergruppe speichern, die in einem DB-Cluster verwendet wird, von dem Sie einen Snapshot erstellen. Wenn Sie dann aus diesem Snapshot wiederherstellen, können Sie dem wiederhergestellten DB-Cluster problemlos die richtige Parametergruppe zuordnen.

Es werden keine benutzerdefinierten Sicherheitsgruppen wiederhergestellt

Einem neuen DB-Cluster, der durch eine Wiederherstellung erstellt wurde, wird automatisch die Standard-Sicherheitsgruppe zugeordnet.

Sobald die Wiederherstellung abgeschlossen und Ihr neuer DB-Cluster verfügbar ist, müssen Sie benutzerdefinierte Sicherheitsgruppen zuordnen, die von der Instance verwendet werden, aus der Sie die Wiederherstellung gestartet haben. Verwenden Sie dazu den Befehl Ändern auf der Neptune-Konsole oder der Modifizieren DBInstance-API.

Sie können einen Cluster nicht aus einem Snapshot wiederherstellen, der sowohl freigegeben als auch verschlüsselt ist.

Sie können einen DB-Cluster jedoch nicht aus einem DB-Cluster-Snapshot wiederherstellen, der zugleich freigegeben und verschlüsselt ist.

Stattdessen können Sie eine nicht freigegebene Kopie des Snapshots erstellen die Wiederherstellung von der Kopie aus durchführen.

Weitere Hinweise zum Umgang mit Verschlüsselung beim Kopieren eines Snapshots finden Sie unterVerschlüsselung beim Kopieren von Snapshots. Weitere Informationen zum Teilen verschlüsselter Snapshots finden Sie unterFreigeben eines verschlüsselten DB-Cluster-Snapshots.

Ein wiederhergestellter DB-Cluster nutzt genauso viel Speicherplatz wie zuvor

Wenn Sie einen DB-Cluster aus einem DB-Cluster-Snapshot wiederherstellen, entspricht die dem neuen Cluster zugewiesene Speichermenge der Speichermenge, die dem DB-Cluster zugewiesen wurde, aus dem der Snapshot erstellt wurde. Dabei spielt es keine Rolle, wie viel von dieser zugewiesenen Speichermenge tatsächlich verwendet wird.

Das heißt, dass sich die Ihnen in Rechnung gestellte „Hochwasssermarkierung“ nicht ändert. Zum Zurücksetzen der Hochwassermarkierung müssen die Daten aus Ihrem Graphen exportiert und anschließend in einen neuen DB-Cluster geladen werden (siehe Neptune-Speicher-Fakturierung).

Sie führen Sie eine Wiederherstellung aus einem Snapshot durch

Sie können einen DB-Cluster aus einem DB-Cluster-Snapshot mithilfe der AWS-Managementkonsole AWS CLI, der oder der Neptune-API wiederherstellen.

Verwenden der Konsole zum Wiederherstellen aus einem Snapshot

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die Amazon Neptune Neptune-Konsole unter. https://console.aws.amazon.com/neptune/home
2. Wählen Sie im Navigationsbereich die Option Snapshots.
3. Wählen Sie den DB-Cluster-Snapshot für die Wiederherstellung aus.
4. Wählen Sie Actions (Aktionen), Restore Snapshot (Snapshot wiederherstellen).
5. Geben Sie auf der Seite Restore DB Instance (DB-Instance wiederherstellen) im Feld DB Instance Identifier (DB-Instance-Kennung) den Namen für den wiederhergestellten DB-Cluster ein.
6. Klicken Sie auf Restore DB Instance (DB-Instance wiederherstellen).
7. Wenn Sie die Funktionalität des DB-Clusters auf die des aus dem Snapshot erstellten DB-Clusters wiederherstellen möchten, müssen Sie den DB-Cluster ändern, um die Sicherheitsgruppe zu verwenden. Die nächsten Schritte setzen voraus, dass sich Ihre DB-Instance in einer Virtual Private Cloud (VPC) befindet. Wenn sich Ihr DB-Cluster nicht in einer VPC befindet, verwenden Sie die Amazon-EC2-Konsole, um die für den DB-Cluster benötigte Sicherheitsgruppe zu finden.

a. Öffnen Sie die Amazon-VPC-Konsole unter https://console.aws.amazon.com/vpc/.

b. Wählen Sie im Navigationsbereich Security Groups (Sicherheitsgruppen) aus.

c. Wählen Sie die Sicherheitsgruppe aus, die Sie für Ihre DB-Cluster verwenden möchten. Falls nötig fügen Sie Regeln hinzu, um die Sicherheitsgruppe mit einer Sicherheitsgruppe für eine EC2-Instance zu verlinken.

Kopieren eines DB-Cluster-Snapshots

Mit Neptune können Sie automatisierte oder manuelle DB-Cluster-Snapshots kopieren. Nach dem Kopieren eines Snapshots ist die Kopie ein manueller Snapshot.

Sie können einen Snapshot innerhalb derselben AWS Region und regionsübergreifend AWS kopieren. Snapshots können in einem einzigen Schritt zwischen Regionen und Konten kopiert werden.

Das Kopieren eines automatisierten Snapshots in ein anderes AWS Konto erfolgt in zwei Schritten: Zuerst erstellen Sie einen manuellen Snapshot aus dem automatisierten Snapshot und dann kopieren Sie den manuellen Snapshot in das andere Konto.

Als Alternative zum Kopieren können Sie manuelle Snapshots auch mit anderen AWS Konten teilen. Weitere Informationen finden Sie unter Freigeben eines DB-Cluster-Snapshots.

Themen

- Einschränkungen beim Kopieren eines Snapshots
- Aufbewahrung von Kopien von DB-Cluster-Snapshots
- Verschlüsselung beim Kopieren von Snapshots
- Schnappschüsse zwischen den anderen kopieren AWS Regionen
- Kopieren eines DB-Cluster-Snapshots mithilfe der Konsole
- Kopieren eines DB-Cluster-Snapshots mit dem AWS CLI

Einschränkungen beim Kopieren eines Snapshots

Im folgenden werden einige Einschränkungen beim Kopieren von Snapshots aufgeführt:

- Sie können einen Snapshot zwischen China (Peking) und China (Ningxia) kopieren, aber Sie können keinen Snapshot zwischen diesen Regionen China und anderen AWS Regionen kopieren.
- Sie können einen Snapshot zwischen AWS GovCloud (US-East) und AWS GovCloud (US-West) kopieren, aber Sie können keinen Snapshot zwischen diesen AWS GovCloud (US) Regionen und anderen AWS Regionen kopieren.
- Wenn Sie einen Quell-Snapshot löschen, bevor der Ziel-Snapshot verfügbar ist, kann das Kopieren des Snapshots fehlschlagen. Verifizieren Sie, dass der Ziel-Snapshot den Status `AVAILABLE` hat, bevor Sie einen Quell-Snapshot löschen.

- Pro Konto können bis zu fünf Snapshot-Kopieranforderungen an eine einzelne Region aktiv sein.
- In Abhängigkeit von den beteiligten Regionen und der Menge der zu kopierenden Daten kann es Stunden dauern, bis eine regionsübergreifende Snapshot-Kopie fertiggestellt wird.

  Wenn es eine große Anzahl von regionsübergreifenden Snapshot-Kopieranforderungen aus einer bestimmten AWS Quellregion gibt, kann Neptune neue regionsübergreifende Kopieranfragen aus dieser AWS Quellregion in eine Warteschlange stellen, bis einige laufende Kopien abgeschlossen sind. Zu Kopieranforderungen, die sich in der Warteschlange befinden, werden keine Fortschrittsinformationen angezeigt. Fortschrittsinformationen werden erst angezeigt, nachdem die Kopie gestartet wurde.

Aufbewahrung von Kopien von DB-Cluster-Snapshots

Neptune löscht automatisierte Snapshots wie folgt:

- Am Ende des Aufbewahrungszeitraums.
- Wenn Sie automatisierte Snapshots für einen DB-Cluster deaktivieren.
- Wenn Sie einen DB-Cluster löschen.

Soll ein automatischer Snapshot länger aufbewahrt werden, erstellen Sie durch Kopieren einen manuellen Snapshot, der aufbewahrt wird, bis Sie ihn löschen. Es können Neptune-Speicherkosten können für manuelle Snapshots anfallen, wenn sie Ihren Standardspeicherplatz überschreiten.

Weitere Information zu Sicherungsspeicherkosten finden Sie unter Neptune – Preise.

Verschlüsselung beim Kopieren von Snapshots

Sie können einen Snapshot kopieren, der mit einem Verschlüsselungsschlüssel verschlüsselt wurde. AWS KMS Wenn Sie einen verschlüsselten Snapshot kopieren, muss auch die Kopie des Snapshots verschlüsselt werden. Sie können die Kopie mit demselben AWS KMS Verschlüsselungsschlüssel wie den ursprünglichen Snapshot verschlüsseln, oder Sie können einen anderen AWS KMS Verschlüsselungsschlüssel angeben.

Sie können einen unverschlüsselten DB-Cluster-Snapshot nicht verschlüsseln, wenn Sie ihn kopieren.

Bei Amazon Neptune DB-Cluster-Snapshots können Sie den DB-Cluster-Snapshot auch unverschlüsselt lassen und stattdessen bei der Wiederherstellung einen AWS KMS

Verschlüsselungsschlüssel angeben. Der wiederhergestellte DB-Cluster wird dann unter Verwendung des angegebenen Schlüssels verschlüsselt.

Schnappschüsse zwischen den anderen kopieren AWS Regionen

Wenn Sie einen Snapshot in eine AWS Region kopieren, die sich von der AWS Region des Quell-Snapshots unterscheidet, ist die erste Kopie eine vollständige Snapshot-Kopie, auch wenn Sie einen inkrementellen Snapshot kopieren. Eine vollständige Snapshot-Kopie enthält alle Daten und Metadaten, die zur Wiederherstellung der DB-Instance erforderlich sind. Nach der ersten Snapshot-Kopie können Sie inkrementelle Snapshots derselben DB-Instance in dieselbe Zielregion innerhalb desselben Kontos kopieren. AWS

Ein inkrementeller Snapshot enthält nur die Daten, die sich nach dem letzten Snapshot derselben DB-Instance geändert haben. Das inkrementelle Kopieren von Snapshots ist schneller und verursacht geringere Speicherkosten als das vollständige Kopieren von Snapshots. Das inkrementelle Kopieren von Snapshots zwischen AWS Regionen wird sowohl für unverschlüsselte als auch für verschlüsselte Snapshots unterstützt.

> **Important**
>
> Bei freigegebenen Snapshots wird das Kopieren inkrementeller Snapshots nicht unterstützt. Bei freigegebenen Snapshots sind alle Kopien vollständige Snapshots, auch innerhalb derselben Region.

Abhängig von den beteiligten AWS Regionen und der Menge der zu kopierenden Daten kann es Stunden dauern, bis eine regionsübergreifende Snapshot-Kopie abgeschlossen ist.

Kopieren eines DB-Cluster-Snapshots mithilfe der Konsole

Wenn es sich bei der Quell-Datenbank-Engine um Neptune handelt, ist der resultierende Snapshot ein DB-Cluster-Snapshot. Für jedes AWS Konto können Sie bis zu fünf DB-Cluster-Snapshots gleichzeitig pro AWS Region kopieren. Sie können sowohl verschlüsselte als auch unverschlüsselte DB-Cluster-Snapshots kopieren.

Weitere Informationen über die Kosten von Datenübertragungen finden Sie unter Neptune – Preise.

Um einen bereits laufenden Kopiervorgang abzubrechen, können Sie den DB-Cluster-Ziel-Snapshot löschen, während der DB-Cluster-Snapshot noch den Status copying hat.

Das folgende Verfahren eignet sich zum Kopieren verschlüsselter und unverschlüsselter DB-Cluster-Snapshots:

So kopieren Sie einen DB-Cluster-Snapshot

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die Amazon Neptune Neptune-Konsole unter. https://console.aws.amazon.com/neptune/home
2. Wählen Sie im Navigationsbereich die Option Snapshots.
3. Aktivieren Sie das Kontrollkästchen des zu kopierenden DB-Cluster-Snapshots.
4. Wählen Sie die Option Actions (Aktionen) und anschließend Copy Snapshot (Snapshot kopieren) aus. Die Seite Make Copy of DB Snapshot (Kopie des DB-Snapshots erstellen) erscheint.
5. Geben Sie den Namen für die Kopie des DB-Cluster-Snapshots in das Feld New DB Snapshot Identifier (Neue DB-Snapshot-Kennung) ein.
6. Wählen Sie zum Kopieren von Tags und Werten aus dem Snapshot in die Kopie des Snapshots die Option Copy Tags (Tags kopieren).
7. Wählen Sie für Enable Encryption (Verschlüsselung aktivieren) eine der folgenden Optionen:
   - Wählen Sie Disable encryption (Verschlüsselung deaktivieren), wenn der DB-Cluster-Snapshot nicht verschlüsselt ist und die Kopie nicht verschlüsselt werden soll.
   - Wählen Sie Enable encryption (Verschlüsselung aktivieren), wenn der DB-Cluster-Snapshot nicht verschlüsselt ist, die Kopie aber verschlüsselt werden soll. Geben Sie in diesem Fall für Master Key die AWS KMS Schlüssel-ID an, die zur Verschlüsselung der DB-Cluster-Snapshot-Kopie verwendet werden soll.
   - Wählen Sie Enable encryption (Verschlüsselung aktivieren), wenn der DB-Cluster-Snapshot verschlüsselt ist. In diesem Fall müssen Sie die Kopie verschlüsseln, Yes (Ja) ist also bereits ausgewählt. Geben Sie für Master Key die AWS KMS Schlüssel-ID an, die zur Verschlüsselung der DB-Cluster-Snapshot-Kopie verwendet werden soll.
8. Wählen Sie Copy Snapshot (Snapshot kopieren) aus.

Kopieren eines DB-Cluster-Snapshots mit dem AWS CLI

Sie können einen DB-Snapshot mit dem AWS CLI -Befehl copy-db–cluster-snapshot kopieren.

Wenn Sie den Snapshot in eine neue AWS Region kopieren, führen Sie den Befehl in der neuen Region aus.

Verwenden Sie die folgenden Beschreibungen und Beispiele von Parametern, um zu ermitteln, welche Parameter beim Kopieren eines Snapshots mit der AWS CLI verwendet werden sollen.

- `--source-db-cluster-snapshot-identifier` – Der Bezeichner des Quell-DB-Snapshots.
  - Wenn sich der Quell-Snapshot in derselben AWS Region wie die Kopie befindet, geben Sie eine gültige DB-Snapshot-ID an, z. `neptune:instance1-snapshot-20130805` B.
  - Wenn sich der Quell-Snapshot in einer anderen AWS Region als die Kopie befindet, geben Sie einen gültigen DB-Snapshot-ARN an wie`arn:aws:neptune:us-west-2:123456789012:snapshot:instance1-snapshot-20130805`.
  - Wenn Sie aus einem freigegebenen manuellen DB-Snapshot kopieren, muss dieser Parameter der Amazon-Ressourcenname (ARN) des freigegebenen DB-Snapshots sein.
  - Wenn Sie einen verschlüsselten Snapshot kopieren, muss dieser Parameter das ARN-Format für die AWS Quellregion haben und mit dem `SourceDBSnapshotIdentifier` im `PreSignedUrl` Parameter übereinstimmen.
- `--target-db-cluster-snapshot-identifier` – Die ID für die neue Kopie des verschlüsselten DB-Snapshots.
- `--kms-key-id`— — Die AWS KMS Schlüssel-ID für einen verschlüsselten DB-Snapshot. Die AWS KMS Schlüssel-ID ist der Amazon-Ressourcenname (ARN), die AWS KMS Schlüssel-ID oder der AWS KMS Schlüsselalias für den AWS KMS Verschlüsselungsschlüssel.
  - Wenn Sie einen verschlüsselten DB-Snapshot von Ihrem AWS Konto kopieren, können Sie einen Wert für diesen Parameter angeben, um die Kopie mit einem neuen AWS KMS Verschlüsselungsschlüssel zu verschlüsseln. Wenn Sie keinen Wert für diesen Parameter angeben, wird die Kopie des DB-Snapshots mit demselben AWS KMS Schlüssel verschlüsselt wie der Quell-DB-Snapshot.
  - Sie können diesen Parameter nicht verwenden, um eine verschlüsselte Kopie eines unverschlüsselten Snapshots zu erstellen. Wenn Sie dies versuchen, wird ein Fehler generiert.
  - Wenn Sie einen verschlüsselten Snapshot in eine andere AWS Region kopieren, müssen Sie einen AWS KMS Schlüssel für die AWS Zielregion angeben. AWS KMS Verschlüsselungsschlüssel sind spezifisch für die AWS Region, in der sie erstellt wurden, und Sie können keine Verschlüsselungsschlüssel aus einer AWS Region in einer anderen AWS Region verwenden.
- `--source-region`— — Die ID der AWS Region, in der sich der Quell-DB-Snapshot befindet. Wenn Sie einen verschlüsselten Snapshot in eine andere AWS Region kopieren, müssen Sie diese Option angeben.

- `--region`— — Die ID der AWS Region, in die Sie den Snapshot kopieren. Wenn Sie einen verschlüsselten Snapshot in eine andere AWS Region kopieren, müssen Sie diese Option angeben.

Example Unverschlüsselte Quelle in derselben Region

Der folgende Code erstellt eine Kopie eines Snapshots mit dem neuen Namen `mydbsnapshotcopy` von der `us-east-1` AWS Region zur `us-west-2` Region.

Für Linux, OS X oder Unix:

```
aws neptune copy-db-cluster-snapshot \
  --source-db-cluster-snapshot-identifier instance1-snapshot-20130805 \
  --target-db-cluster-snapshot-identifier mydbsnapshotcopy
```

Für Windows:

```
aws neptune copy-db-cluster-snapshot ^
  --source-db-cluster-snapshot-identifier instance1-snapshot-20130805 ^
  --target-db-cluster-snapshot-identifier mydbsnapshotcopy
```

Example Unverschlüsselte Quelle in andere Region

Der folgende Code erstellt eine Kopie eines Snapshots mit dem neuen Namen `mydbsnapshotcopy` von der `us-east-1` AWS Region zur `us-west-2` Region. Führen Sie den Befehl in der `us-west-2`-Region aus.

Für Linux, OS X oder 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
```

Für 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 Verschlüsselte Quelle in andere Region

Im folgenden Codebeispiel wird ein verschlüsselter DB-Snapshot von der `us-east-1` AWS Region in die `us-west-2` Region kopiert. Führen Sie den Befehl in der `us-west-2`-Region aus.

Für Linux, OS X oder 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
```

Für 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
```

Freigeben eines DB-Cluster-Snapshots

Mithilfe von Neptune können Sie einen manuellen DB-Cluster-Snapshot wie folgt freigeben:

- Durch die gemeinsame Nutzung eines manuellen DB-Cluster-Snapshots, ob verschlüsselt oder unverschlüsselt, können autorisierte AWS Konten den Snapshot kopieren.
- Durch die gemeinsame Nutzung eines unverschlüsselten manuellen DB-Cluster-Snapshots können autorisierte AWS Konten einen DB-Cluster direkt aus dem Snapshot wiederherstellen, anstatt eine Kopie davon zu erstellen und aus diesem wiederherzustellen. Verschlüsselte Snapshots können nicht direkt wiederhergestellt werden. Sie müssen zuerst kopiert und dann aus der Kopie wiederhergestellt werden.

> **Note**
>
> Um einen automatisierten DB-Cluster-Snapshot freizugeben, erstellen Sie einen manuellen DB-Cluster-Snapshot, indem Sie den automatisierten Snapshot kopieren und diese Kopie dann freigeben.

Weitere Informationen zum Wiederherstellen eines DB-Clusters aus einem DB-Cluster-Snapshot finden Sie unter Sie führen Sie eine Wiederherstellung aus einem Snapshot durch.

Sie können einen manuellen Snapshot mit bis zu 20 anderen AWS Konten teilen. Sie können einen unverschlüsselten manuellen Snapshot auch öffentlich teilen, sodass der Snapshot für alle AWS Konten verfügbar ist. Achten Sie bei der Freigabe eines Snapshots als öffentlich darauf, dass in Ihren öffentlichen Snapshots keine privaten Informationen enthalten sind.

> **Note**
>
> Wenn Sie einen DB-Cluster aus einem gemeinsam genutzten Snapshot mithilfe der AWS Command Line Interface (AWS CLI) - oder Neptune-API wiederherstellen, müssen Sie den Amazon-Ressourcennamen (ARN) des gemeinsam genutzten Snapshots als Snapshot-ID angeben.

Themen

- Freigeben eines verschlüsselten DB-Cluster-Snapshots

- Freigeben eines DB-Cluster-Snapshots

Freigeben eines verschlüsselten DB-Cluster-Snapshots

Sie können DB-Cluster-Snapshots freigeben, die während der Speicherung mittels des AES-256-Verschlüsselungsalgorithmus verschlüsselt wurden. Weitere Informationen finden Sie unter Neptun-Ressourcen im Ruhezustand verschlüsseln. Hierzu müssen Sie die folgenden Schritte ausführen:

1. Teilen Sie den Verschlüsselungsschlüssel AWS Key Management Service (AWS KMS), der zur Verschlüsselung des Snapshots verwendet wurde, mit allen Konten, die auf den Snapshot zugreifen möchten.

   Sie können AWS KMS Verschlüsselungsschlüssel mit einem anderen AWS Konto teilen, indem Sie das andere Konto zur KMS-Schlüsselrichtlinie hinzufügen. Details zum Aktualisieren einer Schlüsselrichtlinie finden Sie unter Schlüsselrichtlinien im AWS KMS -Entwicklerhandbuch. Ein Beispiel für die Erstellung einer Schlüsselrichtlinie finden Sie unter Erstellen einer IAM-Richtlinie, um das Kopieren des verschlüsselten Snapshots zu ermöglichen an späterer Stelle in diesem Thema.
2. Verwenden Sie die AWS-Managementkonsole AWS CLI, oder Neptune-API, um den verschlüsselten Snapshot mit den anderen Konten zu teilen.

Die folgenden Einschränkungen gelten für die Freigabe verschlüsselter Snapshots:

- Sie können verschlüsselte Snapshots nicht als öffentlich freigeben.
- Sie können keinen Snapshot teilen, der mit dem AWS KMS Standardverschlüsselungsschlüssel des AWS Kontos verschlüsselt wurde, das den Snapshot geteilt hat.

Zugriff auf einen AWS KMS Verschlüsselungsschlüssel zulassen

Damit ein anderes AWS Konto einen verschlüsselten DB-Cluster-Snapshot kopieren kann, der von Ihrem Konto gemeinsam genutzt wurde, muss das Konto, mit dem Sie Ihren Snapshot teilen, Zugriff auf den KMS-Schlüssel haben, mit dem der Snapshot verschlüsselt wurde. Um einem anderen AWS Konto Zugriff auf einen AWS KMS Schlüssel zu gewähren, aktualisieren Sie die Schlüsselrichtlinie für den KMS-Schlüssel mit dem ARN des AWS Kontos, für das Sie die Daten teilen, als `Principal` in der KMS-Schlüsselrichtlinie. Lassen Sie anschließend die Aktion `kms:CreateGrant` zu. Allgemeine

Informationen dazu finden Sie unter Benutzern in anderen Konten die Verwendung eines KMS-Schlüssels erlauben im AWS Key Management Service -Entwicklerhandbuch.

Nachdem Sie einem AWS Konto Zugriff auf Ihren KMS-Verschlüsselungsschlüssel gewährt haben, muss dieses AWS Konto zum Kopieren Ihres verschlüsselten Snapshots einen IAM-Benutzer erstellen, falls es noch keinen hat. Die KMS-Sicherheitseinschränkungen lassen die Verwendung einer AWS Root-Kontoidentität zu diesem Zweck nicht zu. Das AWS Konto muss diesem IAM-Benutzer außerdem eine IAM-Richtlinie zuordnen, die es dem IAM-Benutzer ermöglicht, mithilfe Ihres KMS-Schlüssels einen verschlüsselten DB-Cluster-Snapshot zu kopieren.

Im folgenden Beispiel für eine Schlüsselrichtlinie ist der Benutzer `111122223333` der Besitzer des KMS-Verschlüsselungsschlüssels und der Benutzer `444455556666` ist das Konto, für das der Schlüssel freigegeben wird. Diese aktualisierte Schlüsselrichtlinie gewährt dem AWS Konto Zugriff auf den KMS-Schlüssel, indem sie den ARN für die AWS Root-Kontoidentität für den Benutzer `444455556666` als `Principal` für die Richtlinie einbezieht und die `kms:CreateGrant` Aktion zulässt.

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
                }
            }
        }
    ]
}
```

Erstellen einer IAM-Richtlinie, um das Kopieren des verschlüsselten Snapshots zu ermöglichen

Sobald das externe AWS Konto Zugriff auf Ihren KMS-Schlüssel hat, kann der Besitzer dieses Kontos eine Richtlinie erstellen, die es einem für das Konto erstellten IAM-Benutzer ermöglicht, einen verschlüsselten Snapshot zu kopieren, der mit diesem KMS-Schlüssel verschlüsselt wurde.

Das folgende Beispiel zeigt eine Richtlinie, die einem IAM-Benutzer für das AWS -Konto `444455556666` angefügt werden kann. Dies ermöglicht dem IAM-Benutzer das Kopieren eines freigegebenen Snapshots aus dem AWS -Konto `111122223333`, das mit dem KMS-Schlüssel `c989c1dd-a3f2-4a5d-8d96-e793d082ab26` in der Region `us-west-2` verschlüsselt wurde.

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
                }
            }
        }
    ]
}
```

Details zum Aktualisieren einer Schlüsselrichtlinie finden Sie unter Schlüsselrichtlinien im AWS Key Management Service -Entwicklerhandbuch.

Freigeben eines DB-Cluster-Snapshots

Sie können einen DB-Cluster-Snapshot mit der AWS-Managementkonsole AWS CLI, der oder der Neptune-API teilen.

Verwenden der Konsole zum Freigeben eines DB-Cluster-Snapshots

Mithilfe der Neptune-Konsole geben Sie einen manuellen DB-Cluster-Snapshot für bis zu 20 AWS -Konten frei. Sie können auch die Freigabe eines manuellen Snapshots mit einem oder mehreren Konten beenden.

So geben Sie einen manuellen DB-Cluster-Snapshot frei

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die Amazon Neptune Neptune-Konsole zu Hausehttps://console.aws.amazon.com/neptune/.
2. Wählen Sie im Navigationsbereich die Option Snapshots.
3. Wählen Sie den manuellen Snapshot, den Sie freigeben möchten.
4. Wählen Sie Actions (Aktionen), Share Snapshot (Snapshot freigeben) aus.
5. Wählen Sie für DB Snapshot Visibility (Sichtbarkeit des DB-Snapshots) eine der folgenden Optionen.

   - Wenn die Quelle nicht verschlüsselt ist, wählen Sie Öffentlich, um allen AWS -Konten die Wiederherstellung eines DB-Clusters aus Ihrem manuellen DB-Cluster-Snapshot zu gestatteten. Oder wählen Sie Privat, um nur von Ihnen angegebene AWS Konten die Wiederherstellung eines DB-Clusters aus Ihrem manuellen DB-Cluster-Snapshot zuzulassen.

     > **Warning**
     >
     > Wenn Sie die Sichtbarkeit von DB-Snapshots auf Öffentlich setzen, können alle AWS Konten einen DB-Cluster aus Ihrem manuellen DB-Cluster-Snapshot wiederherstellen und haben Zugriff auf Ihre Daten. Geben Sie keine manuellen DB-Cluster-Snapshots als Public (Öffentlich) frei, die private Informationen enthalten.

   - Ist die Quelle verschlüsselt, ist DB Snapshot Visibility (Sichtbarkeit des DB-Snapshots) auf Private (Privat) festgelegt, da verschlüsselte Snapshots nicht als öffentlich freigegeben werden können.
6. Geben Sie AWS unter Konto-ID die AWS Konto-ID für ein Konto ein, dem Sie die Wiederherstellung eines DB-Clusters aus Ihrem manuellen Snapshot erlauben möchten. Wählen Sie dann Add (Hinzufügen). Wiederholen Sie den Vorgang, um weitere AWS Kontokennungen hinzuzufügen, bis zu 20 AWS Konten.

Wenn Ihnen beim Hinzufügen einer AWS Konto-ID zur Liste der zulässigen Konten ein Fehler unterläuft, können Sie diese aus der Liste löschen, indem Sie rechts neben der falschen AWS Konto-ID die Option Löschen auswählen.

7. Nachdem Sie Kennungen für alle AWS Konten hinzugefügt haben, denen Sie die Wiederherstellung des manuellen Snapshots erlauben möchten, wählen Sie Speichern.

Um die gemeinsame Nutzung eines manuellen DB-Cluster-Snapshots mit einem AWS Konto zu beenden

1. Öffnen Sie die Amazon Neptune Neptune-Konsole zu Hausehttps://console.aws.amazon.com/neptune/.
2. Wählen Sie im Navigationsbereich die Option Snapshots.
3. Wählen Sie den manuellen Snapshot, für den Sie die Freigabe beenden möchten.
4. Wählen Sie die Option Aktionen und anschließend Share Snapshot (Snapshot freigeben) aus.
5. Um die Erlaubnis für ein AWS Konto zu entfernen, wählen Sie Löschen als AWS Konto-ID für dieses Konto aus der Liste der autorisierten Konten.
6. Wählen Sie Speichern.

Löschen eines Neptune-Snapshots

Sie können einen DB-Snapshot mit der AWS-Managementkonsole, der oder der AWS CLI Neptune-Management-API löschen:

Löschen über die Konsole

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die Amazon Neptune Neptune-Konsole zu Hausehttps://console.aws.amazon.com/neptune/.
2. Wählen Sie im Navigationsbereich die Option Snapshots.
3. Wählen Sie den DB-Snapshot aus, den Sie löschen möchten.
4. Wählen Sie unter Actions (Aktionen) die Option Delete Snapshot (Snapshot löschen) aus.
5. Wählen Sie auf der Bestätigungsseite die Option Delete (Löschen) aus.

Löschen mit dem AWS CLI

Sie können einen DB-Snapshot auch mit dem Befehl AWS CLI delete_db_cluster_snapshot löschen, indem Sie den zu löschenden Snapshot mithilfe des `--db-snapshot-identifier`-Parameters identifizieren:

Für Linux, OS X oder Unix:

```
aws neptune delete-db-cluster-snapshot \
    --db-snapshot-identifier <name-of-the-snapshot-to-delete>
```

Für Windows:

```
aws neptune delete-db-cluster-snapshot ^
    --db-snapshot-identifier <name-of-the-snapshot-to-delete>
```

Löschen mit der Neptune-Management-API

Sie können einen der verwenden, SDKs um einen DB-Snapshot zu löschen, indem Sie die DBClusterSchnappschuss löschen API aufrufen und die `DBSnapshotIdentifier` Parameter verwenden, um den zu löschenden DB-Snapshot zu identifizieren.

Bewährte Methoden: Optimale Nutzung von Neptune

Im Folgenden finden Sie einige allgemeine Empfehlungen für die Arbeit mit Amazon Neptune. Verwenden Sie diese Informationen, um schnell Empfehlungen für die Verwendung von Amazon Neptune und zur Maximierung der Leistung zu finden.

Inhalt

- Grundlegende Anleitungen für den Amazon-Neptune-Betrieb
  - Bewährte Methoden für die Sicherheit in Amazon Neptune
  - Vermeiden Sie verschiedene DB-Instance-Klassen in einem Cluster
  - Vermeiden Sie wiederholte Neustarts während eines Massenladevorgangs
  - Aktivieren Sie den OSGP-Index, wenn Sie eine große Anzahl von Prädikaten haben
  - Vermeiden Sie lang laufende Transaktionen, wenn möglich
  - Bewährte Methoden zur Verwendung von Neptune-Metriken
  - Bewährte Methoden für das Optimieren von Neptune-Abfragen
  - Load Balancing über Lesereplikate hinweg
  - Schnelleres Laden mithilfe einer vorübergehend größeren Instance
  - Ändern Sie die Größe Ihrer Writer-Instance, indem Sie ein Failover auf ein Lesereplikat durchführen
  - Wiederholen des Uploads nach „Data Prefetch Task Interrupted“-Fehler
- Allgemeine bewährte Methoden für die Verwendung von Gremlin mit Neptune
  - Heartbeat-Konfiguration für Neptune Serverless
  - Strukturieren von Upsert-Abfragen, um die Vorteile der DFE-Engine zu nutzen
  - Testen Sie den Gremlin-Code in dem Kontext, in dem Sie ihn einsetzen werden
  - Erstellen von effizienten Multi-Thread-Gremlin-Schreibvorgängen
  - Bereinigung von Datensätzen mit der Eigenschaft „Erstellungszeit (Creation Time)“
  - Verwendung der datetime () -Methode für Groovy-Zeitdaten
  - Verwendung der nativen Datum- und Uhrzeitangaben für GLV-Zeitdaten
- Bewährte Methoden für die Verwendung des Gremlin-Java-Clients mit Neptune
  - Wiederverwenden des Client-Objekts in mehreren Threads
  - Erstellen separater Gremlin-Java-Client-Objekte für Lese- und Schreibendpunkte

- Hinzufügen mehrerer Lesereplikat-Endpunkte zu einem Gremlin-Java-Verbindungs-Pool
- Schließen des Clients zur Vermeidung des Verbindungs-Limits
- Erstellen einer neuen Verbindung nach einem Failover
- Setzen Sie maxInProcessPerConnection und maxSimultaneousUsagePerConnection auf den selben Wert.
- Senden von Abfragen an den Server als Bytecode und nicht als Zeichenfolgen
- Verwenden Sie den von einer Abfrage zurückgegebenen Oder-Iterator immer vollständig ResultSet
- Massenweises Hinzufügen von Scheitelpunkten und Edges in Stapeln
- Deaktivieren von DNS-Caching in der Java Virtual Machine
- Optionales Festlegen von Zeitüberschreitungen auf Abfrageebene
- Fehlerbehebung für java.util.concurrent.TimeoutException

- Bewährte Methoden für Neptune mit openCypher und Bolt
  - Erstellen einer neuen Verbindung nach einem Failover
  - Verbindungsverwaltung für langlebige Anwendungen
  - Verbindungsmanagement für AWS Lambda
  - Bevorzugung direktionaler gegenüber bidirektionalen Edges in Abfragen
  - Neptune unterstützt nicht mehrere gleichzeitige Abfragen in einer Transaktion
  - Schließen Sie anschließend Treiberobjekte
  - Verwenden expliziter Transaktionsmodi zum Lesen und Schreiben
    - Nur-Lese-Transaktionen
    - Mutationstransaktionen
  - Wiederholungslogik für Ausnahmen
  - Legen Sie mithilfe einer einzigen SET-Klausel mehrere Eigenschaften gleichzeitig fest
    - Verwenden Sie die SET-Klausel, um mehrere Eigenschaften gleichzeitig zu entfernen
  - Parametrisierte Abfragen verwenden
  - Verwenden Sie in der UNWIND-Klausel abgeflachte Maps anstelle von verschachtelten Maps
  - Platzieren Sie restriktivere Knoten in VLP-Ausdrücken (Variable-Length Path) auf der linken Seite
  - Vermeiden Sie redundante Prüfungen von Knotenbezeichnungen, indem Sie detaillierte Beziehungsnamen verwenden

- Geben Sie nach Möglichkeit Kantenbeschriftungen an
- Vermeiden Sie nach Möglichkeit die WITH-Klausel
- Platzieren Sie restriktive Filter so früh wie möglich in der Abfrage
- Prüfen Sie explizit, ob Eigenschaften vorhanden sind
- Verwenden Sie keinen benannten Pfad (es sei denn, er ist erforderlich)
- Vermeiden Sie COLLECT (DISTINCT ())
- Ziehen Sie beim Abrufen aller Eigenschaftswerte die Eigenschaftenfunktion der Suche nach einzelnen Eigenschaften vor
- Führen Sie statische Berechnungen außerhalb der Abfrage durch
- Batch-Eingaben mit UNWIND anstelle von Einzelanweisungen
- Verwenden Sie lieber „Benutzerdefiniert“ IDs für Knoten/Beziehungen
- Vermeiden Sie es, ~id-Berechnungen in der Abfrage durchzuführen
- Aktualisieren/Zusammenführen mehrerer Knoten

- Bewährte Methoden in Neptune für die Verwendung von SPARQL
  - Standardmäßige Abfrage aller benannten Graphen
  - Angeben eines benannten Graphen für Load
  - Auswählen zwischen FILTER, FILTER... IN und VALUES in Ihren Abfragen
- Bewährte Methoden für die Verwendung von Neptune mit Amazon ECS und Amazon EKS
  - Verwalten Sie Verbindungspools und den Lebenszyklus in containerisierten Umgebungen
  - Erledigen Sie den Host-Austausch und den Verbindungsabbruch
  - Konfigurieren Sie Netzwerke, Sicherheitsgruppen und IAM-Authentifizierung
- Bewährte Methoden für die Verwendung des AWS CLI und SDKs mit Neptune

Grundlegende Anleitungen für den Amazon-Neptune-Betrieb

Im Folgenden finden Sie einige grundlegenden Anleitungen für den Betrieb, die bei der Arbeit mit Neptune befolgt werden sollten.

- Machen Sie sich mit Neptune-DB-Instances vertraut, damit Sie sie entsprechend Ihren Leistungs- und Anwendungsanforderungen dimensionieren können. Siehe Amazon-Neptune-DB-Cluster und -Instances.

- Sie müssen die Nutzung von CPU und Arbeitsspeicher überwachen. Auf diese Weise können Sie leichter erkennen, wann auf eine DB-Instance-Klasse mit mehr CPU- oder Speicherkapazität migriert werden sollte, um die erforderliche Abfrageleistung zu erzielen. Sie können Amazon so einrichten CloudWatch , dass Sie benachrichtigt werden, wenn sich die Nutzungsmuster ändern oder wenn Sie sich der Kapazität Ihrer Bereitstellung nähern. Auf diese Weise können Sie leichter die Leistung und Verfügbarkeit des Systems wahren. Weitere Informationen hierzu finden Sie unter Überwachen von Instances und Überwachen von Neptune.

  Da Neptune über einen eigenen Speichermanager verfügt, ist es normal, eine relativ geringe Speichernutzung zu sehen, auch wenn die CPU-Auslastung hoch ist. Das Auftreten out-of-memory von Ausnahmen bei der Ausführung von Abfragen ist der beste Indikator dafür, dass Sie den freien Speicherplatz erhöhen müssen.
- Aktivieren Sie automatische Sicherungen und richten Sie das Sicherungsfenster so ein, dass diese zu einem günstigen Zeitpunkt durchgeführt werden.
- Testen Sie den Failover für Ihre DB-Instance, um zu verstehen, wie lange der Vorgang für Ihren Anwendungsfall dauert. Auch können Sie leichter sicherstellen, dass die Anwendung, mit der auf Ihre DB-Instance zugegriffen wird, nach einem Failover automatisch eine Verbindung mit der neuen DB-Instance herstellen kann.
- Führen Sie Ihren Client und den Neptune-Cluster nach Möglichkeit in derselben Region und VPC aus, da regionsübergreifende Verbindungen mit VPC-Peering zu Verzögerungen bei den Query-Antworten führen können. Bei Query-Antworten im einstelligen Millisekundenbereich müssen sich Client und Neptune-Cluster in derselben Region und VPC befinden.
- Wenn Sie eine Lesereplikat-Instance erstellen, sollte sie mindestens so groß sein wie die primäre Writer-Instance. Dies hilft, die Replikationsverzögerung in Grenzen zu halten und vermeidet einen Neustart des Replicas. Siehe Vermeiden Sie verschiedene DB-Instance-Klassen in einem Cluster.
- Bevor Sie auf eine neue Hauptversion der Engine aktualisieren, sollten Sie Ihre Anwendung darauf testen, bevor Sie das Upgrade durchführen. Sie können dies tun, indem Sie Ihren DB-Cluster klonen, so dass auf dem Klon-Cluster die neue Engine-Version ausgeführt wird, und anschließend Ihre Anwendung auf dem Klon testen.
- Um Failovers zu erleichtern, sollten alle Instances idealerweise gleich groß sein.

Themen

- Bewährte Methoden für die Sicherheit in Amazon Neptune
- Vermeiden Sie verschiedene DB-Instance-Klassen in einem Cluster
- Vermeiden Sie wiederholte Neustarts während eines Massenladevorgangs

- Aktivieren Sie den OSGP-Index, wenn Sie eine große Anzahl von Prädikaten haben
- Vermeiden Sie lang laufende Transaktionen, wenn möglich
- Bewährte Methoden zur Verwendung von Neptune-Metriken
- Bewährte Methoden für das Optimieren von Neptune-Abfragen
- Load Balancing über Lesereplikate hinweg
- Schnelleres Laden mithilfe einer vorübergehend größeren Instance
- Ändern Sie die Größe Ihrer Writer-Instance, indem Sie ein Failover auf ein Lesereplikat durchführen
- Wiederholen des Uploads nach „Data Prefetch Task Interrupted“-Fehler

Bewährte Methoden für die Sicherheit in Amazon Neptune

Verwenden Sie AWS Identity and Access Management (IAM-) Konten, um den Zugriff auf Neptune-API-Aktionen zu kontrollieren. Steuern Sie Aktionen, die Neptune-Ressourcen erstellen, ändern oder löschen (beispielsweise DB-Instances, Sicherheitsgruppen, Optionsgruppen oder Parametergruppen), sowie Aktionen, die allgemeine administrative Aktionen (wie die Sicherung und Wiederherstellung von DB-Instances) ausführen.

- Verwenden Sie nach Möglichkeit temporäre statt persistenter Anmeldeinformationen.
- Weisen Sie jeder Person, die Amazon Relational Database Service (Amazon RDS)-Ressourcen verwaltet, ein individuelles IAM-Konto zu. Verwenden Sie niemals AWS Konto-Root-Benutzer, um Neptune-Ressourcen zu verwalten. Erstellen Sie einen IAM-Benutzer für jede Person einschließlich Sie selbst.
- Gewähren Sie jedem Benutzer nur den Mindestsatz an Berechtigungen, die für die Ausführung seiner Aufgaben erforderlich sind.
- Verwenden Sie IAM-Gruppen, um Berechtigungen für mehrere Benutzer effektiv zu verwalten.
- Wechseln Sie regelmäßig die IAM-Anmeldeinformationen.

Weitere Informationen zur Verwendung von IAM für den Zugriff auf Neptune-Ressourcen finden Sie unter Sicherung Ihrer Amazon Neptune Neptune-Datenbank. Allgemeine Informationen über das Arbeiten mit IAM finden Sie unter AWS Identity and Access Management und Bewährte Methoden für IAM im IAM-Benutzerhandbuch.

Vermeiden Sie verschiedene DB-Instance-Klassen in einem Cluster

Wenn Ihr DB-Cluster Instances verschiedener Klassen enthält, können im Laufe der Zeit Probleme auftreten. Das häufigste Problem besteht darin, dass eine kleine Reader-Instance aufgrund von Replikationsverzögerungen in einen Zyklus wiederholter Neustarts geraten kann. Wenn ein Reader-Knoten eine schwächere DB-Instance-Klassenkonfiguration als die einer Writer-DB-Instance hat, kann das Volumen der Änderungen so groß sein, dass der Reader damit nicht Schritt halten kann.

> **Important**
>
> Um wiederholte Neustarts aufgrund von Verzögerungen bei der Replikation zu vermeiden, konfigurieren Sie Ihren DB-Cluster so, dass alle Instances dieselbe Instance-Klasse (Größe) haben.

Sie können die Verzögerung zwischen der Writer-Instance (der primären) und den Readern in Ihrem DB-Cluster anhand der `ClusterReplicaLag` Metrik in Amazon sehen CloudWatch. Mit der `VolumeWriteIOPs`-Metrik können Sie auch Spitzen von Schreibaktivitäten in Ihrem Cluster erkennen, die zu Verzögerungen bei der Replikation führen können.

Vermeiden Sie wiederholte Neustarts während eines Massenladevorgangs

Wenn es aufgrund von Replikationsverzögerungen während eines Massenladevorgangs zu einem Zyklus wiederholter Lesereplikat-Neustarts kommt, können Ihre Replikate wahrscheinlich nicht mit dem Writer im DB-Cluster Schritt halten.

Sie können die Reader so skalieren, dass sie größer als der Writer sind. Sie können sie auch während des Massenladens vorübergehend entfernen und nach dem Abschluss des Vorgangs erneut erstellen.

Aktivieren Sie den OSGP-Index, wenn Sie eine große Anzahl von Prädikaten haben

Wenn Ihr Datenmodell eine große Anzahl unterschiedlicher Prädikate enthält, führt dies möglicherweise zu einer reduzierten Leistung und höheren Betriebskosten.

In diesem Fall können Sie die Leistung verbessern, indem Sie den OSGP-Index aktivieren. Siehe Der OSGP-Index.

Vermeiden Sie lang laufende Transaktionen, wenn möglich

Lang andauernde (Nur-Lese- oder Lese-Schreib-) Transaktionen können zu unerwarteten Problemen der folgenden Art führen:

Eine lang andauernde Transaktion auf einer Reader- oder Writer-Instance mit gleichzeitigen Schreibvorgängen kann zu einer großen Anhäufung verschiedener Datenversionen führen. Dies kann zu höheren Latenzen bei Leseabfragen führen, die einen großen Teil ihrer Ergebnisse herausfiltern.

In einigen Fällen können die im Laufe von Stunden angesammelten Versionen dazu führen, dass neue Schreibvorgänge gedrosselt werden.

Eine lang andauernde Lese- und Schreibtransaktion mit vielen Schreibvorgängen kann ebenfalls zu Problemen führen, wenn die Instance neu gestartet wird. Wenn eine Instance aufgrund eines Wartungsereignisses oder eines Absturzes neu gestartet wird, werden alle nicht festgeschriebenen Schreibvorgänge zurückgesetzt. Solche Undo-Operationen werden normalerweise im Hintergrund ausgeführt und verhindern nicht, dass die Instance wieder hochgefahren wird. Aber alle neuen Schreibvorgänge, die mit den Vorgängen, die zurückgesetzt werden, in Konflikt geraten, schlagen dann fehl.

Wenn beispielsweise dieselbe Abfrage erneut versucht wird, nachdem die Verbindung beim vorherigen Durchlauf getrennt wurde, schlägt dies möglicherweise fehl, wenn die Instance neu gestartet wird.

Die Zeit, die für das Rückgängigmachen von Vorgängen benötigt wird, ist proportional zum Umfang der Änderungen.

Bewährte Methoden zur Verwendung von Neptune-Metriken

Sie können die Metriken überwachen, die für Ihren Neptune-DB-Cluster verfügbar sind, um Leistungsprobleme aufgrund unzureichender Ressourcen und anderer häufiger Engpässe zu identifizieren.

Überwachen Sie die Leistungsmetriken regelmäßig, um die Durchschnitts-, Höchst- und Mindestwerte für eine Vielzahl von Zeitbereichen zu sammeln. Auf diese Weise können Sie feststellen, wenn die Leistung nachlässt. Mithilfe dieser Daten können Sie CloudWatch Amazon-Alarme für bestimmte Metrik-Schwellenwerte einrichten, sodass Sie benachrichtigt werden, wenn diese erreicht werden.

Wenn Sie einen neuen DB-Cluster einrichten und in diesem ein typisches Workload ausführen, versuchen Sie, die Durchschnitts-, Höchst- und Mindestwerte aller Leistungsmetriken in

unterschiedlichen Intervallen zu erfassen (beispielsweise eine Stunde, 24 Stunden, eine Woche, zwei Wochen). Auf diese Weise erhalten Sie eine Vorstellung davon, was normal ist. Dies hilft, um Vergleichswerte für Betriebsstunden während und außerhalb von Spitzenbelastungen zu erhalten. Sie können diese Informationen anschließend verwenden, um festzustellen, wann die Leistung unter Standardwerte absinkt, und die Alarme entsprechend festlegen.

Weitere Informationen zum Anzeigen von Neptune-Metriken finden Sie unter Überwachung von Neptune mit Amazon CloudWatch.

Es folgen die wichtigsten Metriken, mit denen gestartet werden soll:

- BufferCacheHitRatio— Der Prozentsatz der Anfragen, die vom Puffer-Cache bedient werden. Cache-Fehler erhöhen die Latenz bei der Abfrageausführung erheblich. Wenn die Cache-Trefferquote unter 99,9 % liegt und die Latenz ein Problem für Ihre Anwendung darstellt, sollten Sie ein Upgrade des Instance-Typs in Betracht ziehen, um mehr Daten im Arbeitsspeicher zwischenzuspeichern.
- CPU-Nutzung – Prozentsatz der verwendeten Verarbeitungskapazität des Computers. Hohe Werte für den CPU-Verbrauch können abhängig von Ihren Zielen für die Abfrageleistung angemessen sein.
- Freisetzbarer Arbeitsspeicher – Größe des RAM, der in der DB-Instance verfügbar ist (in Megabyte). Neptune hat seinen eigenen Speichermanager, daher ist diese Metrik möglicherweise niedriger als erwartet. Ein gutes Zeichen dafür, dass Sie ein Upgrade Ihrer Instance-Klasse auf eine mit mehr RAM in Betracht ziehen sollten, ist, wenn Abfragen häufig out-of-memory Ausnahmen auslösen.

Die rote Linie in der Metrik der Registerkarte Monitoring (Überwachung) kennzeichnet 75 % für CPU- und Arbeitsspeichermetriken. Wenn der Speicherverbrauch der Instance diese Linie häufig überschreitet, prüfen Sie Ihren Workload und ziehen Sie ein Upgrade Ihrer Instance in Betracht, um die Abfrageleistung verbessern.

Bewährte Methoden für das Optimieren von Neptune-Abfragen

Eine der besten Möglichkeiten zur Verbesserung der Neptune-Leistung besteht darin, die am häufigsten verwendeten und ressourcenintensivsten Abfragen so anzupassen, dass ihre Ausführung weniger aufwändig wird.

Informationen über das Anpassen von Gremlin-Abfragen finden Sie unter Gremlin-Abfragehinweise und Optimieren von Gremline-Abfragen. Informationen über das Anpassen von SPARQL-Abfragen finden Sie unter SPARQL-Abfragehinweise.

Load Balancing über Lesereplikate hinweg

Das Round-Robin-Routing für den Reader-Endpunkt funktioniert durch Ändern des Hosts, auf den der DNS-Eintrag verweist. Der Client muss eine neue Verbindung herstellen und den DNS-Eintrag auflösen, um eine Verbindung zu einer neuen Read Replica herzustellen, da WebSocket Verbindungen oft über lange Zeiträume bestehen bleiben.

Um verschiedene Read Replicas für aufeinanderfolgende Anforderungen abzurufen, stellen Sie sicher, dass der Client den DNS-Eintrag bei jeder Verbindung auflöst. Dies erfordert möglicherweise das Schließen der Verbindung und das erneute Verbinden mit dem Leser-Endpunkt.

Sie können einen Lastausgleich auch über Read Replicas hinweg durchführen, indem Sie explizit Verbindungen zu Instance-Endpunkten herstellen.

Schnelleres Laden mithilfe einer vorübergehend größeren Instance

Ihre Ladeleistung steigt mit größeren Instances. Wenn Sie keinen großen Instance-Typ verwenden, aber dennoch die Ladegeschwindigkeiten erhöhen möchten, können Sie eine größere Instance zum Laden verwenden und diese dann löschen.

> **Note**
>
> Die folgende Vorgehensweise gilt für einen neuen Cluster. Wenn Sie einen vorhandenen Cluster haben, können Sie eine neue größere Instance hinzufügen und diese dann zu einer primären DB-Instance machen.

So laden Sie Daten mit einer größeren Instance-Größe:

1. Erstellen Sie einen Cluster mit einer einzelnen `r5.12xlarge`-Instance. Diese Instance ist die primäre DB-Instance.
2. Erstellen Sie mindestens ein Lesereplikat derselben Größe (`r5.12xlarge`).

   Sie können die Lesereplikate in einer kleineren Größe erstellen. Wenn sie jedoch nicht groß genug sind, um mit den Schreibvorgängen der primären Instance Schritt zu halten, müssen sie

möglicherweise häufig neu gestartet werden. Die daraus resultierenden Ausfallzeiten reduzieren die Leistung erheblich.

3. Fügen Sie im Bulk-Loader-Befehl `“parallelism” : “OVERSUBSCRIBE”` hinzu, um Neptune anzuweisen, alle verfügbaren CPU-Ressourcen für das Laden zu verwenden (siehe Neptune-Loader-Anforderungsparameter). Der Ladevorgang wird dann so schnell wie I/O möglich fortgesetzt, was in der Regel 60-70% der CPU-Ressourcen erfordert.
4. Laden Sie Ihre Daten mit dem Neptune-Loader. Der Ladevorgang wird auf der primären DB-Instance durchgeführt.
5. Stellen Sie nach Abschluss des Ladens der Daten sicher, dass Sie alle Instances im Cluster auf denselben Instance-Typ herunterskalieren, um zusätzliche Gebühren und wiederholte Neustartprobleme zu vermeiden (siehe Vermeiden Sie unterschiedliche Instance-Größen).

Ändern Sie die Größe Ihrer Writer-Instance, indem Sie ein Failover auf ein Lesereplikat durchführen

Die beste Methode, die Größe einer Instance in Ihrem DB-Cluster, einschließlich der Writer-Instance, zu ändern, besteht darin, eine Lesereplikat-Instance so zu erstellen oder zu ändern, dass sie die gewünschte Größe hat, und dann bewusst auf dieses Lesereplikat umzuschalten. Bei der Ausfallzeit Ihrer Anwendung handelt es sich lediglich um die Zeit, die für die Änderung der IP-Adresse des Writers erforderlich ist. Diese sollte etwa 3 bis 5 Sekunden betragen.

Die Neptune-Verwaltungs-API, die Sie verwenden, um ein absichtliches Failover der aktuellen Writer-Instance auf eine Lesereplikat-Instance durchzuführen, ist Failover DBCluster. Wenn Sie den Gremlin-Java-Client verwenden, müssen Sie nach dem Failover möglicherweise ein neues Client-Objekt erstellen, um die neue IP-Adresse zu übernehmen, wie hier beschrieben.

Stellen Sie sicher, dass Sie alle Ihre Instances auf dieselbe Größe umstellen, um einen Zyklus wiederholter Neustarts zu vermeiden, wie unten beschrieben.

Wiederholen des Uploads nach „Data Prefetch Task Interrupted“-Fehler

Wenn Sie mit dem Massen-Loader Daten in Neptune laden, kann dies gelegentlich zum Status `LOAD_FAILED` führen. In der Antwort auf die Anforderung detaillierter Informationen werden die Meldungen `PARSING_ERROR` und `Data prefetch task interrupted` angezeigt.

```
"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
  }
]
```

Wenn dieser Fehler auftritt, wiederholen Sie einfach die Massen-Upload-Anforderung.

Der Fehler tritt auf, wenn es eine temporäre Unterbrechung gab, die in der Regel nicht durch Ihre Anforderung oder Ihre Daten verursacht wurde, und er kann in der Regel behoben werden, indem die Massen-Upload-Anforderung erneut ausgeführt wird.

Wenn Sie Standardeinstellungen verwenden, nämlich `"mode":"AUTO"` und `"failOnError":"TRUE"`, überspringt der Loader Dateien, die er bereits erfolgreich geladen hat und fährt mit dem Laden von Dateien fort, die noch nicht geladen wurden, als die Unterbrechung stattfand.

Allgemeine bewährte Methoden für die Verwendung von Gremlin mit Neptune

Befolgen Sie diese Empfehlungen bei der Verwendung der Gremlin-Grapht-Traversal-Sprache mit Neptune. Informationen zur Verwendung von Gremlin mit Neptune finden Sie unter the section called “Gremlin”.

Important

In TinkerPop Version 3.4.11 wurde eine Änderung vorgenommen, die die Richtigkeit der Verarbeitung von Abfragen verbessert, die Abfrageleistung jedoch vorerst ernsthaft beeinträchtigen kann.
Zum Beispiel könnte eine Abfrage dieser Art deutlich langsamer ausgeführt werden:

```
g.V().hasLabel('airport').
  order().
    by(out().count(),desc).
  limit(10).
  out()
```

Die Scheitelpunkte nach dem Grenzwertschritt werden aufgrund der Änderung in 3.4.11 jetzt nicht optimal abgerufen. TinkerPop Um dies zu vermeiden, können Sie die Abfrage ändern, indem Sie den Schritt barrier() an einer beliebigen Stelle nach `order().by()` hinzufügen. Beispiel:

```
g.V().hasLabel('airport').
  order().
    by(out().count(),desc).
  limit(10).
  barrier().
  out()
```

TinkerPop 3.4.11 wurde in der Neptune-Engine-Version 1.0.5.0 aktiviert.

Themen

- Heartbeat-Konfiguration für Neptune Serverless
- Strukturieren von Upsert-Abfragen, um die Vorteile der DFE-Engine zu nutzen
- Testen Sie den Gremlin-Code in dem Kontext, in dem Sie ihn einsetzen werden
- Erstellen von effizienten Multi-Thread-Gremlin-Schreibvorgängen
- Bereinigung von Datensätzen mit der Eigenschaft „Erstellungszeit (Creation Time)“
- Verwendung der datetime () -Methode für Groovy-Zeitdaten
- Verwendung der nativen Datum- und Uhrzeitangaben für GLV-Zeitdaten

Heartbeat-Konfiguration für Neptune Serverless

Wenn Sie WebSocket Gremlin-Clients mit Neptune Serverless verwenden, müssen Sie das Ping-Intervall des Clients entsprechend konfigurieren, um stabile Verbindungen während Skalierungsereignissen aufrechtzuerhalten. Der Gremlin-Client verwendet WebSocket Verbindungen und sendet regelmäßig Pings, um zu überprüfen, ob die Verbindung aktiv ist. Der Client erwartet innerhalb des Ping-Intervalls eine Antwort vom Server. Wenn der Server nicht antwortet, schließt der Client die Verbindung automatisch.

Für von Neptune bereitgestellte Instances empfehlen wir, das Ping-Intervall auf 5 Sekunden festzulegen. Für Neptune Serverless-Cluster empfehlen wir, das Ping-Intervall auf mindestens 20 Sekunden festzulegen, um möglichen Verzögerungen bei Skalierungsvorgängen Rechnung zu

tragen. Dieser Parameter steuert, wie lange der Client zwischen Schreibvorgängen auf den Server wartet, bevor er einen Ping sendet, um zu überprüfen, ob die Verbindung noch aktiv ist.

Die Konfiguration dieses Parameters hängt von der Client-Implementierung ab:

Konfiguration des Java-Clients

Konfigurieren Sie für den TinkerPop Java-Gremlin-Client den `keepAliveInterval` Parameter:

```
Cluster.Builder builder = Cluster.build()
    .addContactPoint(endpoint)
    .keepAliveInterval(20000); // Configure ping interval in milliseconds
```

Weitere Informationen zur Java-Treiberkonfiguration finden Sie in der TinkerPop Java-Dokumentation.

Gehen Sie zur Client-Konfiguration

Konfigurieren Sie für den Gremlin Go-Client den `KeepAliveInterval` Parameter:

```
rc, err := driver.NewDriverRemoteConnection(endpoint,
    func(settings *driver.DriverRemoteConnectionSettings) {
        settings.TraversalSource = "g"
        settings.AuthInfo = auth
        settings.KeepAliveInterval = 20 * time.Second // Configure ping interval
        ...
    })
```

Weitere Informationen zur Go-Treiberkonfiguration finden Sie in der TinkerPop Go-Dokumentation.

JavaScript/Node.js Client-Konfiguration

Konfigurieren Sie für den JavaScript/Node.js Gremlin-Client den `pingInterval` Parameter:

```
const gremlin = require('gremlin');
const DriverRemoteConnection = gremlin.driver.DriverRemoteConnection;

const connection = new DriverRemoteConnection(endpoint, {
    traversalSource: 'g',
    pingInterval: 20000  // Configure ping interval in milliseconds
});
```

Weitere Informationen zur JavaScript Treiberkonfiguration finden Sie in der JavaScript TinkerPop Dokumentation.

Python-Client-Konfiguration

Für den Python-Gremlin-Client wird das Ping-Intervall normalerweise auf der Transportschicht verwaltet. Informationen zu den Konfigurationsoptionen finden Sie in der spezifischen Dokumentation zur Transportimplementierung:

```
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))))
```

Weitere Informationen zur Python-Treiberkonfiguration finden Sie in der TinkerPop Python-Dokumentation.

Diese Konfiguration stellt sicher, dass Ihr Client bei Skalierungsereignissen mit Neptune Serverless die Verbindungsstabilität beibehält, wodurch unnötige Verbindungsabbrüche verhindert und die Anwendungszuverlässigkeit verbessert wird.

Strukturieren von Upsert-Abfragen, um die Vorteile der DFE-Engine zu nutzen

`Mit den Schritten Gremlin mergeV () und mergeE () effiziente Upserts erstellen` erläutert, wie Upsert-Abfragen strukturiert werden, um die DFE-Engine so effektiv wie möglich zu nutzen.

Testen Sie den Gremlin-Code in dem Kontext, in dem Sie ihn einsetzen werden

Gremlin bietet Clients mehrere Möglichkeiten, Anfragen an den Server zu senden. Diese Übermittlungsmodi unterscheiden sich darin, wie Abfragen ausgewertet werden und wie sich Transaktionen verhalten. Diese Unterschiede können zu unerwarteten Ergebnissen führen, wenn Sie in einem Modus entwickeln und in einem anderen bereitstellen.

Skriptmodus (auf Zeichenketten basierende Übermittlung)

Im Skriptmodus sendet der Client die gesamte Abfrage als Textzeichenfolge an den Server. Der Server wertet die gesamte Zeichenfolge, einschließlich aller Terminalschritte, aus und streamt die Ergebnisse zurück an den Client. Die folgenden Tools und Methoden verwenden den Skriptmodus:

- Gremlin Console — wenn Sie den Befehl verwenden `:remote`
- Notizbuch mit Neptun-Diagramm
- HTTP mit AWS SDK oder AWS CLI
- Senden von Abfragezeichenfolgen über TinkerPop Treiber (z. B. `client.submit("g.V().count()")` in Java)

Im Skriptmodus sendet der Client eine Abfrage wie die folgende als vollständige Zeichenfolge an den Server:

```
// 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()");
```

Der Server wertet die vollständige Abfrage aus, einschließlich `.next()`. Wenn keine `.next()` Ergebnisse gefunden werden, löst der Server eine aus `NoSuchElementException` und die Transaktion wird zurückgesetzt.

Bytecode-Modus (GLV-Traversal-Objekte)

Im Bytecode-Modus erstellt der Client ein Traversal-Objekt lokal mithilfe einer Gremlin Language Variant (GLV). Der Treiber serialisiert die Durchlaufschritte als Bytecode und sendet sie an den Server. Terminalschritte wie `.toList()` oder werden auf der Client-Seite *`.next()`* ausgeführt, um

über die vom Server zurückgegebenen Ergebnisse zu iterieren. Sie können den Bytecode-Modus mit Java -, Python -, Go -, .NET - und anderen TinkerPop-compliant Treibern von Drittanbietern verwenden JavaScript, die für die verwendete Neptune-Engine-Version geeignet sind.

Im Bytecode-Modus sieht dieselbe Abfrage wie folgt aus:

```
// 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();
```

Der Treiber sendet nur die Traversalschritte () `g.V().addV().V()` als Bytecode. Der Server wertet die Durchquerung erfolgreich aus, schreibt die Transaktion fest und gibt die Ergebnismenge zurück. Der Client ruft dann `.next()` lokal auf, um aus der Ergebnismenge zu lesen. Wenn die Ergebnismenge leer ist, löst der Client eine aus`NoSuchElementException`, aber die Transaktion wurde bereits auf dem Server festgeschrieben.

Unterschiede im Transaktionsverhalten

Der entscheidende Unterschied zwischen diesen Modi besteht darin, wie sich Terminalschritte auf Transaktionen auswirken:

- Skriptmodus — Der Server wertet die Terminalschritte aus. Wenn ein Terminalschritt wie z. B. `.next()` fehlschlägt, weil die Ergebnismenge leer ist, behandelt der Server die Abfrage als fehlgeschlagen und setzt die Transaktion zurück. Der Server behält keine Mutationen bei derselben Durchquerung bei (z. B.). `addV()`
- Bytecode-Modus — Der Client wertet die Terminalschritte aus. Der Server wertet nur die Durchlaufschritte aus, schreibt die Transaktion erfolgreich fest und gibt Ergebnisse zurück. Wenn der Client dann eine leere Ergebnismenge `.next()` aufruft, `NoSuchElementException` ist das Ergebnis nur ein clientseitiger Fehler. Die Transaktion wurde bereits festgeschrieben, sodass der Server alle Mutationen beibehält.

Schritte im Terminal

In Gremlin sind terminale Schritte die Schritte, die dazu führen, dass eine Traversal zur Auswertung an Neptune übermittelt wird. Im Bytecode-Modus veranlasst der Terminalschritt den Treiber,

den Traversal zu serialisieren und zu senden. Im Skriptmodus ist der Terminalschritt Teil der Abfragezeichenfolge, die auf dem Server ausgewertet wird. Die Terminalschritte sind:

- `hasNext()`— Gibt zurück`true`, ob Ergebnisse verfügbar sind, `false` andernfalls.
- `next()`— Gibt das nächste Ergebnis zurück. Wirft, `NoSuchElementException` wenn keine Ergebnisse vorhanden sind.
- `next(n)`— Gibt die nächsten *n* Ergebnisse als Liste zurück.
- `toList()`— Gibt alle Ergebnisse als Liste zurück. Gibt eine leere Liste zurück, wenn keine Ergebnisse vorhanden sind.
- `toSet()`— Gibt alle Ergebnisse als Satz zurück. Gibt eine leere Menge zurück, wenn keine Ergebnisse vorhanden sind.
- `iterate()`— Iteriert alle Ergebnisse, ohne sie zurückzugeben. Verwenden Sie dies für Mutationen, bei denen Sie den Rückgabewert nicht benötigen.

> **Note**
>
> GLVs für einzelne Sprachen können zusätzliche terminale Schritte bereitstellen, die für ihre Implementierung spezifisch sind. Einzelheiten finden Sie auf den sprachspezifischen Seiten.

Wenn Sie Ihren Code in einem Kontext entwickeln und testen, können Probleme auftreten. Beispielsweise sendet die Gremlin-Konsole Abfragen als Skripte. Ihr Code kann sich in der Produktion anders verhalten, wenn Sie ihn in einem anderen Kontext bereitstellen, z. B. über den Java-Treiber mithilfe von Bytecode.

> **Important**
>
> Stellen Sie sicher, dass Sie den Gremlin-Code im gleichen Übertragungsmodus testen, in dem er bereitgestellt wird, um unerwartetes Transaktionsverhalten zu vermeiden.

Erstellen von effizienten Multi-Thread-Gremlin-Schreibvorgängen

Es gibt einige Richtlinien für das Multi-Thread-Laden von Daten in Neptune mithilfe von Gremlin.

Weisen Sie nach Möglichkeit jedem Thread eine Reihe von Vertices und/oder Edges zum Einfügen oder Ändern zu, die nicht kollidieren. Beispiel: Thread 1 adressiert den ID-Bereich 1–50 000,

Thread 2 adressiert den ID-Bereich 50 001–100 000 und so weiter. Dadurch verringert sich das Risiko, eine `ConcurrentModificationException` zu erhalten. Um sicher zu gehen, sollten Sie alle Schreibvorgänge mit einem `try/catch`-Block umgeben. Wenn einer der Blöcke fehlschlägt, können Sie es nach einer kurzen Verzögerung erneut versuchen. Hinweise zur Entwicklung einer Wiederholungsstrategie mit exponentiellem Backoff und Jitter finden Sie unter. the section called “Ausnahmen und Wiederholungen”

Stapelverarbeitungs-Schreibvorgänge in einer Stapelgröße zwischen 50 und 100 (Vertices oder Edges) funktionieren im Allgemeinen gut. Wenn zahlreiche Eigenschaften für jeden Vertex hinzugefügt werden, ist eine Zahl näher an 50 als an 100 möglicherweise die bessere Wahl. Es lohnt sich, etwas zu experimentieren. Für gestapelte Schreibvorgänge können Sie etwas verwenden, das in etwa wie folgt aussieht:

```
g.addV('test').property(id,'1').as('a').
  addV('test').property(id,'2').
  addE('friend').to('a').
```

Dies wird dann in jeder Stapeloperation wiederholt.

Das Verwenden von Stapeln ist deutlich effizienter als das Hinzufügen von einem Vertex oder einer Edge pro Gremlin-Umlaufzeit zum Server.

Wenn Sie einen Gremlin Language Variant (GLV)-Client verwenden, können Sie programmgesteuert einen Stapel erstellen, indem Sie zunächst ein Traversierung erstellen. Erweitern Sie ihn dann und durchlaufen Sie ihn schließlich, z. B.:

```
  t.addV('test').property(id,'1').as('a')
  t.addV('test').property(id,'2')
  t.addE('friend').to('a')
  t.iterate()
```

Es wird empfohlen, nach Möglichkeit den Gremlin Language Variant-Client zu verwenden. Sie können aber auch ähnlich mit einem Client verfahren, der Abfragen als Textzeichenfolgen übermittelt, indem Sie einen Stapel durch Verketten von Zeichenfolgen erstellen.

Wenn Sie für Abfragen eine der Gremlin-Client-Bibliotheken anstatt des allgemeinen HTTP verwenden, sollten die Threads alle über denselben Client-, Cluster- oder Verbindungspool verfügen. Möglicherweise müssen Sie die Einstellungen optimieren, um den bestmöglichen Durchsatz zu erhalten. Dazu gehören Einstellungen wie die Größe des Verbindungspools und die Anzahl der Worker-Threads, die der Gremlin-Client verwendet.

Bereinigung von Datensätzen mit der Eigenschaft „Erstellungszeit (Creation Time)“

Sie können veraltete Datensätze bereinigen, indem Sie die Erstellungszeit als Eigenschaft auf Vertices speichern und die Datensätze regelmäßig verwerfen.

Wenn Sie Daten für einen bestimmten Zeitraum speichern und dann aus dem Graphen entfernen müssen (Vertex-Lebenszeit), können Sie bei der Erstellung des Vertex eine Zeitstempeleigenschaft speichern. Anschließend können Sie regelmäßig eine `drop()`-Abfrage für alle Vertices ausgeben, die vor einem bestimmten Zeitpunkt erstellt wurden, zum Beispiel:

```
g.V().has(“timestamp”, lt(datetime('2018-10-11')))
```

Verwendung der **`datetime () -Methode für Groovy-Zeitdaten`**

Neptune bietet die `datetime`-Methode für Datums- und Zeitangaben für in der Gremlin-Groovy-Variante gesendete Abfragen. Dies umfasst die Gremlin-Konsole, Textzeichenfolgen mit der HTTP-REST-API, und jede andere Serialisierung, die Groovy verwendet.

> **Important**
>
> Dies gilt nur für Methoden, bei denen die Gremlin-Abfrage als Textzeichenfolge gesendet wird. Wenn Sie eine Gremlin Language Variant verwenden, müssen Sie die nativen Datumsklassen und Funktionen für die Sprache verwenden. Weitere Informationen finden Sie im folgenden Abschnitt, the section called “Systemeigenes Datum und systemeigene Uhrzeit”. Beginnend mit TinkerPop `3.5.2` (eingeführt in Neptune Engine Version 1.1.1.0), `datetime` ist ein integraler Bestandteil von. TinkerPop

Sie können die `datetime` -Methode verwenden, um Daten zu speichern und zu vergleichen:

```
g.V('3').property('date',datetime('2001-02-08'))
```

```
g.V().has('date',gt(datetime('2000-01-01')))
```

Verwendung der nativen Datum- und Uhrzeitangaben für GLV-Zeitdaten

Wenn Sie eine Gremlin Language Variant (GLV) verwenden, müssen Sie die nativen Datum- und Uhrzeit-Klassen und Funktionen der Programmiersprache für Gremlin-Zeitdaten verwenden.

Die offiziellen TinkerPop Bibliotheken sind alle Gremlin Language Variant-Bibliotheken.

- Go
- Java
- JavaScript
- .NET
- Python

> **Important**
>
> Diese Seite bezieht sich nur auf Gremlin Language Variant (GLV) -Bibliotheken. Wenn Sie eine Methode verwenden, bei der Sie die Gremlin-Abfrage als Textzeichenfolge senden, müssen Sie die datetime () -Funktion von Gremlin verwenden. Dazu gehören die Gremlin-Konsole, Textzeichenfolgen, die die HTTP-REST-API verwenden, oder das direkte Senden von Gremlin-Strings über die Treiber.

Go

Das Folgende ist ein Teilbeispiel in Go, das eine einzelne Eigenschaft namens 'date' für den Scheitelpunkt mit der ID '3' erstellt. Es legt den Wert auf ein Datum fest, das mit der Funktion Go Time.now () generiert wurde.

```
import ( "time" )

g.V('3').property('date', time.Now()).next();
```

Ein vollständiges Beispiel für die Verbindung zu Neptune mithilfe von Go finden Sie unter Verwenden eines Go-Clients, um eine Verbindung zu einer Neptune-DB-Instance herzustellen.

Java

Nachfolgend finden Sie ein Teilbeispiel in Java, bei dem eine einzelne Eigenschaft mit der Bezeichnung „`date`“ für den Vertex mit der ID „`3`“ erstellt wird. Diese setzt den Wert auf ein mit dem Java-Konstruktor `Date()` generiertes Datum.

```
import java.util.date

g.V('3').property('date', new Date()).next();
```

Ein vollständiges Beispiel für die Verbindung mit Neptune mithilfe von Java finden Sie unter. Herstellen einer Verbindung zu einer Neptune-DB-Instance über einen Java-Client

Node.js (JavaScript)

Im Folgenden finden Sie ein unvollständiges Beispiel JavaScript , bei dem eine einzelne Eigenschaft mit dem Namen '`date`' für den Scheitelpunkt mit der ID '' erstellt wird. `3` Es legt den Wert auf ein Datum fest, das mit dem Node.js `Date()` Konstruktor generiert wurde.

```
g.V('3').property('date', new Date()).next()
```

Ein vollständiges Beispiel für die Verbindung mit Neptune finden Sie Node.js unter. Wird verwendet Node.js , um eine Verbindung zu einer Neptune-DB-Instance herzustellen

.NET (C#)

Nachfolgend finden Sie ein Teilbeispiel in C#, bei dem eine einzelne Eigenschaft mit der Bezeichnung „`date`“ für den Vertex mit der ID „`3`“ erstellt wird. Diese setzt den Wert auf ein mit der .NET-Eigenschaft *`DateTime.UtcNow`* generiertes Datum.

```
Using System;

g.V('3').property('date', DateTime.UtcNow).next()
```

Ein vollständiges Beispiel für die Verbindung mit Neptune mithilfe von C# finden Sie unter. Herstellen einer Verbindung mit einer Neptune-DB-Instance über .NET

Python

Nachfolgend finden Sie ein Teilbeispiel in Python, bei dem eine einzelne Eigenschaft mit der Bezeichnung „`date`“ für den Vertex mit der ID „`3`“ erstellt wird. Diese setzt den Wert auf ein mit der Python-Methode `datetime.now()` generiertes Datum.

```
import datetime

g.V('3').property('date',datetime.datetime.now()).next()
```

Ein vollständiges Beispiel für die Verbindung mit Neptune mithilfe von Python finden Sie unter. Herstellen einer Verbindung mit einer Neptune-DB-Instance über Python

Bewährte Methoden für die Verwendung des Gremlin-Java-Clients mit Neptune

Befolge diese Empfehlungen, wenn du den Gremlin Java-Client mit Neptune verwendest. Diese bewährten Methoden helfen Ihnen dabei, die Leistung zu optimieren, Verbindungen effektiv zu verwalten und häufige Fallstricke bei der Arbeit mit dem Java-Treiber zu vermeiden.

Hinweise zur Konfiguration von Heartbeat-Intervallen für Neptune Serverless finden Sie unter. Heartbeat-Konfiguration für Neptune Serverless

Themen

- Wiederverwenden des Client-Objekts in mehreren Threads
- Erstellen separater Gremlin-Java-Client-Objekte für Lese- und Schreibendpunkte
- Hinzufügen mehrerer Lesereplikat-Endpunkte zu einem Gremlin-Java-Verbindungs-Pool
- Schließen des Clients zur Vermeidung des Verbindungs-Limits
- Erstellen einer neuen Verbindung nach einem Failover
- Setzen Sie maxInProcessPerConnection und maxSimultaneousUsagePerConnection auf den selben Wert.
- Senden von Abfragen an den Server als Bytecode und nicht als Zeichenfolgen
- Verwenden Sie den von einer Abfrage zurückgegebenen Oder-Iterator immer vollständig ResultSet
- Massenweises Hinzufügen von Scheitelpunkten und Edges in Stapeln
- Deaktivieren von DNS-Caching in der Java Virtual Machine
- Optionales Festlegen von Zeitüberschreitungen auf Abfrageebene
- Fehlerbehebung für java.util.concurrent.TimeoutException

Wiederverwenden des Client-Objekts in mehreren Threads

Sie können dasselbe Client (oder `GraphTraversalSource`)-Objekt in mehreren Threads wiederverwenden. Das bedeutet, dass Sie eine gemeinsame Instanz einer `org.apache.tinkerpop.gremlin.driver.Client`-Klasse in Ihrer Anwendung statt in jedem einzelnen Thread erstellen können. Das `Client`-Objekt ist threadsicher und es besteht ein erheblicher Initialisierungsaufwand.

Dies gilt auch für `GraphTraversalSource`, das intern ein `Client`-Objekt erstellt. Der folgende Code führt beispielsweise dazu, dass ein neues `Client`-Objekt instanziiert wird:

```
import static
 org.apache.tinkerpop.gremlin.process.traversal.AnonymousTraversalSource.traversal;

  /////

GraphTraversalSource traversal = traversal()
                                    .withRemote(DriverRemoteConnection.using(cluster));
```

Erstellen separater Gremlin-Java-Client-Objekte für Lese- und Schreibendpunkte

Sie können die Leistung erhöhen, indem Sie nur Schreibvorgänge auf dem Writer-Endpunkt und Lesen-Vorgänge von einem oder mehreren schreibgeschützten Endpunkten aus durchführen.

```
Client readerClient = Cluster.build("https://reader-endpoint")
          ...
          .connect()

Client writerClient = Cluster.build("https://writer-endpoint")
          ...
          .connect()
```

Hinzufügen mehrerer Lesereplikat-Endpunkte zu einem Gremlin-Java-Verbindungs-Pool

Beim Erstellen eines Gremlin-Java-`Cluster`-Objekts können Sie mithilfe der Methode `.addContactPoint()` den Kontaktpunkten des Verbindungs-Pools mehrere Read Replica-Instances hinzufügen.

```
Cluster.Builder readerBuilder = Cluster.build()
          .port(8182)
          .minConnectionPoolSize(…)
          .maxConnectionPoolSize(…)
          ……
          .addContactPoint("reader-endpoint-1")
          .addContactPoint("reader-endpoint-2")
```

Schließen des Clients zur Vermeidung des Verbindungs-Limits

Es ist wichtig, den Client zu schließen, wenn Sie damit fertig sind, um sicherzustellen, dass die WebSocket Verbindungen vom Server geschlossen und alle mit den Verbindungen verbundenen Ressourcen freigegeben werden. Dies geschieht automatisch, wenn Sie den Cluster mithilfe von `Cluster.close( )` schließen, da `client.close( )` in diesem Fall intern aufgerufen wird.

Wenn der Client nicht ordnungsgemäß geschlossen wird, beendet Neptune alle inaktiven WebSocket Verbindungen nach 20 bis 25 Minuten. Wenn Sie WebSocket Verbindungen jedoch nicht explizit schließen, wenn Sie mit ihnen fertig sind und die Anzahl der Live-Verbindungen das Limit für WebSocket gleichzeitige Verbindungen erreicht, werden weitere Verbindungen mit einem HTTP-Fehlercode abgelehnt. `429` An diesem Punkt müssen Sie die Neptune-Instance neu starten, um die Verbindungen zu schließen.

Die Empfehlung, `cluster.close()` aufzurufen, gilt nicht für Java- AWS Lambda -Funktionen. Details dazu finden Sie unter Verwaltung von Gremlin-Verbindungen WebSocket in Funktionen AWS Lambda.

Erstellen einer neuen Verbindung nach einem Failover

Im Falle eines Failovers kann der Gremlin-Treiber die Verbindung mit dem alten Writer fortsetzen, da der Cluster-DNS-Name zu einer IP-Adresse aufgelöst wurde. Wenn dies geschieht, können Sie nach dem Failover ein neues `Client`-Objekt erstellen.

Setzen Sie **`maxInProcessPerConnection`** und **`maxSimultaneousUsagePerConnection`** auf den selben Wert.

`maxInProcessPerConnection`Sowohl der als auch der `maxSimultaneousUsagePerConnection` Parameter beziehen sich auf die maximale Anzahl gleichzeitiger Abfragen, die Sie über eine einzelne WebSocket Verbindung senden können. Diese

Parameter beziehen sich intern aufeinander. Wenn ein Parameter geändert wird, ohne dass der andere Parameter geändert wird, könnte dies dazu führen, dass für den Client beim Versuch, eine Verbindung aus den Client-Verbindungs-Pool abzurufen, eine Zeitüberschreitung eintritt.

Sie sollten die Standardmindestwerte für Verarbeitung und gleichzeitige Nutzung beibehalten und `maxInProcessPerConnection` und `maxSimultaneousUsagePerConnection` auf denselben Wert festlegen.

Der Wert, auf den diese Parameter festgelegt werden, ist eine Funktion der Abfragekomplexität und des Datenmodells. Ein Anwendungsfall, in dem die Abfrage viele Daten zurückgibt, würde eine höhere Verbindungsbandbreite pro Abfrage erfordern und sollte daher niedrigere Werte für die Parameter und einen höheren Wert für `maxConnectionPoolSize` verwenden.

Wenn die Abfrage jedoch eine kleinere Menge von Daten zurückgibt, sollten `maxInProcessPerConnection` und `maxSimultaneousUsagePerConnection` auf einen höheren Wert als `maxConnectionPoolSize` festgelegt werden.

Senden von Abfragen an den Server als Bytecode und nicht als Zeichenfolgen

Die Verwendung von Bytecode statt Zeichenfolgen beim Übermitteln von Abfragen hat Vorteile:

- Frühzeitiges Abfangen einer ungültigen Abfragesyntax Bei Verwendung der Bytecode Variante können Sie eine ungültige Abfragesyntax auf der Kompilierungsstufe entdecken. Bei Verwendung der Zeichenfolgen-Variante entdecken Sie die ungültige Syntax erst, wenn die Abfrage an den Server übermittelt wurde und ein Fehler zurückgegeben wird.
- Vermeiden Sie Leistungseinbußen aufgrund von Zeichenketten: Jede auf Zeichenfolgen basierende Abfrage, unabhängig davon, ob Sie HTTP verwenden WebSockets , führt zu einem getrennten Scheitelpunkt, was bedeutet, dass das Vertex-Objekt aus der ID, dem Label und allen mit dem Scheitelpunkt verknüpften Eigenschaften besteht (siehe Eigenschaften von Elementen).

  Dies kann in Fällen, in denen die Eigenschaften nicht erforderlich sind, zu unnötigen Berechnungen auf dem Server führen. Wenn der Kunde beispielsweise den Eckpunkt mit der ID „hakuna#1“ über die Abfrage `g.V("hakuna#1")` abrufen möchte: Wenn die Abfrage als zeichenfolgebasierte Übermittlung gesendet wird, würde der Server Zeit für den Abruf von ID, Beschriftung und aller Eigenschaften für diesen Eckpunkt aufwenden. Wenn die Abfrage als Bytecode-Übermittlung gesendet wird, wendet der Server lediglich Zeit für den Abruf von ID und Beschriftung des Eckpunkts auf.

Sie sollten also anstelle der Übermittlung einer Abfrage wie der folgenden:

```
 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();
 }
```

Die Abfrage als Bytecode-Übermittlung wie folgt senden:

```
 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();
 }
```

Verwenden Sie den von einer Abfrage zurückgegebenen Oder-Iterator immer vollständig ResultSet

Das Client-Objekt sollte den `ResultSet` (im Fall einer zeichenfolgebasierten Übermittlung) oder den von `GraphTraversal` zurückgegebenen Iterator stets vollständig nutzen. Wenn die

Abfrageergebnisse nicht vollständig genutzt werden, bewahrt der Server sie auf und wartet darauf, dass der Client sie vollständig nutzt.

Wenn Ihre Anwendung nur eine Teilmenge von Ergebnissen benötigt, können Sie in Ihrer Abfrage einen `limit(X)`-Schritt verwenden, um die Anzahl der vom Server generierten Ergebnisse einzuschränken.

Massenweises Hinzufügen von Scheitelpunkten und Edges in Stapeln

Jede Abfrage an die Neptune-DB wird als einzelne Transaktion ausgeführt, es sei denn, Sie verwenden eine Sitzung. Wenn Sie daher viele Daten mithilfe von Gremlin-Abfragen einfügen müssen, wird die Leistung verbessert, wenn Sie diese Abfragen in Form von Batches aus 50 bis 100 Abfragen ausführen. Dies reduziert die Anzahl der für die Last erstellten Transaktionen.

Wenn Sie der Datenbank beispielsweise 5 Eckpunkte hinzufügen, würde dies wie folgt aussehen:

```
// 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();
```

In ähnlicher Weise können Sie Kanten stapelweise hinzufügen mit. `addE` Verwenden Sie diese `V()` Option, um bestehende Scheitelpunkte als Quelle und Ziel jeder Kante zu referenzieren:

```
// 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();
```

Sie können die Erstellung von Scheitelpunkten und Kanten auch in einem einzigen Stapel kombinieren. Wird verwendet`as()`, um neu erstellte Scheitelpunkte zu beschriften, sodass Sie sie referenzieren können, wenn Sie Kanten in derselben Traversierung hinzufügen:

```
// 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();
```

Deaktivieren von DNS-Caching in der Java Virtual Machine

In einer Umgebung, in der Sie Anfragen auf mehrere Read Replicas verteilen möchten, müssen Sie das DNS-Caching in der Java Virtual Machine (JVM) deaktivieren und bei der Erstellung des Cluster-Objekts den Reader-Endpunkt von Neptune angeben. Das Deaktivieren des JVM-DNS-Caching stellt sicher, dass DNS für jede neue Verbindung erneut aufgelöst wird, sodass die Anforderungen über alle Read Replicas verteilt werden. Sie können dies im Initialisierungscode Ihrer Anwendung mit der folgenden Zeile tun:

```
java.security.Security.setProperty("networkaddress.cache.ttl", "0");
```

Eine vollständigere und robustere Lösung für den Lastenausgleich bietet jedoch der Amazon Gremlin Java-Client Code on. GitHub Der Amazon-Java-Gremlin-Client kennt Ihre Cluster-Topologie und verteilt Verbindungen und Anfragen in fairer Weise auf eine Reihe von Instances in Ihrem Neptune-Cluster. In diesem Blog-Beitrag finden Sie ein Beispiel für eine Java-Lambda-Funktion, die diesen Client verwendet.

Optionales Festlegen von Zeitüberschreitungen auf Abfrageebene

Neptune bietet Ihnen die Möglichkeit, mithilfe der Parametergruppenoption `neptune_query_timeout` (siehe Parameters) ein Timeout für Ihre Abfragen einzurichten. Sie können das globale Timeout auch mit folgendem Code überschreiben:

> **Note**
>
> Die Einstellungen für das Abfrage-Timeout gelten nur für die Auswertung von Abfragen. Bytecode-basierte Transaktionssteuerungsoperationen wie Abfrage-Timeouts `tx().commit()` und `tx().rollback()` unterliegen keinen Abfrage-Timeouts. Weitere Informationen finden Sie unter Timeout-Verhalten bei Bytecode-Commit und -Rollback.

```
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();
 }
```

Für die Übermittlung zeichenfolgebasierter Abfragen würde der Code wie folgt aussehen:

```
RequestOptions options = RequestOptions.build().timeout(500).create();
List<Result> result = client.submit("g.V()", options).all().get();
```

Wenn eine Abfrage das Timeout pro Abfrage überschreitet, beendet Neptune sie. Ob Sie eine Abfrage, bei der das Timeout abgelaufen ist, erneut versuchen müssen, hängt von der Art des Fehlers und Ihrer Arbeitslast ab. Anleitungen finden Sie unter the section called “Ausnahmen und Wiederholungen”.

> **Note**
>
> Es können unerwartete Kosten entstehen, wenn Sie den Wert für das Abfrage-Timeout zu hoch festlegen, insbesondere bei einer Serverless-Instance. Ohne eine angemessene Timeout-Einstellung läuft Ihre Abfrage möglicherweise viel länger als erwartet, wodurch Kosten entstehen können, die Sie nie erwartet haben. Dies gilt insbesondere für eine Serverless-Instance, die während der Ausführung der Abfrage auf einen großen, teuren Instance-Typ hochskaliert werden könnte.
>
> Sie können unerwartete Ausgaben dieser Art vermeiden, indem Sie einen Wert für das Abfrage-Timeout verwenden, der der von Ihnen erwarteten Laufzeit entspricht und nur bei einer ungewöhnlich langen Ausführung zu einer Zeitüberschreitung führt.
>
> Ab der Neptune-Engine-Version 1.3.2.0 unterstützt Neptune einen neuen neptune_lab_mode-Parameter als. `StrictTimeoutValidation` Wenn dieser Parameter den Wert von hat, darf ein Timeout-Wert pro Abfrage`Enabled`, der als Anforderungsoption oder als Abfragehinweis angegeben wurde, den global in der Parametergruppe festgelegten Wert nicht überschreiten. In einem solchen Fall wirft Neptune. `InvalidParameterException`

Diese Einstellung kann in einer Antwort auf den Endpunkt '/status' bestätigt werden, wenn der Wert lautet. `Disabled` In der Engine-Version `1.3.2.0` ist `Disabled` der Standardwert dieses Parameters. Ab der Engine-Version `1.4.0.0` ist der `StrictTimeoutValidation` Parameter `Enabled` standardmäßig.
Weitere Informationen darüber, wie die Timeout-Priorität bestimmt wird, wenn mehrere Timeout-Einstellungen konfiguriert sind, finden Sie in der Dokumentation zum Parameter neptune_query_timeout.

Fehlerbehebung für `java.util.concurrent.TimeoutException`

Der Gremlin-Java-Client löst eine aus, `java.util.concurrent.TimeoutException` wenn bei einer Gremlin-Anfrage das Timeout am Client selbst überschritten wird, während er darauf wartet, dass ein Slot in einer der Verbindungen verfügbar wird. WebSocket Diese Timeout-Dauer wird durch den clientseitig konfigurierbaren Parameter `maxWaitForConnection` gesteuert.

> **Note**
>
> Da Anfragen, bei denen das Timeout auf dem Client auftritt, nie an den Server gesendet werden, spiegeln sie sich in keiner der auf dem Server erfassten Metriken wider, wie z. B. `GremlinRequestsPerSec`.

Diese Art von Timeout wird im Allgemeinen auf zwei Arten verursacht:

- Der Server hat tatsächlich die maximale Kapazität erreicht. Wenn dies der Fall ist, füllt sich die Warteschlange auf dem Server, was Sie anhand der Metrik erkennen können. MainRequestQueuePendingRequests CloudWatch Die Anzahl der parallelen Abfragen, die der Server verarbeiten kann, hängt von seiner Instance-Größe ab.

  Wenn die `MainRequestQueuePendingRequests`-Metrik keine Anhäufung ausstehender Anfragen auf dem Server anzeigt, kann der Server mehr Anfragen bearbeiten und das Timeout-Ereignis wird durch die clientseitige Drosselung verursacht.
- Drosselung von Anfragen durch den Client. Dies kann im Allgemeinen behoben werden, indem die Client-Konfigurationseinstellungen geändert werden.

  Die maximale Anzahl von parallelen Anforderungen, die der Client senden kann, kann grob wie folgt geschätzt werden:

```
maxParallelQueries = maxConnectionPoolSize * Max( maxSimultaneousUsagePerConnection,
 maxInProcessPerConnection )
```

Wenn mehr als `maxParallelQueries` an den Client gesendet wird, treten `java.util.concurrent.TimeoutException`-Ausnahmen auf. Sie können dies generell auf verschiedene Weise beheben:

- Erhöhen der Dauer des Verbindungs-Timeouts. Wenn die Latenz für Ihre Anwendung nicht entscheidend ist, erhöhen Sie die `maxWaitForConnection`-Einstellung des Clients. Der Client wartet dann länger, bis das Timeout eintritt, was wiederum die Latenz erhöhen kann.
- Erhöhen der maximalen Anzahl an Anfragen pro Verbindung. Dadurch können mehr Anfragen über dieselbe WebSocket Verbindung gesendet werden. Erhöhen Sie dazu die Einstellungen `maxSimultaneousUsagePerConnection` und `maxInProcessPerConnection` des Clients. Diese Einstellungen sollten im Allgemeinen den gleichen Wert haben.
- Erhöhen Sie die Anzahl der Verbindungen im Verbindungspool. Erhöhen Sie dazu die `maxConnectionPoolSize`-Einstellung des Clients. Die Kosten sind ein erhöhter Ressourcenverbrauch, da jede Verbindung Speicher und einen Betriebssystem-Dateideskriptor verwendet und bei der Initialisierung SSL und WebSocket Handshake erforderlich sind.

Bewährte Methoden für Neptune mit openCypher und Bolt

Befolgen Sie diese bewährten Methoden bei der Verwendung der openCypher-Abfragesprache mit Neptune. Informationen über die Verwendung von openCypher in Neptune finden Sie unter Zugriff auf das Neptun-Diagramm mit openCypher.

Themen

- Erstellen einer neuen Verbindung nach einem Failover
- Verbindungsverwaltung für langlebige Anwendungen
- Verbindungsmanagement für AWS Lambda
- Bevorzugung direktionaler gegenüber bidirektionalen Edges in Abfragen
- Neptune unterstützt nicht mehrere gleichzeitige Abfragen in einer Transaktion
- Schließen Sie anschließend Treiberobjekte
- Verwenden expliziter Transaktionsmodi zum Lesen und Schreiben
- Wiederholungslogik für Ausnahmen

- Legen Sie mithilfe einer einzigen SET-Klausel mehrere Eigenschaften gleichzeitig fest
- Parametrisierte Abfragen verwenden
- Verwenden Sie in der UNWIND-Klausel abgeflachte Maps anstelle von verschachtelten Maps
- Platzieren Sie restriktivere Knoten in VLP-Ausdrücken (Variable-Length Path) auf der linken Seite
- Vermeiden Sie redundante Prüfungen von Knotenbezeichnungen, indem Sie detaillierte Beziehungsnamen verwenden
- Geben Sie nach Möglichkeit Kantenbeschriftungen an
- Vermeiden Sie nach Möglichkeit die WITH-Klausel
- Platzieren Sie restriktive Filter so früh wie möglich in der Abfrage
- Prüfen Sie explizit, ob Eigenschaften vorhanden sind
- Verwenden Sie keinen benannten Pfad (es sei denn, er ist erforderlich)
- Vermeiden Sie COLLECT (DISTINCT ())
- Ziehen Sie beim Abrufen aller Eigenschaftswerte die Eigenschaftenfunktion der Suche nach einzelnen Eigenschaften vor
- Führen Sie statische Berechnungen außerhalb der Abfrage durch
- Batch-Eingaben mit UNWIND anstelle von Einzelanweisungen
- Verwenden Sie lieber „Benutzerdefiniert“ IDs für Knoten/Beziehungen
- Vermeiden Sie es, ~id-Berechnungen in der Abfrage durchzuführen
- Aktualisieren/Zusammenführen mehrerer Knoten

Erstellen einer neuen Verbindung nach einem Failover

Im Falle eines Failovers kann der Bolt-Treiber weiterhin eine Verbindung zur alten Writer-Instance und nicht zur neuen aktiven Instance herstellen, da der DNS-Name zu einer bestimmten IP-Adresse aufgelöst wurde.

Um dies zu verhindern, schließen Sie das `Driver`-Objekt und verbinden Sie es nach einem Failover erneut.

Verbindungsverwaltung für langlebige Anwendungen

Wenn Sie langlebige Anwendungen erstellen, z. B. solche, die in Containern oder auf Amazon-EC2-Instances ausgeführt werden, müssen Sie ein `Driver`-Objekt einmal instanziieren und dieses

Objekt dann für die gesamte Lebensdauer der Anwendung wiederverwenden. Das `Driver`-Objekt ist threadsicher und es besteht ein erheblicher Initialisierungsaufwand.

Verbindungsmanagement für AWS Lambda

Bolt-Treiber werden aufgrund ihres Verbindungsaufwands und der Verwaltungsanforderungen nicht für den Einsatz innerhalb von AWS Lambda Funktionen empfohlen. Verwenden Sie stattdessen den HTTPS-Endpunkt.

Bevorzugung direktionaler gegenüber bidirektionalen Edges in Abfragen

Wenn Neptune Abfrageoptimierungen durchführt, erschweren bidirektionale Edges die Erstellung optimaler Abfragepläne. Bei suboptimalen Plänen muss die Engine unnötige Arbeit verrichten, was zu einer schlechteren Leistung führt.

Verwenden Sie daher nach Möglichkeit direktionale anstelle von bidirektionalen Edges. Verwenden Sie beispielsweise:

```
MATCH p=(:airport {code: 'ANC'})-[:route]->(d) RETURN p)
```

anstelle von:

```
MATCH p=(:airport {code: 'ANC'})-[:route]-(d) RETURN p)
```

Die meisten Datenmodelle müssen Edges nicht wirklich in beide Richtungen durchqueren, so dass Abfragen durch die Umstellung auf die Verwendung direktionaler Edges zu erheblichen Leistungsverbesserungen führen können.

Wenn Ihr Datenmodell das Durchqueren bidirektionaler Edges erfordert, machen Sie den ersten Knoten (linke Seite) im `MATCH`-Muster zum Knoten mit der restriktivsten Filterung.

Nehmen wir das Beispiel „Finde alle `routes` zum und vom `ANC`-Flughafen“. Schreiben Sie diese Abfrage wie folgt so, dass sie am `ANC`-Flughafen beginnt:

```
MATCH p=(src:airport {code: 'ANC'})-[:route]-(d) RETURN p
```

Die Engine kann den geringsten Arbeitsaufwand zur Erfüllung der Abfrage wählen, da der Knoten mit den meisten Einschränkungen als erster Knoten (linke Seite) im Muster platziert wird. Die Engine kann dann die Abfrage optimieren.

Das ist weitaus besser, als den ANC-Flughafen am Ende des Musters wie folgt zu filtern:

```
MATCH p=(d)-[:route]-(src:airport {code: 'ANC'}) RETURN p
```

Wenn der Knoten mit den meisten Einschränkungen nicht an erster Stelle im Muster steht, muss die Engine zusätzliche Arbeit leisten, da sie die Abfrage nicht optimieren kann und zusätzliche Suchvorgänge durchführen muss, um zu den Ergebnissen zu gelangen.

Neptune unterstützt nicht mehrere gleichzeitige Abfragen in einer Transaktion

Obwohl der Bolt-Treiber selbst gleichzeitige Abfragen in einer Transaktion zulässt, unterstützt Neptune nicht mehrere Abfragen in einer gleichzeitig laufenden Transaktion. Stattdessen verlangt Neptune, dass mehrere Abfragen in einer Transaktion nacheinander ausgeführt werden und dass die Ergebnisse jeder Abfrage vollständig verarbeitet werden, bevor die nächste Abfrage initiiert wird.

Das folgende Beispiel zeigt, wie Bolt verwendet wird, um mehrere Abfragen nacheinander in einer Transaktion auszuführen, so dass die Ergebnisse jeder Abfrage vollständig verbraucht werden, bevor die nächste beginnt:

```
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());
    }
  }
}
```

Schließen Sie anschließend Treiberobjekte

Es ist wichtig, den Client nach Abschluss des Vorgangs zu schließen, damit die Bolt-Verbindungen vom Server geschlossen und alle Ressourcen im Zusammenhang mit den Verbindungen freigegeben werden. Dies geschieht automatisch, wenn Sie den Treiber mit `driver.close()` schließen.

Wenn der Treiber nicht korrekt geschlossen wird, beendet Neptune alle inaktiven Bolt-Verbindungen nach 20 Minuten oder nach 10 Tagen, wenn Sie die IAM-Authentifizierung verwenden.

Neptune unterstützt nicht mehr als 1 000 gleichzeitige Bolt-Verbindungen. Wenn Sie Verbindungen nicht explizit schließen, wenn Sie mit ihnen fertig sind, und die Anzahl der Live-Verbindungen diese Grenze von 1 000 erreicht, schlagen alle neuen Verbindungsversuche fehl.

Verwenden expliziter Transaktionsmodi zum Lesen und Schreiben

Wenn Sie Transaktionen mit Neptune und dem Bolt-Treiber verwenden, ist es am besten, den Zugriffsmodus für Lese- und Schreibtransaktionen explizit auf die korrekten Einstellungen einzustellen.

Nur-Lese-Transaktionen

Wenn Sie bei schreibgeschützten Transaktionen beim Erstellen der Sitzung nicht die entsprechende Konfiguration für den Zugriffsmodus angeben, wird die Standardisolationsstufe verwendet, d. h. die Isolation von Mutationsabfragen. Daher ist es für schreibgeschützte Transaktionen wichtig, den Zugriffsmodus explizit auf `read` festzulegen.

Beispiel für eine Lesetransaktion mit automatischem Commit:

```
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()
}
```

Beispiel für eine Lese-Transaktion:

```
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)
    }
  }
);
```

In beiden Fällen wird die SNAPSHOT-Isolation mithilfe der Neptune-Nur-Lese-Transaktionssemantik erreicht.

Da Lesereplikate nur schreibgeschützte Abfragen akzeptieren, wird jede Abfrage, die an ein Lesereplikat gesendet wird, mit SNAPSHOT-Isolationssemantik ausgeführt.

Für schreibgeschützte Transaktionen gibt es keine „Dirty Reads“ oder nicht wiederholbaren Lesevorgänge.

Mutationstransaktionen

Für Mutationsabfragen gibt es drei verschiedene Mechanismen, um eine Schreibtransaktion zu erstellen, die nachfolgend erläutert werden:

Beispiel für eine implizite Schreibtransaktion:

```
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)
    }
  }
);
```

Beispiel für eine Auto-Commit-Schreibtransaktion:

```
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()
}
```

Beispiel für eine explizite Schreibtransaktion:

```
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();
```

Isolationsstufen für Schreibtransaktionen

- Lesevorgänge im Rahmen von Mutationsabfragen werden unter `READ COMMITTED`-Transaktionsisolation ausgeführt.
- Für Lesevorgänge im Rahmen von Mutationsabfragen gibt es keine „Dirty Reads“.
- Datensätze und Datensatzbereiche werden beim Lesen in einer Mutationsabfrage gesperrt.
- Wenn ein Bereich des Indexes von einer Mutationstransaktion gelesen wurde, gibt es eine starke Garantie dafür, dass dieser Bereich bis zum Ende der Lesetransaktion nicht durch gleichzeitige Transaktionen geändert wird.

Mutationsabfragen sind nicht Thread-sicher.

Informationen zu Konflikten finden Sie unter Konfliktlösung mithilfe von Timeouts Lock-Wait.

Mutationsabfragen werden im Fehlerfall nicht automatisch wiederholt.

Wiederholungslogik für Ausnahmen

Nicht alle Ausnahmen sollten erneut versucht werden. Beispielsweise sollten Syntaxfehler, Authentifizierungsfehler oder ungültige Abfragen keinen erneuten Versuch auslösen. Verwenden Sie für Ausnahmen, die einen erneuten Versuch zulassen, eine exponentielle Backoff-Strategie mit Jitter, um die Wartezeit zwischen Wiederholungsversuchen schrittweise zu verlängern und synchronisierte Wiederholungsstürme zu vermeiden. Legen Sie eine angemessene maximale Anzahl von Wiederholungen fest, um Endlosschleifen und eine Erschöpfung der Ressourcen zu verhindern. Eine ausführliche Anleitung zur Entwicklung einer praktischen Wiederholungsstrategie finden Sie unter. the section called “Ausnahmen und Wiederholungen”

Im Folgenden finden Sie ein Beispiel für ein exponentielles Backoff- und Wiederholungsmuster zur Behandlung von Fehlern: `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);
  }
}
```

Legen Sie mithilfe einer einzigen SET-Klausel mehrere Eigenschaften gleichzeitig fest

Anstatt mehrere SET-Klauseln zu verwenden, um einzelne Eigenschaften festzulegen, verwenden Sie eine Map, um mehrere Eigenschaften für eine Entität gleichzeitig festzulegen.

Sie können Folgendes verwenden:

```
MATCH (n:SomeLabel {`~id`: 'id1'})
SET n += {property1 : 'value1',
property2 : 'value2',
property3 : 'value3'}
```

Anstatt:

```
MATCH (n:SomeLabel {`~id`: 'id1'})
SET n.property1 = 'value1'
SET n.property2 = 'value2'
SET n.property3 = 'value3'
```

Die SET-Klausel akzeptiert entweder eine einzelne Eigenschaft oder eine Map. Wenn mehrere Eigenschaften einer einzelnen Entität aktualisiert werden, ermöglicht die Verwendung einer einzigen

SET-Klausel mit einer Map, dass die Aktualisierungen in einem einzigen Vorgang statt in mehreren Vorgängen durchgeführt werden können, was effizienter ausgeführt werden kann.

Verwenden Sie die SET-Klausel, um mehrere Eigenschaften gleichzeitig zu entfernen

Wenn Sie die OpenCypher-Sprache verwenden, wird REMOVE verwendet, um Eigenschaften aus einer Entität zu entfernen. In Neptune erfordert jede Eigenschaft, die entfernt wird, eine separate Operation, wodurch die Abfragelatenz erhöht wird. Sie können SET stattdessen mit einer Map verwenden, um alle Eigenschaftswerte auf festzulegen`null`, was in Neptune dem Entfernen von Eigenschaften entspricht. Neptune wird eine höhere Leistung haben, wenn mehrere Eigenschaften auf einer einzelnen Entität entfernt werden müssen.

Verwenden:

```
WITH {prop1: null, prop2: null, prop3: null} as propertiesToRemove
MATCH (n)
SET n += propertiesToRemove
```

Anstatt:

```
MATCH (n)
REMOVE n.prop1, n.prop2, n.prop3
```

Parametrisierte Abfragen verwenden

Es wird empfohlen, bei Abfragen mit OpenCypher immer parametrisierte Abfragen zu verwenden. Die Abfrage-Engine kann wiederholte parametrisierte Abfragen für Funktionen wie den Abfrageplan-Cache nutzen, wobei durch wiederholtes Aufrufen derselben parametrisierten Struktur mit unterschiedlichen Parametern die zwischengespeicherten Pläne genutzt werden können.
Der für parametrisierte Abfragen generierte Abfrageplan wird erst zwischengespeichert und wiederverwendet, wenn er innerhalb von 100 ms abgeschlossen ist und die Parametertypen entweder NUMBER, BOOLEAN oder STRING lauten.

Verwenden:

```
MATCH (n:foo) WHERE id(n) = $id RETURN n
```

Mit Parametern:

```
parameters={"id": "first"}
parameters={"id": "second"}
parameters={"id": "third"}
```

Anstatt:

```
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
```

Verwenden Sie in der UNWIND-Klausel abgeflachte Maps anstelle von verschachtelten Maps

Eine tief verschachtelte Struktur kann die Fähigkeit der Abfrage-Engine einschränken, einen optimalen Abfrageplan zu generieren. Um dieses Problem teilweise zu beheben, werden anhand der folgenden definierten Muster optimale Pläne für die folgenden Szenarien erstellt:

- Szenario 1: UNWIND mit einer Liste von Chiffrierliteralen, die NUMBER, STRING und BOOLEAN umfasst.
- Szenario 2: UNWIND mit einer Liste von abgeflachten Zuordnungen, die nur Chiffrierliterale (NUMBER, STRING, BOOLEAN) als Werte enthält.

Verwenden Sie beim Schreiben einer Abfrage, die eine UNWIND-Klausel enthält, die obige Empfehlung, um die Leistung zu verbessern.

Beispiel für Szenario 1:

```
UNWIND $ids as x
MATCH(t:ticket {`~id`: x})
```

Mit Parametern:

```
parameters={
  "ids": [1, 2, 3]
}
```

Ein Beispiel für Szenario 2 ist die Generierung einer Liste von Knoten, die ERSTELLT oder ZUSAMMENGEFÜHRT werden sollen. Anstatt mehrere Anweisungen auszugeben, verwenden

Sie das folgende Muster, um die Eigenschaften als eine Gruppe von abgeflachten Zuordnungen zu definieren:

```
UNWIND $props as p
CREATE(t:ticket {title: p.title, severity:p.severity})
```

Mit Parametern:

```
parameters={
  "props": [
    {"title": "food poisoning", "severity": "2"},
    {"title": "Simone is in office", "severity": "3"}
  ]
}
```

Anstelle von verschachtelten Knotenobjekten wie:

```
UNWIND $nodes as n
CREATE(t:ticket n.properties)
```

Mit Parametern:

```
parameters={
  "nodes": [
    {"id": "ticket1", "properties": {"title": "food poisoning", "severity": "2"}},
    {"id": "ticket2", "properties": {"title": "Simone is in office", "severity": "3"}}
  ]
}
```

Platzieren Sie restriktivere Knoten in VLP-Ausdrücken (Variable-Length Path) auf der linken Seite

Bei VLP-Abfragen (Variable-Length Path) optimiert die Abfrage-Engine die Auswertung, indem sie entscheidet, ob die Durchquerung auf der linken oder rechten Seite des Ausdrucks gestartet wird. Die Entscheidung basiert auf der Kardinalität der Muster auf der linken und rechten Seite. Die Kardinalität ist die Anzahl der Knoten, die dem angegebenen Muster entsprechen.

- Wenn das richtige Muster eine Kardinalität von eins hat, ist die rechte Seite der Ausgangspunkt.

- Wenn die linke und die rechte Seite die Kardinalität eins haben, wird die Expansion auf beiden Seiten geprüft und beginnt auf der Seite mit der kleineren Ausdehnung. Expansion ist die Anzahl der ausgehenden oder eingehenden Kanten für den Knoten auf der linken und den Knoten auf der rechten Seite des VLP-Ausdrucks. Dieser Teil der Optimierung wird nur verwendet, wenn die VLP-Beziehung unidirektional ist und der Beziehungstyp angegeben ist.
- Andernfalls wird die linke Seite der Ausgangspunkt sein.

Bei einer Kette von VLP-Ausdrücken kann diese Optimierung nur auf den ersten Ausdruck angewendet werden. Die anderen VLPs werden beginnend mit der linken Seite ausgewertet. Als Beispiel sei die Kardinalität von (a), (b) eins und die Kardinalität von (c) größer als eins.

- `(a)-[*1..]->(c)`: Die Bewertung beginnt mit (a).
- `(c)-[*1..]->(a)`: Die Bewertung beginnt mit (a).
- `(a)-[*1..]-(c)`: Die Bewertung beginnt mit (a).
- `(c)-[*1..]-(a)`: Die Bewertung beginnt mit (a).

Nun seien die eingehenden Kanten von (a) zwei und die ausgehenden Kanten von (a) drei, die eingehenden Kanten von (b) vier und die ausgehenden Kanten von (b) fünf.

- `(a)-[*1..]->(b)`: Die Auswertung beginnt mit (a), da die Ausgangskanten von (a) kleiner sind als die eingehenden Kanten von (b).
- `(a)<-[*1..]-(b)`: Die Auswertung beginnt mit (a), da die eingehenden Kanten von (a) kleiner sind als die ausgehenden Kanten von (b).

Als allgemeine Regel gilt: Platzieren Sie das restriktivere Muster auf der linken Seite eines VLP-Ausdrucks.

Vermeiden Sie redundante Prüfungen von Knotenbezeichnungen, indem Sie detaillierte Beziehungsnamen verwenden

Bei der Leistungsoptimierung ermöglicht die Verwendung von Beziehungsbezeichnungen, die ausschließlich für Knotenmuster gelten, den Labelfilter für Knoten zu entfernen. Stellen Sie sich ein Graphmodell vor, bei dem die Beziehung `likes` nur verwendet wird, um eine Beziehung zwischen zwei `person` Knoten zu definieren. Wir könnten die folgende Abfrage schreiben, um dieses Muster zu finden:

```
MATCH (n:person)-[:likes]->(m:person)
RETURN n, m
```

Die `person` Labelprüfung für n und m ist überflüssig, da wir die Beziehung so definiert haben, dass sie nur erscheint, wenn beide vom Typ sind`person`. Um die Leistung zu optimieren, können wir die Abfrage wie folgt schreiben:

```
MATCH (n)-[:likes]->(m)
RETURN n, m
```

Dieses Muster kann auch angewendet werden, wenn Eigenschaften nur für ein einzelnes Knotenlabel gelten. Gehen Sie davon aus, dass nur `person` Knoten über die Eigenschaft verfügen. `email` Daher `person` ist es überflüssig, zu überprüfen, ob die Knotenbezeichnungen übereinstimmen. Diese Abfrage schreiben als:

```
MATCH (n:person)
WHERE n.email = 'xxx@gmail.com'
RETURN n
```

Ist weniger effizient als das Schreiben dieser Abfrage als:

```
MATCH (n)
WHERE n.email = 'xxx@gmail.com'
RETURN n
```

Sie sollten dieses Muster nur verwenden, wenn Leistung wichtig ist und Sie in Ihrem Modellierungsprozess überprüft haben, um sicherzustellen, dass diese Kantenbeschriftungen nicht für Muster wiederverwendet werden, an denen andere Knotenbeschriftungen beteiligt sind. Wenn Sie später eine `email` Eigenschaft für eine andere Knotenbeschriftung hinzufügen`company`, z. B., dann unterscheiden sich die Ergebnisse zwischen diesen beiden Versionen der Abfrage.

Geben Sie nach Möglichkeit Kantenbeschriftungen an

Es wird empfohlen, bei der Angabe einer Kante in einem Muster nach Möglichkeit eine Kantenbeschriftung anzugeben. Stellen Sie sich die folgende Beispielabfrage vor, die verwendet wird, um alle in einer Stadt lebenden Personen mit allen Personen zu verknüpfen, die diese Stadt besucht haben.

```
MATCH (person)-->(city {country: "US"})-->(anotherPerson)
```

```
RETURN person, anotherPerson
```

Wenn Ihr Grafikmodell mithilfe mehrerer Kantenbeschriftungen Personen mit anderen Knoten als Städten verbindet, muss Neptune zusätzliche Pfade auswerten, die später verworfen werden, ohne die Endbeschriftung anzugeben. Da in der obigen Abfrage keine Kantenbeschriftung angegeben wurde, erledigt die Engine zunächst mehr Arbeit und filtert dann Werte heraus, um das richtige Ergebnis zu erhalten. Eine bessere Version der obigen Abfrage könnte sein:

```
MATCH (person)-[:livesIn]->(city {country: "US"})-[:visitedBy]->(anotherPerson)
RETURN person, anotherPerson
```

Dies hilft nicht nur bei der Auswertung, sondern ermöglicht es dem Abfrageplaner auch, bessere Pläne zu erstellen. Sie könnten diese bewährte Methode sogar mit redundanten Prüfungen von Knotenbeschriftungen kombinieren, um die Prüfung der Ortsbeschriftung zu entfernen und die Abfrage wie folgt zu schreiben:

```
MATCH (person)-[:livesIn]->({country: "US"})-[:visitedBy]->(anotherPerson)
RETURN person, anotherPerson
```

Vermeiden Sie nach Möglichkeit die WITH-Klausel

Die WITH-Klausel in OpenCypher dient als Grenze, an der alles vor der Ausführung ausgeführt wird. Anschließend werden die resultierenden Werte an die verbleibenden Teile der Abfrage übergeben. Die WITH-Klausel wird benötigt, wenn Sie eine Zwischenaggregation benötigen oder die Anzahl der Ergebnisse einschränken möchten. Abgesehen davon sollten Sie jedoch versuchen, die WITH-Klausel zu vermeiden. Die allgemeine Anleitung besteht darin, diese einfachen WITH-Klauseln (ohne Aggregation, Order by oder Limit) zu entfernen, damit der Abfrageplaner die gesamte Abfrage bearbeiten kann, um einen global optimalen Plan zu erstellen. Nehmen wir als Beispiel an, Sie haben eine Abfrage geschrieben, um alle Personen zurückzugeben, die in folgenden Ländern leben: `India`

```
MATCH (person)-[:lives_in]->(city)
WITH person, city
MATCH (city)-[:part_of]->(country {name: 'India'})
RETURN collect(person) AS result
```

In der obigen Version schränkt die WITH-Klausel die Platzierung des Musters `(city)-[:part_of]->(country {name: 'India'})` (was restriktiver ist) zuvor `(person)-[:lives_in]->(city)` ein. Das macht den Plan suboptimal. Eine Optimierung dieser Abfrage

bestünde darin, die WITH-Klausel zu entfernen und den Planer den besten Plan berechnen zu lassen.

```
MATCH (person)-[:lives_in]->(city)
MATCH (city)-[:part_of]->(country {name: 'India'})
RETURN collect(person) AS result
```

Platzieren Sie restriktive Filter so früh wie möglich in der Abfrage

In allen Szenarien trägt die frühzeitige Platzierung von Filtern in der Abfrage dazu bei, die Anzahl der Zwischenlösungen zu reduzieren, die ein Abfrageplan berücksichtigen muss. Das bedeutet, dass weniger Speicher und weniger Rechenressourcen für die Ausführung der Abfrage benötigt werden.

Das folgende Beispiel hilft Ihnen, diese Auswirkungen zu verstehen. Angenommen, Sie schreiben eine Abfrage, um alle Personen zurückzugeben, die in`India`. Eine Version der Abfrage könnte sein:

```
MATCH (n)-[:lives_in]->(city)-[:part_of]->(country)
WITH country, collect(n.firstName + " "  + n.lastName) AS result
WHERE country.name = 'India'
RETURN result
```

Die obige Version der Abfrage ist nicht der optimale Weg, um diesen Anwendungsfall zu erreichen. Der Filter `country.name = 'India'` erscheint später im Abfragemuster. Er erfasst zunächst alle Personen und ihren Wohnort und gruppiert sie nach Ländern. Anschließend wird nur nach der Gruppe für gefiltert`country.name = India`. Die optimale Methode, nur nach Personen abzufragen, die in diesem Land leben, `India` und dann die Sammelaggregation durchzuführen.

```
MATCH (n)-[:lives_in]->(city)-[:part_of]->(country)
WHERE country.name = 'India'
RETURN collect(n.firstName + " "  + n.lastName) AS result
```

Eine allgemeine Regel besteht darin, einen Filter so bald wie möglich nach der Einführung der Variablen zu platzieren.

Prüfen Sie explizit, ob Eigenschaften vorhanden sind

Basierend auf der OpenCypher-Semantik entspricht der Zugriff auf eine Eigenschaft einem optionalen Join und muss alle Zeilen beibehalten, auch wenn die Eigenschaft nicht existiert. Wenn Sie anhand

Ihres Diagrammschemas wissen, dass eine bestimmte Eigenschaft für diese Entität immer existieren wird, kann die Abfrage-Engine durch die explizite Überprüfung dieser Eigenschaft optimale Pläne erstellen und die Leistung verbessern.

Stellen Sie sich ein Graphmodell vor, bei dem Typknoten `person` immer über eine Eigenschaft verfügen`name`. Anstatt das zu tun:

```
MATCH (n:person)
RETURN n.name
```

Überprüfen Sie das Vorhandensein einer Eigenschaft in der Abfrage explizit mit einer IS-NOT-NULL-Prüfung:

```
MATCH (n:person)
WHERE n.name IS NOT NULL
RETURN n.name
```

Verwenden Sie keinen benannten Pfad (es sei denn, er ist erforderlich)

Ein benannter Pfad in einer Abfrage ist immer mit zusätzlichen Kosten verbunden, was zu zusätzlichen Kosten in Form einer höheren Latenz und Speicherauslastung führen kann. Betrachten Sie folgende Abfrage:

```
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
```

In der obigen Abfrage ist die Verwendung des Pfads „p“ unnötig, vorausgesetzt, wir wollen nur die Eigenschaften der Knoten kennen. Durch die Angabe des benannten Pfads als Variable wird die Aggregationsoperation mit DISTINCT sowohl in Bezug auf Zeit als auch in Bezug auf den Speicherverbrauch teuer. Eine optimiertere Version der obigen Abfrage könnte sein:

```
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
```

Vermeiden Sie COLLECT (DISTINCT ())

> **Note**
>
> Ab Engine-Version 1.4.7.0 ist dieses empfohlene Umschreiben nicht mehr erforderlich.

COLLECT (DISTINCT ()) wird immer dann verwendet, wenn eine Liste mit unterschiedlichen Werten gebildet werden soll. COLLECT ist eine Aggregationsfunktion, und die Gruppierung erfolgt auf der Grundlage zusätzlicher Schlüssel, die in derselben Anweisung projiziert werden. Wenn distinct verwendet wird, wird die Eingabe in mehrere Blöcke aufgeteilt, wobei jeder Abschnitt eine Gruppe bezeichnet, die reduziert werden soll. Die Leistung wird beeinträchtigt, wenn die Anzahl der Gruppen zunimmt. In Neptune ist es viel effizienter, DISTINCT vor collecting/forming der eigentlichen Liste auszuführen. Auf diese Weise kann die Gruppierung direkt auf den Gruppierungstasten für den gesamten Block erfolgen.

Betrachten Sie folgende Abfrage:

```
MATCH (n:Person)-[:commented_on]->(p:Post)
WITH n, collect(distinct(p.post_id)) as post_list
RETURN n, post_list
```

Eine optimalere Art, diese Abfrage zu schreiben, ist:

```
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
```

Ziehen Sie beim Abrufen aller Eigenschaftswerte die Eigenschaftenfunktion der Suche nach einzelnen Eigenschaften vor

Die `properties()` Funktion wird verwendet, um eine Map zurückzugeben, die alle Eigenschaften für eine Entität enthält, und ist viel effizienter als die individuelle Rückgabe von Eigenschaften.

Angenommen, Ihre `Person` Knoten enthalten 5 Eigenschaften,`firstName`,`lastName`, und `agedept`,, und`company`, dann wäre die folgende Abfrage vorzuziehen:

```
MATCH (n:Person)
WHERE n.dept = 'AWS'
RETURN properties(n) as personDetails
```

Anstatt Folgendes zu verwenden:

```
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
```

Führen Sie statische Berechnungen außerhalb der Abfrage durch

Es wird empfohlen, statische Berechnungen (einfache mathematical/string Operationen) auf der Clientseite aufzulösen. Stellen Sie sich dieses Beispiel vor, bei dem Sie alle Personen suchen möchten, die ein Jahr oder jünger als der Autor sind:

```
MATCH (m:Message)-[:HAS_CREATOR]->(p:person)
WHERE p.age <= ($age + 1)
RETURN m
```

Hier `$age` wird es über Parameter in die Abfrage eingespeist und dann zu einem festen Wert hinzugefügt. Dieser Wert wird dann mit verglichen`p.age`. Stattdessen wäre es besser, die Addition auf der Client-Seite vorzunehmen und den berechneten Wert als Parameter $ageplusone zu übergeben. Dies hilft der Abfrage-Engine, optimierte Pläne zu erstellen, und vermeidet statische Berechnungen für jede eingehende Zeile. Nach diesen Richtlinien wäre eine effizientere Version der Abfrage wie folgt:

```
MATCH (m:Message)-[:HAS_CREATOR]->(p:person)
WHERE p.age <= $ageplusone
RETURN m
```

Batch-Eingaben mit UNWIND anstelle von Einzelanweisungen

Wenn dieselbe Abfrage für verschiedene Eingaben ausgeführt werden muss, wäre es viel leistungsfähiger, eine Abfrage für einen Stapel von Eingaben auszuführen, anstatt eine Abfrage pro Eingabe auszuführen.

Wenn Sie auf einer Gruppe von Knoten zusammenführen möchten, besteht eine Option darin, eine Zusammenführungsabfrage pro Eingabe auszuführen:

```
MERGE (n:Person {`~id`: $id})
SET n.name = $name, n.age = $age, n.employer = $employer
```

Mit Parametern:

```
params = {id: '1', name: 'john', age: 25, employer: 'Amazon'}
```

Die obige Abfrage muss für jede Eingabe ausgeführt werden. Dieser Ansatz funktioniert zwar, erfordert jedoch möglicherweise die Ausführung vieler Abfragen für einen großen Satz von Eingaben. In diesem Szenario kann Batching dazu beitragen, die Anzahl der auf dem Server ausgeführten Abfragen zu reduzieren und den Gesamtdurchsatz zu verbessern.

Verwenden Sie das folgende Muster:

```
UNWIND $persons as person
MERGE (n:Person {`~id`: person.id})
SET n += person
```

Mit Parametern:

```
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'}...]}
```

Es wird empfohlen, mit verschiedenen Chargengrößen zu experimentieren, um herauszufinden, was für Ihren Workload am besten geeignet ist.

Verwenden Sie lieber „Benutzerdefiniert“ IDs für Knoten/Beziehungen

Neptune ermöglicht es Benutzern, Knoten und Beziehungen explizit zuzuweisen IDs . Die ID muss im Datensatz global eindeutig und deterministisch sein, um nützlich zu sein. Eine deterministische

ID kann genau wie Eigenschaften als Such- oder Filtermechanismus verwendet werden. Die Verwendung einer ID ist jedoch aus Sicht der Abfrageausführung viel optimierter als die Verwendung von Eigenschaften. Die Verwendung von benutzerdefinierten IDs Funktionen bietet mehrere Vorteile -

- Eigenschaften können für eine bestehende Entität Null sein, aber die ID muss existieren. Dadurch kann die Abfrage-Engine während der Ausführung einen optimierten Join verwenden.
- Wenn Abfragen mit gleichzeitigen Mutationen ausgeführt werden, verringert sich die Wahrscheinlichkeit, dass Ausnahmen (CMEs) gleichzeitig geändert werden, wenn sie für den Zugriff auf Knoten verwendet IDs werden, erheblich, da aufgrund ihrer erzwungenen Eindeutigkeit weniger Sperren IDs als Eigenschaften übernommen werden.
- Durch die Verwendung IDs wird die Möglichkeit vermieden, dass doppelte Daten erstellt werden, da Neptune im Gegensatz zu Eigenschaften die Einzigartigkeit erzwingt. IDs

Im folgenden Abfragebeispiel wird eine benutzerdefinierte ID verwendet:

> **Note**
>
> Die Eigenschaft `~id` wird verwendet, um die ID anzugeben, wohingegen sie einfach wie jede andere Eigenschaft gespeichert `id` wird.

```
CREATE (n:Person {`~id`: '1', name: 'alice'})
```

Ohne eine benutzerdefinierte ID zu verwenden:

```
CREATE (n:Person {id: '1', name: 'alice'})
```

Wenn Sie den letztgenannten Mechanismus verwenden, wird die Eindeutigkeit nicht erzwungen und Sie könnten die Abfrage später ausführen:

```
CREATE (n:Person {id: '1', name: 'john'})
```

Dadurch wird ein zweiter Knoten mit `id=1` named `john` erstellt. In diesem Szenario hätten Sie jetzt zwei Knoten mit `id=1` jeweils einem anderen Namen (alice und john).

Vermeiden Sie es, ~id-Berechnungen in der Abfrage durchzuführen

Wenn Sie „custom“ IDs in den Abfragen verwenden, führen Sie immer statische Berechnungen außerhalb der Abfragen durch und geben Sie diese Werte in den Parametern an. Wenn statische Werte bereitgestellt werden, ist die Engine besser in der Lage, Suchvorgänge zu optimieren und zu vermeiden, dass diese Werte gescannt und gefiltert werden.

Wenn Sie Kanten zwischen Knoten erstellen möchten, die in der Datenbank vorhanden sind, könnte eine Option wie folgt aussehen:

```
UNWIND $sections as section
MATCH (s:Section {`~id`: 'Sec-' + section.id})
MERGE (s)-[:IS_PART_OF]->(g:Group {`~id`: 'g1'})
```

Mit Parametern:

```
parameters={sections: [{id: '1'}, {id: '2'}]}
```

In der obigen Abfrage wird der `id` Wert des Abschnitts in der Abfrage berechnet. Da die Berechnung dynamisch ist, kann die Engine IDs nicht statisch einbinden und scannt am Ende alle Abschnittsknoten. Die Engine führt dann eine Nachfilterung nach den erforderlichen Knoten durch. Dies kann kostspielig sein, wenn die Datenbank viele Abschnittsknoten enthält.

Ein besserer Weg, dies zu erreichen, besteht darin, die IDs, die `Sec-` an die Datenbank übergeben werden, voranzustellen:

```
UNWIND $sections as section
MATCH (s:Section {`~id`: section.id})
MERGE (s)-[:IS_PART_OF]->(g:Group {`~id`: 'g1'})
```

Mit Parametern:

```
parameters={sections: [{id: 'Sec-1'}, {id: 'Sec-2'}]}
```

Aktualisieren/Zusammenführen mehrerer Knoten

Beim Ausführen `MERGE` von `CREATE` Abfragen auf mehreren Knoten wird empfohlen, eine `UNWIND` in Kombination mit einer einzelnen MERGE/CREATE Klausel zu verwenden, anstatt eine Klausel

für jeden Knoten zu verwenden. MERGE/CREATE Abfragen, die eine Klausel für einen Knoten verwenden, führen zu einem ineffizienten Ausführungsplan, da jede Zeile optimiert werden muss. Dies führt dazu, dass der Großteil der Ausführungszeit der Abfrage für die statische Verarbeitung und nicht für die eigentliche Aktualisierung aufgewendet wird.

Eine Klausel pro Knoten ist nicht optimal, da sie nicht mit zunehmender Anzahl von Knoten skaliert:

```
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'}
```

Die Verwendung von `UNWIND` in Verbindung mit einer MERGE/CREATE Klausel ermöglicht dasselbe Verhalten, aber einen optimaleren Ausführungsplan. Vor diesem Hintergrund würde die geänderte Abfrage wie folgt aussehen:

```
## 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')
```

Bewährte Methoden in Neptune für die Verwendung von SPARQL

Befolgen Sie diese bewährten Methoden bei der Verwendung der SPARQL-Abfragesprache mit Neptune. Informationen zur Verwendung von SPARQL in Neptune finden Sie unter Zugriff auf das Neptune-Diagramm mit SPARQL.

Themen

- Standardmäßige Abfrage aller benannten Graphen

- Angeben eines benannten Graphen für Load
- Auswählen zwischen FILTER, FILTER... IN und VALUES in Ihren Abfragen

Standardmäßige Abfrage aller benannten Graphen

Amazon Neptune ordnet jedem Triple einen benannten Graphen zu. Der Standard-Graph ist als die Vereinigung aller benannten Graphen definiert.

Wenn Sie eine SPARQL-Anfrage absenden, ohne ausdrücklich einen Graphen über das `GRAPH`-Schlüsselwort oder Konstrukte wie `FROM NAMED` anzugeben, berücksichtigt Neptune stets alle Triples in Ihrer DB-Instance. Die folgende Abfrage gibt beispielsweise alle Triples von einem Neptune-SPARQL-Endpunkt zurück:

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

Triples, die in mehr als einem Graphen vorhanden sind, werden nur einmal zurückgegeben.

Weitere Informationen über die Standard-Graph-Spezifikation finden Sie im RDF-Dataset-Abschnitt der Query Language SPARQL 1.1-Spezifikation.

Angeben eines benannten Graphen für Load

Amazon Neptune ordnet jedem Triple einen benannten Graphen zu. Wenn Sie beim Laden, Einfügen oder Aktualisieren von Triples keinen benannten Graphen angeben, verwendet Neptune den benannten Fallback-Graphen, `http://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph`, der durch die URI definiert ist.

Wenn Sie den Neptune-Bulk-Loader verwenden, können Sie den benannten Graphen für alle Triples (oder Quads mit leerer vierter Position) angeben. Dazu verwenden Sie den `parserConfiguration: namedGraphUri`-Parameter. Weitere Informationen zur `Load`-Befehlssyntax des Neptune-Bulk-Loaders finden Sie unter the section called “Loader-Befehl”.

Auswählen zwischen FILTER, FILTER... IN und VALUES in Ihren Abfragen

Es gibt drei grundlegende Möglichkeiten zum Einfügen von Werten in SPARQL-Abfragen: `FILTER`, `FILTER...IN` und `VALUES`.

Angenommen, Sie möchten die Freunde mehrerer Personen in einer Abfrage nachschlagen. Bei Wahl von `FILTER` könnten Sie Ihre Abfrage wie folgt strukturieren:

```
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)}
```

Dadurch werden alle Triples in dem Graphen zurückgegeben, bei denen `?s` an `ex:person1` oder `ex:person2` gebunden ist oder bei denen ein ausgehendes Edge mit der Bezeichnung `foaf:knows` vorhanden ist.

Sie können auch eine Abfrage mit `FILTER...IN` erstellen, bei der gleichwertige Ergebnisse zurückgegeben werden:

```
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))}
```

Sie können auch eine Abfrage mit `VALUES` erstellen, die in diesem Fall auch gleichwertige Ergebnisse zurückgibt:

```
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}}
```

Obwohl diese Abfragen in vielen Fällen semantisch gleichwertig sind, gibt es einige Fälle, in denen sich die beiden `FILTER`-Varianten von der `VALUES`-Variante unterscheiden:

- Der erste Fall tritt ein, wenn Sie doppelte Werte einfügen, wie z. B., wenn Sie dieselbe Person zweimal einfügen. In diesem Fall schließt die Abfrage `VALUES` die Duplikate in Ihr Ergebnis ein. Sie können solche Duplikate explizit eliminieren, indem Sie einen `DISTINCT`-Parameter zur `SELECT`-Klausel hinzufügen. Unter bestimmten Umständen kann es jedoch sinnvoll sein, wenn Abfrageergebnisse beim Einschluss redundanter Werte Duplikate enthalten.

  Die `FILTER...IN`- und `FILTER`-Versionen extrahieren den Wert jedoch nur einmal, wenn derselbe Wert mehrmals vorkommt.

- Der zweite Fall bezieht sich auf die Tatsache, dass `VALUES` immer eine genaue Übereinstimmung vornimmt, während `FILTER` in einigen Fällen eine Typenhochstufung anwenden oder eine Fuzzyübereinstimmung vornehmen kann.

  Wenn Sie beispielsweise ein Literal wie z. B. `"2.0"^^xsd:float` in Ihre Werte-Klausel einschließen, entspricht eine `VALUES`-Abfrage genau diesem Literal, einschließlich Literalwert und Datentyp.

  Im Gegensatz dazu produziert `FILTER` für diese numerischen Literale eine Fuzzyübereinstimmung. Die Übereinstimmungen könnten Literale mit dem gleichen Wert, aber mit unterschiedlichen numerischen Datentypen, wie z. B. `xsd:double`, enthalten.

  > **Note**
  >
  > Es gibt keinen Unterschied zwischen dem `VALUES` Verhalten `FILTER` und bei der Aufzählung von Zeichenfolgenliteralen oder. URIs

Die Unterschiede zwischen `FILTER` und `VALUES` können sich auf die Optimierung und die resultierende Abfrageauswertungsstrategie auswirken. Außer wenn Ihr Anwendungsfall eine Fuzzyübereinstimmung erfordert, raten wir zur Verwendung von `VALUES`, da damit eine Suche nach besonderen Fällen im Zusammenhang mit der Umwandlung von Datentypen vermieden wird. Dies hat zur Folge, dass `VALUES` häufig eine effizientere Abfrage produziert, die schneller und kostengünstiger ausgeführt wird.

Bewährte Methoden für die Verwendung von Neptune mit Amazon ECS und Amazon EKS

Wenn Sie Ihre Diagrammanwendung in Containern auf Amazon Elastic Container Service (Amazon ECS) oder Amazon Elastic Kubernetes Service (Amazon EKS) ausführen, werden Überlegungen angestellt, die für herkömmliche Serverbereitstellungen mit langer Laufzeit nicht gelten. Container sind kurzlebig — sie starten, stoppen und ersetzen sich gegenseitig bei fortlaufenden Updates und Skalierungsereignissen. Dieser Lebenszyklus wirkt sich darauf aus, wie Ihre Anwendung Verbindungen zu Neptune verwaltet, Hostwechsel durchführt und sich bei IAM authentifiziert.

In den folgenden Abschnitten werden praktische Überlegungen zur Ausführung von Neptune-Workloads in containerisierten Umgebungen beschrieben.

Themen

- Verwalten Sie Verbindungspools und den Lebenszyklus in containerisierten Umgebungen
- Erledigen Sie den Host-Austausch und den Verbindungsabbruch
- Konfigurieren Sie Netzwerke, Sicherheitsgruppen und IAM-Authentifizierung

Verwalten Sie Verbindungspools und den Lebenszyklus in containerisierten Umgebungen

Die effektive Verwaltung von Verbindungen zu Neptune ist in containerisierten Umgebungen, in denen Container (Pods in Kubernetes) häufig gestartet, gestoppt und skaliert werden, unerlässlich. Neptune erzwingt mehrere WebSocket Verbindungslimits:

- Jeder Instanztyp hat eine maximale Anzahl gleichzeitiger Verbindungen WebSocket
- Verbindungen im Leerlauf werden nach 20—25 Minuten Inaktivität geschlossen
- Bei der IAM-Authentifizierung werden Verbindungen unabhängig von der Aktivität nach 10 Tagen zwangsweise geschlossen (siehe Verbindungslimits) WebSocket

Einige Treiber unterstützen regelmäßige Keep-Alive-Heartbeats, um Verbindungsabbrüche im Leerlauf zu verhindern. Beispielsweise `keepAliveInterval` sendet der Gremlin-Java-Treiber regelmäßig einen Ping an den Server. Ermitteln Sie anhand der Konfigurationsoptionen Ihres jeweiligen Treibers, welche Keep-Alive-Einstellungen verfügbar sind und wie sie mit dem Leerlauf-Timeout von Neptune interagieren.

Wenn Sie diese Grenzen kennen, können Sie die Verbindungsverwaltung so gestalten, dass sie gut mit Container-Orchestrierungsmustern wie rollierenden Bereitstellungen, Autoscaling und Container-Recycling funktioniert. Allgemeine Hinweise zur Verbindungsverwaltung mit dem Java-Gremlin-Treiber finden Sie unter und. Schließen des Clients zur Vermeidung des Verbindungs-Limits Erstellen einer neuen Verbindung nach einem Failover

Schließen Sie die Verbindungen beim Herunterfahren des Containers

Implementieren Sie ein ordnungsgemäßes Herunterfahren, indem Sie einen `SIGTERM` Handler hinzufügen, der Ihren Graph-Client schließt, bevor der Container beendet wird. Rufen Sie zum Beispiel Gremlin Java`cluster.close()`, Gremlins `driverRemoteConnection.close()` in Go oder Python oder Bolts `driver.close()` für den Neo4j-Treiber auf. Verwenden Sie in

Kubernetes einen `preStop` Hook, um Verbindungen abzuleiten, bevor `SIGTERM` sie gesendet werden, insbesondere bei fortlaufenden Updates.

> **Note**
>
> Docker und Amazon ECS liefern nur `SIGTERM` an PID 1 im Container. Verwenden Sie das Exec-Formular in Ihrem Dockerfile (z. B. `ENTRYPOINT ["java", "-jar", "app.jar"]` nicht`ENTRYPOINT java -jar app.jar`) oder einen Init-Prozess, `tini` um sicherzustellen, dass Ihre Anwendung das Signal empfängt.

Verwenden Sie eine einzelne Client-Instanz pro Container

Das Erstellen mehrerer Gremlin `DriverRemoteConnection` - oder `neo4j.Driver` Bolt-Instanzen im selben Container vervielfacht die Verbindungsnutzung. Teilen Sie eine einzelne Client-Instanz für alle Threads im Container. Die Client-Objekte sind threadsicher.

Größe der Verbindungspools im Verhältnis zu Ihrer Flotte

Stellen Sie `maxConnectionPoolSize` es so ein, dass die Gesamtzahl der Verbindungen in allen Containern das Verbindungslimit für Neptune-Instanzen nicht überschreitet. Wenn Sie beispielsweise 20 Container mit jeweils 8 Threads ausführen, berechnen Sie die Poolgröße pro Container so, dass das 20-fache der Poolgröße innerhalb des Instance-Limits für Ihren Instance-Typ bleibt. Weitere Informationen zur Konfiguration der Verbindungspoolgrößen finden Sie unterSetzen Sie `maxInProcessPerConnection` und `maxSimultaneousUsagePerConnection` auf den selben Wert..

Legen Sie ein Timeout für die Verbindungswartezeit fest

Stellen Sie im Java-Gremlin-Treiber einen angemessenen Wert ein`maxWaitForConnection`, z. B. 5—10 Sekunden. Die Standardeinstellung ist oft unendlich oder sehr lang, was dazu führt, dass Threads hängen bleiben, wenn der Pool erschöpft ist, und nicht schnell mit einem eindeutigen Fehler ausfallen. Nicht alle Sprachtreiber unterstützen diese Einstellung. Die entsprechende Konfigurationsoption finden Sie in der Dokumentation Ihres Treibers.

Erledigen Sie den Host-Austausch und den Verbindungsabbruch

Wenn Neptune einen Host ersetzt (z. B. während einer Wartung oder eines Failovers), werden bestehende Verbindungen zu diesem Host ungültig. In containerisierten Umgebungen kann dies dazu

führen, dass alle Threads in einem Container zum Stillstand kommen, wenn der Client die Ersetzung nicht ordnungsgemäß handhabt.

Verwenden Sie aktuelle Client-Versionen

Wenn Sie die Gremlin-Abfragesprache verwenden, verwenden Sie eine TinkerPop Treiberversion, die mit Ihrer Neptune-Engine-Version kompatibel ist (siehe Zugriff auf ein Neptune-Diagramm mit Gremlin Kompatibilitätstabelle). Wenn Sie den Java-Treiber verwenden, sollten Sie einen Wrapper für `neptune-gremlin-client` den TinkerPop Java-Treiber in Betracht ziehen, der Verbindungsverwaltungsfunktionen wie die Überprüfung des Endpunkts und die Failover-Behandlung hinzufügt. Er folgt den gleichen Versionskompatibilitätsregeln wie der zugrunde liegende TinkerPop Treiber.

Verwenden Sie `neptune-gremlin-client` Version 3.x (oder mindestens Version 2.0.7), je nachdem, was Ihre Neptune-Version zulässt. Diese neueren Versionen verbessern die Ausfallsicherheit und die Verbindungsverarbeitung.

OpenCypher-Benutzer mit dem Neo4j-Treiber sollten das `Driver` Objekt schließen und neu erstellen, wenn Sie während des Failovers einen Verbindungsfehler feststellen. Neptune unterstützt die Bolt-Protokollversionen 1 bis 4.0. Weitere Informationen finden Sie unter Bewährte Methoden für Neptune mit openCypher und Bolt.

Verwenden Sie Cluster- oder Reader-Endpunkte

Stellen Sie keine direkte Verbindung zu Instanzendpunkten her. Verwenden Sie den Cluster-Endpunkt für Schreibvorgänge und den Reader-Endpunkt für Lesevorgänge. Wenn Sie Instanzendpunkte mit verwenden müssen`neptune-gremlin-client`, aktivieren Sie die Filterung der Endpunkt-Integritätsprüfung über die API. `/status`

Konfigurieren Sie Liveness Tests mit Toleranz

Stellen Sie Ihren Kubernetes-Liveness Probe `failureThreshold` auf mindestens 30 mit einem Zeitraum von 10 Sekunden (insgesamt 300 Sekunden) ein. Dadurch wird verhindert, dass Kubernetes Pods während des Zeitfensters von etwa 5 Minuten neu startet, in dem Neptune einen Host-Austausch abschließt.

Implementieren Sie Retry mit Backoff

Eine einzelne fehlgeschlagene Anfrage beim Host-Austausch sollte den Container nicht zum Absturz bringen. Implementieren Sie eine Wiederholungslogik mit exponentiellem Backoff bei Verbindungsausfällen, sodass vorübergehende Fehler beim Austausch ohne Eingreifen behoben

werden können. Hinweise zu Ausnahmen, die wiederholt werden können, finden Sie unter Ausnahmen für Neptune-Transaktionen.

Konfigurieren Sie Netzwerke, Sicherheitsgruppen und IAM-Authentifizierung

Container-Bereitstellungen auf Amazon ECS oder Amazon EKS erfordern eine korrekte Netzwerkkonfiguration und IAM-Anmeldeinformationsverwaltung, um eine Verbindung zu Neptune herzustellen.

Checkliste für Sicherheitsgruppen

- Die Neptune-Cluster-Sicherheitsgruppe muss eingehendes TCP auf Port 8182 (oder dem für Ihren Neptune-Cluster konfigurierten benutzerdefinierten Port) vom Amazon EKS-Knoten oder Pod-CIDR-Bereich zulassen.
- Wenn Sie das Amazon VPC CNI-Plugin mit benutzerdefiniertem Netzwerk verwenden, stellen Sie sicher, dass das Pod-Subnetz über eine Route zum Neptune-Subnetz verfügt.
- Neptune muss sich in derselben Amazon VPC wie Ihr Amazon ECS- oder Amazon EKS-Cluster befinden oder über Amazon VPC-Peering oder Transit Gateway verbunden sein.

Überprüfen Sie die Konnektivität innerhalb des Containers

Bevor Sie Probleme auf Anwendungsebene debuggen, überprüfen Sie den Netzwerkpfad, indem Sie im Container einen der folgenden Befehle ausführen:

AWS CLI

```
aws neptunedata get-engine-status \
  --endpoint-url https://your-neptune-endpoint:8182
```

Weitere Informationen finden Sie unter get-engine-status in der Befehlsreferenz. 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**
>
> In diesem Beispiel wird davon ausgegangen, dass Ihre AWS Anmeldeinformationen in Ihrer Umgebung konfiguriert sind. Ersetze es *us-east-1* durch die Region deines Neptun-Clusters.

curl

```
curl -k https://your-neptune-endpoint:8182/status
```

> **Note**
>
> Die `-k` Flagge überspringt die Überprüfung des Zertifikats. Verwenden Sie dies nur für Konnektivitätstests, nicht für Produktionscode.

Eine erfolgreiche Antwort bestätigt, dass Sicherheitsgruppen, Routing und DNS-Auflösung funktionieren. Ein Timeout weist auf ein Problem auf Netzwerkebene hin.

Aktualisierung der IAM-Anmeldeinformationen

Wenn Sie die IAM-Authentifizierung mit Neptune verwenden, müssen Ihre Pods die Anmeldeinformationen aktualisieren, bevor sie ablaufen. Wir empfehlen Amazon EKS Pod Identity für neue Bereitstellungen. IAM Roles for Service Accounts (IRSA) wird ebenfalls unterstützt und ist weit verbreitet.

Rufen Sie bei beiden Ansätzen die Anmeldeinformationen innerhalb der Authentifizierungsgeneratorfunktion ab und nicht nur einmal beim Start. IRSA- und Pod Identity-

Token laufen nach 1 Stunde ab. Wenn Sie Anmeldeinformationen bei der Initialisierung zwischenspeichern, schlagen Anfragen fehl, sobald das Token `AccessDeniedException: security token expired` abgelaufen ist.

- Für gremlin-go: Aufruf `cfg.Credentials.Retrieve(ctx)` innerhalb der Funktion, an die übergeben wurde. `gen NewDynamicAuth`
- Für Java: `NeptuneGremlinClient` mit einem verwendenHandshakeInterceptor, das die SigV4-Signatur bei jeder neuen Verbindung aktualisiert.

Cross-account Zugriff

Neptune unterstützt keine direkte kontoübergreifende IAM-Authentifizierung. Wenn sich Ihr Amazon EKS-Cluster in einem anderen AWS Konto als Ihr Neptune-Cluster befindet, muss der Pod zunächst eine Rolle im Neptune-Konto annehmen, bevor er sich authentifiziert.

Eingangskonfiguration für Graph Explorer

Wenn Sie Graph Explorer in einer containerisierten Umgebung ausführen, stellen Sie sicher, dass der Proxy auf alle Pfade zugreifen kann — nicht nur. `/explorer` Konfigurieren Sie in Amazon EKS Ihre Ingress-Controller-Regeln so, dass alle Pfade zulässig sind. Konfigurieren Sie auf Amazon ECS Ihre Application Load Balancer Balancer-Zielgruppe und Listener-Regeln so, dass alle Pfade an den Graph Explorer-Container weitergeleitet werden. Der Proxy von Graph Explorer ruft, `/summary` / `gremlin/openCypher`, und auf. `/sparql` Path-restricted Routing-Regeln unterbrechen den Proxy im Hintergrund.

Bewährte Methoden für die Verwendung des AWS CLI und SDKs mit Neptune

Standardmäßig laufen die AWS SDKs nach 60 Sekunden ab AWS CLI und versuchen automatisch, fehlgeschlagene Anfragen erneut auszuführen. Bei Abfragen, deren Abschluss länger als 60 Sekunden dauert, führt dieses Standardverhalten zur doppelten Ausführung von Abfragen. Durch den erneuten Versuch wird eine neue Anforderung ausgelöst, während die ursprüngliche Abfrage noch auf dem Server ausgeführt wird.

> **Important**
>
> Wenn Sie Abfragen mit langer Laufzeit über die Neptune-Datenebene-API ausführen (z. B.`execute-open-cypher-query`), deaktivieren Sie automatische Wiederholungen.

Erhöhen oder entfernen Sie auch das Lese-Timeout. Andernfalls wiederholt der Client die Abfrage, während das Original noch ausgeführt wird, wodurch sich die Belastung Ihres Neptune-Clusters verdoppelt.

Die folgenden Empfehlungen gelten für alle Neptune-Datenebenenoperationen, die länger als das standardmäßige 60-Sekunden-Timeout laufen können:

- Setzen Sie das Lese-Timeout auf 0 (unendlich) oder einen Wert, der größer ist als die längste erwartete Abfragedauer.
- Setzen Sie die maximale Gesamtzahl der Versuche auf 1 (ein Versuch, keine Wiederholungen). Dadurch wird verhindert, dass der Client eine Anfrage, die der Server noch verarbeitet, erneut versucht.

Die folgenden Beispiele zeigen, wie Sie diese Einstellungen für die AWS CLI und beliebte SDKs konfigurieren.

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
```

Der `--cli-read-timeout 0` Parameter deaktiviert das Lese-Timeout, sodass sie unbegrenzt AWS CLI auf eine Antwort warten können. Die `AWS_MAX_ATTEMPTS=1` Umgebungsvariable legt die Gesamtzahl der maximalen Versuche auf 1 fest, was bedeutet, dass die Anfrage genau einmal ohne Wiederholungen versucht wird.

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
    )
)
```

Diese Einstellung `read_timeout=None` deaktiviert das Lese-Timeout. Die Einstellung `total_max_attempts` auf 1 bedeutet, dass die Anfrage genau einmal ohne erneute Versuche versucht wird.

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();
```

Die Einstellung `apiCallTimeout` auf `Duration.ZERO` konfiguriert ein unendliches Timeout für den API-Aufruf. Durch die Verwendung werden alle `RetryPolicy.none()` automatischen Wiederholungsversuche deaktiviert.

Amazon Neptune-Limits

Regionen

Amazon Neptune ist in den folgenden AWS Regionen verfügbar:

- USA Ost (Nord-Virginia): `us-east-1`
- USA Ost (Ohio): `us-east-2`
- USA West (Nordkalifornien): `us-west-1`
- USA West (Oregon): `us-west-2`
- Kanada (Zentral): `ca-central-1`
- Kanada West (Calgary): `ca-west-1`
- Südamerika (São Paulo): `sa-east-1`
- Europa (Stockholm): `eu-north-1`
- Europa (Spanien): `eu-south-2`
- Europa (Irland): `eu-west-1`
- Europa (London): `eu-west-2`
- Europa (Paris): `eu-west-3`
- Europa (Frankfurt): `eu-central-1`
- Naher Osten (Bahrain): `me-south-1`
- Naher Osten (VAE): `me-central-1`
- Israel (Tel Aviv): `il-central-1`
- Afrika (Kapstadt): `af-south-1`
- Asien-Pazifik (Hongkong): `ap-east-1`
- Asien-Pazifik (Tokio): `ap-northeast-1`
- Asien-Pazifik (Seoul): `ap-northeast-2`
- Asien-Pazifik (Osaka): `ap-northeast-3`
- Asien-Pazifik (Singapur): `ap-southeast-1`
- Asien-Pazifik (Sydney): `ap-southeast-2`

- Asien-Pazifik (Jakarta): `ap-southeast-3`
- Asien-Pazifik (Melbourne): `ap-southeast-4`
- Asien-Pazifik (Malaysia): `ap-southeast-5`
- Asien-Pazifik (Mumbai): `ap-south-1`
- Asien-Pazifik (Hyderabad): `ap-south-2`
- China (Peking): `cn-north-1`
- China (Ningxia): `cn-northwest-1`
- AWS GovCloud (US-West): `us-gov-west-1`
- AWS GovCloud (US-East): `us-gov-east-1`

Unterschiede in China-Regionen

Wie bei vielen AWS Diensten funktioniert Amazon Neptune in China (Peking) und China (Ningxia) etwas anders als in anderen Regionen. AWS

Wenn Neptune ML beispielsweise Amazon API Gateway zur Erstellung des Exportservice verwendet, ist die IAM-Authentifizierung standardmäßig aktiviert. In China-Regionen unterscheidet sich das Verfahren für die Änderung dieser Option geringfügig vom Verfahren für andere Regionen.

Diese und andere Unterschiede werden hier erklärt.

Unterschiede in der Region Asien-Pazifik (Hyderabad)

Die folgenden Neptun-Funktionen werden in der Region Asien-Pazifik (Hyderabad) (`ap-south-2`) nicht unterstützt:

- Globale Neptun-Datenbanken
- Cross-region Kopieren von Schnappschüssen (siehe) Kopieren eines Snapshots

Maximale Größe von Speicher-Cluster-Volumes

Ein Neptun-Cluster-Volume kann in allen unterstützten Regionen auf eine maximale Größe von 128 Tebibyte (TiB) anwachsen. Siehe Speicher, Zuverlässigkeit und Verfügbarkeit von Amazon Neptune.

Unterstützte DB-Instance-Größen

Neptune unterstützt verschiedene DB-Instance-Klassen in verschiedenen AWS Regionen. Wenn Sie wissen möchten, welche Klassen in einer bestimmten Region unterstützt werden, gehen Sie zu Amazon Neptune-Preise und wählen Sie die Region aus, an der Sie interessiert sind.

Grenzwerte für jeden AWS Konto

Für bestimmte Verwaltungs-Features verwendet Amazon Neptune die gleiche operative Technologie wie Amazon Relational Database Service (Amazon RDS).

Jedes AWS Konto hat für jede Region Beschränkungen für die Anzahl der Amazon Neptune- und Amazon RDS-Ressourcen, die Sie erstellen können. Zu diesen Ressourcen gehören DB-Instances und DB-Cluster.

Nachdem eine Größenbeschränkung für eine Ressource erreicht wurde, schlagen zusätzliche Aufrufe zum Erstellen dieser Ressource mit einer Ausnahme fehl.

Eine Liste der Einschränkungen für Amazon Neptune und Amazon RDS finden Sie unter Einschränkungen in Amazon RDS im Amazon-RDS-Benutzerhandbuch.

Verbindung mit Neptune erfordert eine VPC

Amazon Neptune ist ausschließlich ein Virtual-Private-Cloud (VPC)-Service.

Zudem lassen Instances einen Zugriff von außerhalb der VPC nicht zu.

Neptune erfordert SSL

Ab Engine-Version `1.0.4.0` lässt Amazon Neptune ausschließlich Secure-Sockets-Layer (SSL)-Verbindungen über HTTPS zu Instances oder Cluster-Endpunkten zu.

Neptune benötigt TLS-Version 1.2 und verwendet die folgenden Suiten für eine starke Verschlüsselung:

- `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`

Availability Zones und DB-Subnetzgruppen

Amazon Neptune erfordert für jeden Cluster mit Subnetzen in mindestens zwei unterstützten Availability Zones (AZs) eine DB-Subnetzgruppe.

Wir empfehlen die Verwendung von drei oder mehr Subnetzen in verschiedenen Availability Zones.

Maximale HTTP-Anforderungsnutzlast (150 MB)

Gremlin- und SPARQL-HTTP-Anforderungen müssen sich insgesamt auf weniger als 150 MB belaufen. Wenn eine Anforderung diese Größe überschreitet, gibt Neptune `HTTP 400: BadRequestException` zurück.

Dieses Limit gilt nicht für WebSockets Gremlin-Verbindungen.

Unterschiede bei der Gremlin-Implementierung

Für die Die Amazon-Neptune-Gremlin-Implementierung gelten spezifische Implementierungsdetails, die sich von anderen Gremlin-Implementierungen unterscheiden können.

Weitere Informationen finden Sie unter Einhaltung der Gremlin-Standards in Amazon Neptune.

Neptune unterstützt keine Nullzeichen in Zeichenfolgendaten

Neptune unterstützt keine Nullzeichen in Zeichenfolgen. Dies gilt für Eigenschaftsdiagrammdaten für Gremlin und OpenCypher sowie für Daten. RDF/SPARQL

SPARQL UPDATE LOAD aus der URI

`SPARQL UPDATE LOAD` per URI funktioniert nur bei Ressourcen innerhalb derselben VPC.

Dies schließt Amazon-S3-URLs ein, die sich in derselben Region wie der Cluster mit erstelltem Amazon-S3-VPC-Endpunkt befinden.

Die Amazon-S3-URL muss HTTPS verwenden und eine mögliche Authentifizierung muss in der URL enthalten sein. Weitere Informationen finden Sie unter Authentifizieren von Anforderungen: Verwenden von Parametern in der Amazon-Simple-Storage-Service-API-Referenz.

Informationen zum Erstellen eines VPC-Endpunkts finden Sie unter Erstellen eines Amazon-S3-VPC-Endpunkts.

Wenn Sie Daten aus einer Datei laden müssen, sollten Sie die Amazon-Neptune-Loader-API verwenden. Weitere Informationen finden Sie unter Verwenden des Amazon Neptune Bulk Loaders zur Datenaufnahme.

> **Note**
>
> Die Amazon-Neptune-Loader API unterstützt ACID nicht.

IAM-Authentifizierung und Zugriffskontrolle

In Neptune-Engine-Versionen vor Version 1.2.0.0 werden IAM-Authentifizierung und Zugriffskontrolle nur auf DB-Cluster-Ebene unterstützt. Ab Version `1.2.0.0` können Sie den abfragebasierten Zugriff jedoch detaillierter steuern, indem Sie in IAM-Richtlinien Bedingungsschlüssel verwenden. Weitere Informationen finden Sie unter Verwenden von Abfrageaktionen in Neptune-Datenzugriff-Richtlinienanweisungen und Authentifizieren Sie Ihre Amazon Neptune Neptune-Datenbank mit AWS Identity and Access Management

Für die Amazon Neptune Neptune-Konsole sind Berechtigungen erforderlich NeptuneReadOnlyAccess. Sie können den Zugriff für IAM-Benutzer einschränken, indem Sie diesen Zugriff widerrufen. Weitere Informationen finden Sie unter Verwenden AWS verwaltete Richtlinien für den Zugriff auf Amazon Neptune Neptune-Datenbanken.

Amazon Neptune unterstützt keine name/password benutzerbasierte Zugriffskontrolle.

WebSocket gleichzeitige Verbindungen und maximale Verbindungszeit

Die Anzahl der gleichzeitigen WebSocket Verbindungen pro Neptune-DB-Instance ist begrenzt. Wenn dieses Limit erreicht ist, drosselt Neptune jede Anfrage zum Öffnen einer neuen WebSocket Verbindung, um zu verhindern, dass der gesamte zugewiesene Heap-Speicher aufgebraucht wird.

Für alle größeren Instance-Typen, die von Neptune unterstützt werden, und für alle serverlosen Instances beträgt die maximale Anzahl gleichzeitiger WebSocket Verbindungen 32.000 (32.768).

Die maximale Anzahl gleichzeitiger WebSocket Verbindungen für kleinere Instance-Typen ist in der folgenden Tabelle aufgeführt:

Instance-Typ	Maximale Anzahl gleichzeitiger Verbindungen WebSocket
Serverless	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
db.r5d.xlarge	4096
db.r5d.large	2048
db.t3.medium	512
db.t4g.medium	512
db.r6g.4xlarge	16384

Instance-Typ	Maximale Anzahl gleichzeitiger Verbindungen WebSocket
db.r6g.2xlarge	8192
db.r6g.xlarge	4096
db.r6g.large	2048
db.r6gd.4xlarge	16384
db.r6gd.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
db.x2g.4xlarge	16384
db.x2g.2xlarge	8192
db.x1e.xlarge	4096
db.x1e.2xlarge	16384
db.r6i.xlarge	4096

Instance-Typ	Maximale Anzahl gleichzeitiger Verbindungen WebSocket
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 unterstützt keine R4 Instance-Typen mehr.

Wenn ein Client eine Verbindung korrekt geschlossen hat, wird dieses Schließen sofort in der Anzahl der offenen Verbindungen reflektiert.

Wenn der Client eine Verbindung nicht schließt, kann die Verbindung nach einem Leerlauf-Timeout von 20 bis 25 Minuten automatisch geschlossen werden. (Das Leerlauf-Timeout ist die Zeit, die seit

dem Empfang der letzten Nachricht vom Client vergangen ist.) Solange der Wert für den Leerlauf-Timeout jedoch nicht erreicht ist, hält Neptune die Verbindung auf unbestimmte Zeit offen.

Wenn die IAM-Authentifizierung aktiviert ist, wird eine WebSocket Verbindung nach mehr als 10 Tagen immer einige Minuten unterbrochen, sofern sie bis dahin nicht bereits geschlossen wurde.

Einschränkungen für Eigenschaften und Bezeichnungen

Es gibt keine Begrenzung für die Anzahl der Vertices und Edges oder RDF-Quads, die in einem Diagramm zulässig sind.

Es gibt auch keine Begrenzung für die Anzahl der Eigenschaften oder Beschriftungen, die ein Vertex oder Edge haben kann.

Es gibt eine Größenbeschränkung von 55 MB für die Größe einer einzelnen Eigenschaft oder Bezeichnung. In RDF bedeutet dies, dass der Wert in jeder Spalte (S, P, O oder G) eines RDF-Quads 55 MB nicht überschreiten darf.

Wenn Sie in Ihrem Diagramm ein größeres Objekt verknüpfen müssen, z. B. ein Image mit einem Eckpunkt oder Knoten, können Sie es als Datei in Amazon S3 speichern und den Amazon-S3-Pfad als Eigenschaft oder Bezeichnung verwenden.

Einschränkungen, die sich auf den Neptune-Massen-Loader auswirken

Sie können nicht mehr als 64 Neptune-Massenladeaufträge gleichzeitig in die Warteschlange stellen.

Neptune verfolgt nur die letzten 1.024 Massenladeaufträge.

Neptune speichert nur die letzten 10 000 Fehlerdetails pro Auftrag.

Zusammenarbeit mit anderen AWS Diensten

Sie können Amazon Neptune in Verbindung mit vielen anderen AWS Diensten verwenden:

Neptune-Integrationen mit anderen Services

- AWS Glue – AWS Glue ist ein Serverless-Service für die Datenintegration, um Aufträge für Extract, Transform, Load (ETL) für Daten auszuführen.

  Neptune bietet eine Open-Source-Bibliothek neptune-python-utilities, die die Verwendung von Python und Gremlin innerhalb eines Glue-Jobs vereinfacht. Der Neo4j Spark Connector wird auch für die Ausführung von Scala- und openCypher-Glue-Aufträgen unterstützt.
- Amazon SageMaker AI — Amazon SageMaker AI ist eine Plattform für maschinelles Lernen mit vollem Funktionsumfang für die Erstellung, Schulung und Bereitstellung hochwertiger Modelle für maschinelles Lernen.

  Neptune lässt sich hauptsächlich auf zwei SageMaker Arten in KI integrieren:
  - Neptune stellt ein Open-Source-Python-Paket für Jupyter-Notebooks bereit, das im Neptune Graph Notebook-Projekt unter zu finden ist. GitHub Dieses Paket enthält Jupyter-Magics, Tutorial-Notebooks und Codebeispiele in einer interaktiven Programmierumgebung, in der Sie mehr über Diagrammtechnologien und Neptune erfahren können. Neptune bietet eine vollständig verwaltete Umgebung für Jupyter-Notebooks, die von SageMaker KI gehostet werden, und stellt automatisch eine Verbindung zu den Notizbüchern im Open-Source-Neptune Graph Notebook-Projekt her.
  - Das Neptune-ML-Feature ermöglicht das Erstellen und Trainieren nützlicher Machine-Learning-Modelle anhand großer Diagramme in wenigen Stunden statt in Wochen. Um dies zu erreichen, verwendet Neptune ML die Graph Neural Network (GNN) -Technologie, die von Amazon SageMaker AI und der Deep Graph Library (DGL) unterstützt wird.
- AWS Lambda – AWS Lambda -Funktionen können in Neptune-Anwendungen auf viele Arten verwendet werden.

  Informationen zur Verwendung von Lambda-Funktionen mit allen gängigen Gremlin-Treibern und Sprachvarianten sowie spezifische Beispiele für in Java und Python geschriebene Lambda-Funktionen finden Sie unter. JavaScript AWS Lambda Funktionen in Amazon Neptune verwenden
- Amazon Athena – Amazon Athena ist ein interaktiver Abfrageservice, der die Analyse von Daten in Amazon Simple Storage Service und anderen Verbunddatenquellen vereinfacht, die Standard-SQL verwenden.

Neptune stellt einen Konnektor für Athena bereit, der Athena die Kommunikation mit Ihren in Neptune gespeicherten Daten ermöglicht.

- AWS Database Migration Service (AWS DMS) — AWS Database Migration Service ist ein AWS Webdienst, mit dem Sie Daten von einer Datenbank in eine andere migrieren können.

  AWS DMS kann Daten aus unterstützten Quelldatenbanken schnell und sicher in Neptune laden. Die Quelldatenbank bleibt während der Migration voll betriebsbereit, wodurch die Ausfallzeiten für Anwendungen, die von ihr abhängig sind, minimiert werden.

- AWS Backup— AWS Backup ist ein vollständig verwalteter Backup-Service, der es einfach macht, die Sicherung von Daten zwischen AWS Diensten in der Cloud und vor Ort zu zentralisieren und zu automatisieren.

  AWS Backup ermöglicht es Ihnen, mithilfe Ihrer zentralen Datenschutzrichtlinie für die unterstützten AWS Dienste für Datenbank, Speicher und Datenverarbeitung automatische regelmäßige Snapshots von Neptune-Clustern zu erstellen.

- AWS SDK für Pandas — Das AWS SDK für Pandas (früher bekannt als AWS Data Wrangler oder`awswrangler`) ist eine Open-Source-Python-Initiative von AWS Professional Service, die die Leistungsfähigkeit der Python-Datenanalysebibliothek auf mehr als 30 AWS datenbezogene Dienste, darunter Neptune AWS, erweitert, verbindet `DataFrames` und diese verbindet. `pandas`

  Zusätzlich zum SDK gibt es auch ein Tutorial zur Verwendung mit Neptune und mehrere Beispiele für Neptune-Notebooks, nämlich Fraud Ring Detection, Synthetic Identity Detection und Logistics Analysis.

- JDBC-Treiber – Der Neptune-JDBC-Treiber unterstützt openCypher-, Gremlin-, SQL-Gremlin- und SPARQL-Abfragen.

  Die JDBC-Konnektivität vereinfacht die Herstellung von Verbindungen mit Neptune über Business-Intelligence (BI)-Tools wie Tableau.

Neptune-Tools und -Hilfsprogramme

Amazon Neptune bietet eine Reihe von Tools und Dienstprogrammen, mit denen Sie Ihre Arbeit mit einem Graph vereinfachen und automatisieren können. Darunter die Folgenden:

Amazon Neptune-Tools

- Amazon Neptune Utility for GraphQL — Das Amazon Neptune Utility for GraphQL ist ein Node.js Open-Source-Befehlszeilentool, mit dem Sie eine GraphQL-API für eine Neptune Property-Graph-Datenbank erstellen und verwalten können. Es ist eine Methode ohne Code, einen GraphQL-Resolver für GraphQL-Abfragen zu erstellen, die eine variable Anzahl von Eingabeparametern haben und eine variable Anzahl verschachtelter Felder zurückgeben.
- Nodestream — Nodestream ist ein Framework für den Umgang mit der semantischen Modellierung von Daten als Graph. Es ist so konzipiert, dass es flexibel und erweiterbar ist, sodass Sie definieren können, wie Daten gesammelt und als Diagramm modelliert werden. Es verwendet einen Pipeline-basierten Ansatz, um zu definieren, wie Daten gesammelt und verarbeitet werden, und es bietet eine Möglichkeit, zu definieren, wie das Diagramm aktualisiert werden soll, wenn sich das Schema ändert.
- Amazon Neptune MCP Query Servers — Model Context Protocol (MCP) -Server für Amazon Neptune, der sowohl Neptune Database als auch Neptune Analytics unterstützt, indem er die Möglichkeit bietet, Graphabfragen (OpenCypher und Gremlin) auszuführen und das Graphschema abzurufen.
- Amazon Neptune MCP Memory Servers — Model Context Protocol (MCP) -Server für Amazon Neptune, der Agenten Speicher zur Verfügung stellt, der in einem Knowledge Graph für Amazon Neptune gespeichert ist.

Amazon Neptune Neptune-Hilfsprogramm für GraphQL

Das Amazon Neptune Neptune-Hilfsprogramm für GraphQL ist ein Node.js Open-Source-Befehlszeilentool, mit dem Sie eine GraphQL-API für eine Neptune-Property-Graph-Datenbank erstellen und verwalten können (es funktioniert noch nicht mit RDF-Daten). Es ist eine Methode ohne Code, einen GraphQL-Resolver für GraphQL-Abfragen zu erstellen, die eine variable Anzahl von Eingabeparametern haben und eine variable Anzahl verschachtelter Felder zurückgeben.

Es wurde als https://github.com/aws/amazon-neptune-for-graphqlOpen-Source-Projekt veröffentlicht und befindet sich unter.

Sie können das Hilfsprogramm mit NPM wie folgt installieren (weitere Informationen finden Sie unter Installation und Einrichtung):

```
npm i @aws/neptune-for-graphql -g
```

Das Hilfsprogramm kann das Graphschema eines vorhandenen Neptun-Eigenschaftsgraphen ermitteln, einschließlich Knoten, Edges, Eigenschaften und Edge-Kardinalität. Anschließend generiert es ein GraphQL-Schema mit den Anweisungen, die benötigt werden, um die GraphQL-Typen den Knoten und Edges der Datenbank zuzuordnen, und generiert automatisch Resolver-Code. Der Resolver-Code wurde entwickelt, um die Latenz zu minimieren, indem nur die von der GraphQL-Abfrage angeforderten Daten ausgegeben werden.

Sie können auch mit einem vorhandenen GraphQL-Schema und einer leeren Neptune-Datenbank beginnen und das Hilfsprogramm die Anweisungen ableiten lassen, die erforderlich sind, um dieses GraphQL-Schema den Knoten und Edges von Daten zuzuordnen, die in die Datenbank geladen werden sollen. Oder Sie können mit einem GraphQL-Schema und Anweisungen beginnen, die Sie bereits erstellt oder geändert haben.

Das Hilfsprogramm ist in der Lage, alle AWS Ressourcen zu erstellen, die es für seine Pipeline benötigt, einschließlich der AWS AppSync API, der IAM-Rollen, der Datenquelle, des Schemas und des Resolvers sowie der AWS Lambda-Funktion, die Neptune abfragt.

> **Note**
>
> Command-line Die Beispiele hier gehen von einer Linux-Konsole aus. Wenn Sie Windows verwenden, ersetzen Sie die umgekehrten Schrägstriche (‚\') am Ende der Zeilen durch Carets (‚^').

Themen

- Installation und Einrichtung des Amazon Neptune Neptune-Dienstprogramms für GraphQL
- Daten in einer bestehenden Neptune-Datenbank scannen
- Ausgehend von einem GraphQL-Schema ohne Direktiven
- Arbeiten mit Direktiven für ein GraphQL-Schema
- Command-line Argumente für das GraphQL-Hilfsprogramm

Installation und Einrichtung des Amazon Neptune Neptune-Dienstprogramms für GraphQL

Wenn Sie das Hilfsprogramm mit einer vorhandenen Neptune-Datenbank verwenden möchten, benötigen Sie es, um eine Verbindung zum Datenbank-Endpunkt herstellen zu können. Standardmäßig ist eine Neptune-Datenbank nur von der VPC aus zugänglich, in der sie sich befindet.

Da es sich bei dem Hilfsprogramm um ein Node.js Befehlszeilentool handelt, müssen Sie Node.js (Version 18 oder höher) installiert haben, damit das Programm ausgeführt werden kann. Folgen Sie den Anweisungen hier, um Node.js auf einer EC2-Instance in derselben VPC wie Ihre Neptune-Datenbank zu installieren. Die Mindestgröße für die Instance zum Ausführen des Dienstprogramms ist t2.micro. Wählen Sie bei der Erstellung der Instance die Neptune-Datenbank-VPC aus dem Pulldown-Menü Common Security Groups aus.

Verwenden Sie NPM, um das Hilfsprogramm selbst auf einer EC2-Instance oder Ihrem lokalen Computer zu installieren:

```
npm i @aws/neptune-for-graphql -g
```

Anschließend können Sie den Hilfebefehl des Dienstprogramms ausführen, um zu überprüfen, ob es ordnungsgemäß installiert wurde:

```
neptune-for-graphql --help
```

Möglicherweise möchten Sie den auch installieren, um Ressourcen zu verwalten. AWS CLI AWS

Daten in einer bestehenden Neptune-Datenbank scannen

Unabhängig davon, ob Sie mit GraphQL vertraut sind oder nicht, der folgende Befehl ist der schnellste Weg, eine GraphQL-API zu erstellen. Dies setzt voraus, dass Sie das Neptune-Hilfsprogramm für GraphQL wie im Installationsabschnitt beschrieben installiert und konfiguriert haben, sodass es mit dem Endpunkt Ihrer Neptune-Datenbank verbunden ist.

```
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
```

Das Hilfsprogramm analysiert die Datenbank, um das Schema der darin enthaltenen Knoten, Edges und Eigenschaften zu ermitteln. Basierend auf diesem Schema leitet es ein GraphQL-Schema mit zugehörigen Abfragen und Mutationen ab. Dann erstellt es eine AppSync GraphQL-API und die erforderlichen AWS Ressourcen, um sie zu verwenden. Zu diesen Ressourcen gehören ein Paar IAM-Rollen und eine Lambda-Funktion, die den GraphQL-Resolver-Code enthält.

Wenn das Tool fertig ist, finden Sie in der AppSync Konsole eine neue GraphQL-API unter dem Namen, den Sie im Befehl zugewiesen haben. Um es zu testen, verwenden Sie die Option AppSync Abfragen im Menü.

Wenn Sie denselben Befehl erneut ausführen, nachdem Sie der Datenbank weitere Daten hinzugefügt haben, werden die AppSync API und der Lambda-Code entsprechend aktualisiert.

Um alle mit dem Befehl verknüpften Ressourcen freizugeben, führen Sie folgenden Befehl aus:

```
neptune-for-graphql \
  --remove-aws-pipeline-name (your new GraphQL API name from above)
```

Ausgehend von einem GraphQL-Schema ohne Direktiven

Sie können mit einer leeren Neptune-Datenbank beginnen und ein GraphQL-Schema ohne Direktiven verwenden, um die Daten zu erstellen und abzufragen. Der folgende Befehl erstellt automatisch AWS -Ressourcen zu diesem Zweck:

```
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
```

Die GraphQL-Schemadatei muss die GraphQL-Schematypen enthalten, wie im folgenden TODO-Beispiel gezeigt. Das Hilfsprogramm analysiert Ihr Schema und erstellt eine erweiterte Version, die auf Ihren Typen basiert. Es fügt Abfragen und Mutationen für die in der Graphdatenbank gespeicherten Knoten hinzu. Wenn Ihr Schema verschachtelte Typen enthält, fügt es Beziehungen zwischen den Typen hinzu, die als Edges in der Datenbank gespeichert sind.

Das Hilfsprogramm erstellt eine AppSync GraphQL-API und alle erforderlichen AWS Ressourcen. Dazu gehören ein Paar IAM-Rollen und eine Lambda-Funktion, die den GraphQL-Resolver-Code

enthält. Wenn der Befehl abgeschlossen ist, können Sie eine neue GraphQL-API mit dem Namen finden, den Sie in der AppSync Konsole angegeben haben. Um es zu testen, verwenden Sie Queries im AppSync Menü.

Das folgende Beispiel veranschaulicht, wie dies funktioniert:

Todo-Beispiel, ausgehend von einem GraphQL-Schema ohne Direktiven

In diesem Beispiel gehen wir von einem Todo GraphQL-Schema ohne Direktiven aus, das Sie im *???samples???* Verzeichnis finden. Es umfasst diese beiden Typen:

```
type Todo {
  name: String
  description: String
  priority: Int
  status: String
  comments: [Comment]
}

type Comment {
  content: String
}
```

Dieser Befehl verarbeitet das Todo-Schema und einen Endpunkt einer leeren Neptune-Datenbank, um eine GraphQL-API zu erstellen in: 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
```

Das Hilfsprogramm erstellt eine neue Datei im `TodoExample.source.graphql` aufgerufenen Ausgabeordner und die GraphQL-API in AppSync. Das Hilfsprogramm leitet daraus Folgendes ab:

- Im Typ Todo wurde es `@relationship` für einen neuen CommentEdge Typ hinzugefügt. Dadurch wird der Resolver angewiesen, Todo mit Comment zu verbinden, indem er eine Graphdatenbank namens Edge verwendet. CommentEdge

- Es wurde eine neue Eingabe hinzugefügt, die aufgerufen wurde TodoInput , um die Abfragen und Mutationen zu unterstützen.
- Es wurden zwei Abfragen für jeden Typ hinzugefügt (Todo, Comment): eine zum Abrufen eines einzelnen Typs mithilfe eines `id` oder eines der in der Eingabe aufgeführten Typfelder und die andere zum Abrufen mehrerer Werte, die anhand der Eingabe für diesen Typ gefiltert wurden.
- Für jeden Typ wurden drei Mutationen hinzugefügt: erstellen, aktualisieren und löschen. Der zu löschende Typ wird mit einer `id` oder der Eingabe für diesen Typ angegeben. Diese Mutationen wirken sich auf die in der Neptun-Datenbank gespeicherten Daten aus.
- Es wurden zwei Mutationen für Verbindungen hinzugefügt: Verbinden und Löschen. Sie verwenden als Eingabe die Edge-IDs der von Neptune verwendeten Von- und Zielscheitelpunkte, und die Verbindung sind Edges in der Datenbank.

Der Resolver erkennt die Abfragen und Mutationen anhand ihrer Namen, aber Sie können sie wie unten gezeigt anpassen.

Hier ist der Inhalt der `TodoExample.source.graphql`-Datei:

```
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
}

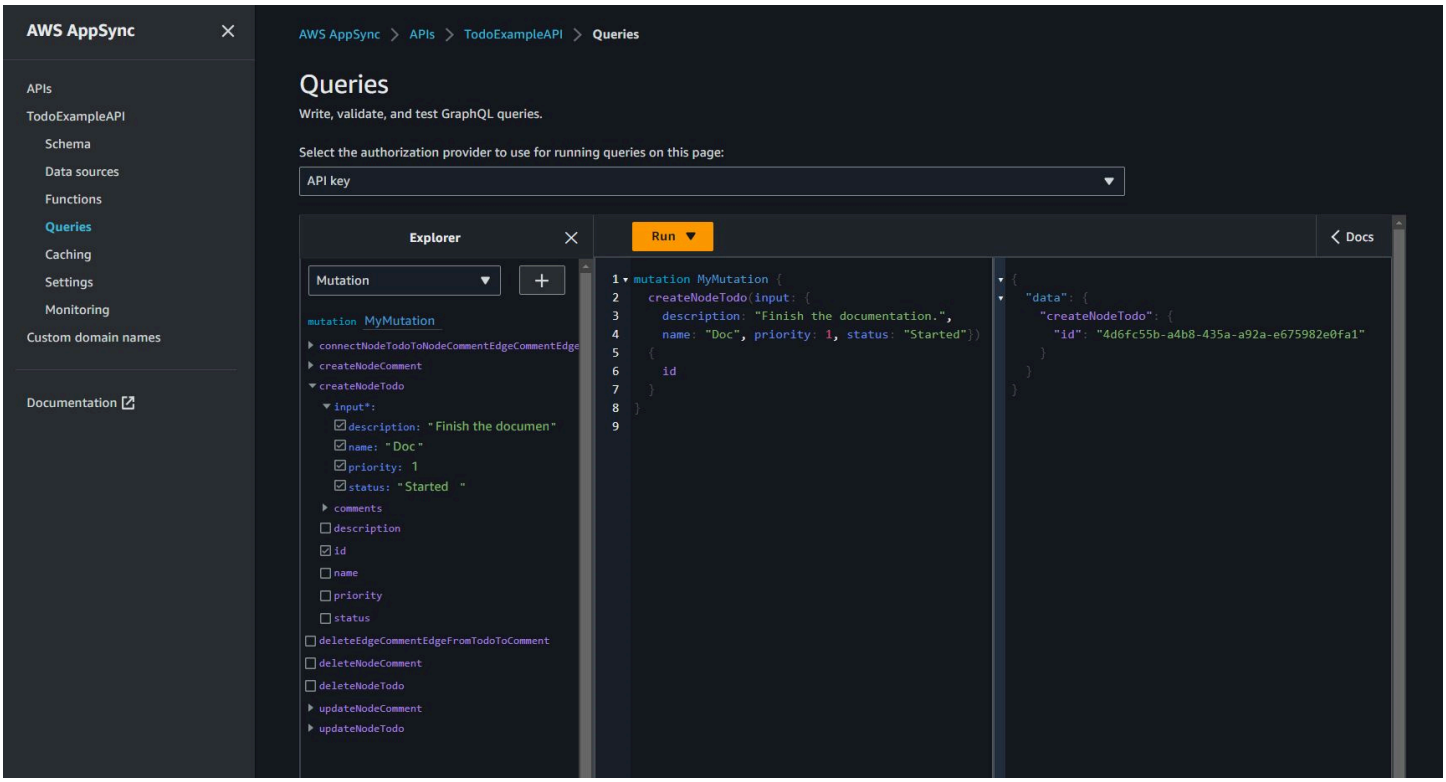
input Options {
  limit: Int
}

type Query {
  getNodeTodo(filter: TodoInput, options: Options): Todo
  getNodeTodos(filter: TodoInput): [Todo]
  getNodeComment(filter: CommentInput, options: Options): Comment
  getNodeComments(filter: CommentInput): [Comment]
}

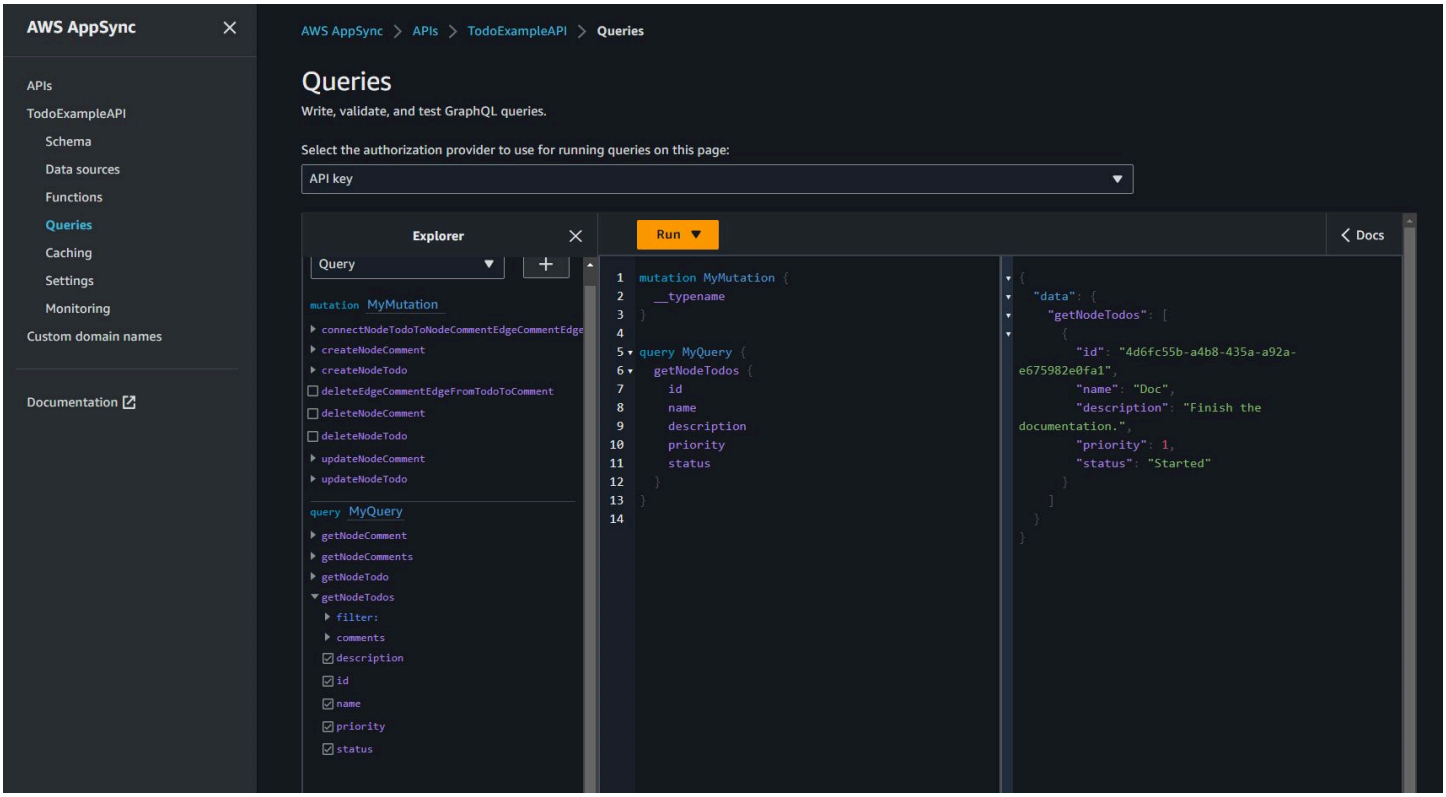
type Mutation {
  createNodeTodo(input: TodoInput!): Todo
  updateNodeTodo(input: TodoInput!): Todo
  deleteNodeTodo(_id: ID!): Boolean
  connectNodeTodoToNodeCommentEdgeCommentEdge(from_id: ID!, to_id: ID!): CommentEdge
  deleteEdgeCommentEdgeFromTodoToComment(from_id: ID!, to_id: ID!): Boolean
  createNodeComment(input: CommentInput!): Comment
  updateNodeComment(input: CommentInput!): Comment
  deleteNodeComment(_id: ID!): Boolean
}

schema {
  query: Query
  mutation: Mutation
}
```

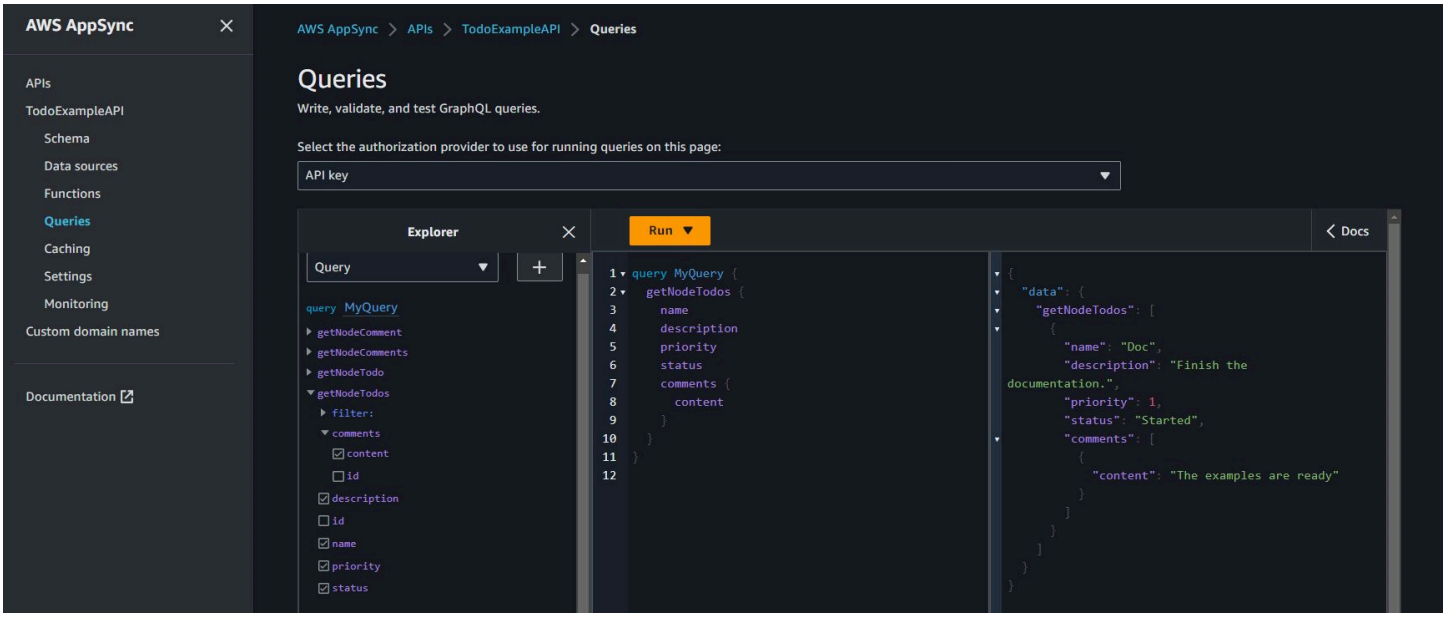
Jetzt können Sie Daten erstellen und abfragen. Hier ist ein Snapshot der AppSync Query-Konsole, die zum Testen der neuen GraphQL-API verwendet wurde, die `TodoExampleAPI` in diesem Fall benannt ist. Im mittleren Fenster zeigt Ihnen der Explorer eine Liste von Abfragen und Mutationen, aus der Sie eine Abfrage, die Eingabeparameter und die Rückgabefelder auswählen können. Dieser Screenshot zeigt die Erstellung eines Todo-Knotentyps unter Verwendung der `createNodeTodo`-Mutation:



Dieser Screenshot zeigt, wie alle Todo-Knoten mithilfe der `getNodeTodos`-Abfrage abgefragt werden:



Nachdem Sie einen Kommentar mit `createNodeComment` erstellt haben, können Sie die `connectNodeTodoToNodeCommentEdgeCommentEdge`-Mutation verwenden, um sie zu verbinden, indem Sie ihre IDs angeben. Hier ist eine verschachtelte Abfrage zum Abrufen von Todos und ihren angehängten Kommentaren:



Wenn Sie Änderungen an der `TodoExample.source.graphql`-Datei vornehmen möchten, wie unter Arbeiten mit Direktiven beschrieben, können Sie das bearbeitete Schema als Eingabe verwenden und das Hilfsprogramm erneut ausführen. Das Hilfsprogramm ändert dann die GraphQL-API entsprechend.

Arbeiten mit Direktiven für ein GraphQL-Schema

Sie können mit einem GraphQL-Schema beginnen, das bereits Direktiven enthält, indem Sie einen Befehl wie den folgenden verwenden:

```
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
```

Sie können Direktiven ändern, die das Hilfsprogramm erstellt hat, oder Ihre eigenen Direktiven zu einem GraphQL-Schema hinzufügen. Hier sind einige Möglichkeiten, mit Direktiven zu arbeiten:

Das Hilfsprogramm so ausführen, dass es keine Mutationen generiert

Um zu verhindern, dass das Hilfsprogramm Mutationen in der GraphQL-API generiert, verwenden Sie die `--output-schema-no-mutations`-Option im Befehl `neptune-for-graphql`.

`Die @alias -Direktive`

Die `@alias`-Direktive kann auf GraphQL-Schematypen oder -Felder angewendet werden. Es ordnet verschiedene Namen zwischen der Graphdatenbank und dem GraphQL-Schema zu. Die Syntax lautet wie folgt:

```
@alias(property: (property name))
```

Im folgenden Beispiel `airport` ist die Graphdatenbankknotenbezeichnung dem GraphQL-Typ `Airport` zugeordnet, und `desc` es ist die Graphknoteneigenschaft, die dem `description`-Feld zugeordnet ist (siehe das Beispiel Air Routes):

```
type Airport @alias(property: "airport") {
  city: String
  description: String @alias(property: "desc")
```

```
}
```

Beachten Sie, dass die standardmäßige GraphQL-Formatierung Pascal-casing Typnamen und Feldnamen in Kamelform erfordert.

Die `@relationship` -Direktive

Die `@relationship`-Direktive ordnet verschachtelte GraphQL-Typen den Edges von Graphdatenbanken zu. Die Syntax lautet wie folgt:

```
@relationship(edgeType: (edge name), direction: (IN or OUT))
```

Hier sehen Sie ein Beispiel für einen Befehl.

```
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)
}
```

`@relationship`-Direktiven finden Sie sowohl im Todo-Beispiel als auch im Air Routes-Beispiel.

Die Direktiven `@graphQuery` und `@cypher`

Sie können openCypher-Abfragen definieren, um einen Feldwert aufzulösen, Abfragen hinzuzufügen oder Mutationen hinzuzufügen. Dadurch wird dem `Airport`-Typ beispielsweise ein neues `outboundRoutesCount`-Feld hinzugefügt, um die ausgehenden Routen zu zählen:

```
type Airport @alias(property: "airport") {
  ...
  outboundRoutesCount: Int @graphQuery(statement: "MATCH (this)-[r:route]->(a) RETURN
 count(r)")
}
```

Hier ein Beispiel für neue Abfragen und Mutationen:

```
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")
}
```

Beachten Sie, dass, wenn Sie `RETURN` weglassen, der Resolver annimmt, dass das Schlüsselwort `this` der zurückkehrende Bereich ist.

Sie können eine Abfrage oder Mutation auch mithilfe einer Gremlin-Abfrage hinzufügen:

```
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
}
```

Derzeit sind Gremlin-Abfragen auf Abfragen beschränkt, die Skalarwerte zurückgeben, oder `elementMap()` für einen einzelnen Knoten oder `elementMap().fold()` für eine Liste von Knoten.

Die `@id -Direktive`

Die `@id`-Direktive identifiziert das Feld, das der `id` Graphdatenbank-Entität zugeordnet ist. Graphdatenbanken wie Amazon Neptune haben immer eine eindeutige `id` für Knoten und Edges, die bei Massenimporten zugewiesen oder automatisch generiert wird. Beispiel:

```
type Airport {
  _id: ID! @id
  city: String
  code: String
}
```

Reservierte Typ-, Abfrage- und Mutationsnamen

Das Hilfsprogramm generiert automatisch Abfragen und Mutationen, um eine funktionierende GraphQL-API zu erstellen. Das Muster dieser Namen wird vom Resolver erkannt und ist reserviert. Hier sind Beispiele für den Typ `Airport` und den Verbindungstyp `Route`:

Der Name `Options` ist reserviert.

```
input Options {
  limit: Int
}
```

Die Funktionsparameter `filter` und `options` sind reserviert.

```
type Query {
  getNodeAirports(filter: AirportInput, options: Options): [Airport]
}
```

Das GetNode-Präfix von Abfragenamen ist reserviert, und Präfixe von Mutationsnamen wie `createNode`, `updateNode`, `deleteNode`, `connectNode`, `deleteNode`, `updateEdge` und `deleteEdge` sind reserviert.

```
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
}
```

Änderungen am GraphQL-Schema anwenden

Sie können das GraphQL-Quellschema ändern und das Hilfsprogramm erneut ausführen, um das neueste Schema aus Ihrer Neptune-Datenbank abzurufen. Jedes Mal, wenn das Hilfsprogramm ein neues Schema in der Datenbank entdeckt, generiert es ein neues GraphQL-Schema.

Sie können das GraphQL-Quellschema auch manuell bearbeiten und das Hilfsprogramm erneut ausführen, indem Sie das Quellschema als Eingabe anstelle des Neptune-Datenbankendpunkts verwenden.

Schließlich können Sie Ihre Änderungen mit diesem JSON-Format in eine Datei einfügen:

```
[
  {
    "type": "(GraphQL type name)",
    "field": "(GraphQL field name)",
    "action": "(remove or add)",
    "value": "(value)"
  }
]
```

Beispiel:

```
[
  {
    "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": ""
  }
```

```
]
```

Wenn Sie dann das Hilfsprogramm mithilfe des `--input-schema-changes-file`-Parameters im Befehl für diese Datei ausführen, wendet das Hilfsprogramm Ihre Änderungen sofort an.

Command-line Argumente für das GraphQL-Hilfsprogramm

- **`--help, -h`** – Gibt den Hilfetext für das GraphQL-Hilfsprogramm an die Konsole zurück.

- **`--input-schema`** ***`(schema text)`*** – Ein GraphQL-Schema, mit oder ohne Direktiven, das als Eingabe verwendet werden soll.

- **`--input-schema-file`** ***`(file URL)`*** – Die URL einer Datei, die ein GraphQL-Schema enthält, das als Eingabe verwendet werden soll.

- **`--input-schema-changes-file`** ***`(file URL)`*** – Die URL einer Datei, die Änderungen enthält, die Sie an einem GraphQL-Schema vornehmen möchten. Wenn Sie das Hilfsprogramm mehrmals für eine Neptune-Datenbank ausführen und auch das GraphQL-Quellschema manuell ändern, indem Sie zum Beispiel eine benutzerdefinierte Abfrage hinzufügen, gehen Ihre manuellen Änderungen verloren. Um dies zu vermeiden, fügen Sie Ihre Änderungen in eine Änderungsdatei ein und übergeben Sie sie mit diesem Argument.

  Die Datei der Änderungen verwendet das folgenden JSON-Format:

  ```
  [
    {
      "type": "(GraphQL type name)",
      "field": "(GraphQL field name)",
      "action": "(remove or add)",
      "value": "(value)"
    }
  ]
  ```

  Weitere Informationen finden Sie unter Beispiele.

- **`--input-graphdb-schema`** ***`(schema text)`*** – Anstatt das Hilfsprogramm für eine Neptune-Datenbank auszuführen, können Sie ein Graphdb-Schema in Textform ausdrücken, um es als Eingabe zu verwenden. Ein Graphdb-Schema hat ein JSON-Format wie dieses:

```
{
  "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)`*** – Anstatt das Hilfsprogramm für eine Neptune-Datenbank auszuführen, können Sie ein Graphdb-Schema in einer Datei speichern, um es als Eingabe zu verwenden. Unter `--input-graphdb-schema` oben finden Sie ein Beispiel für das JSON-Format für eine Graphdb-Schemadatei.

- **`--input-graphdb-schema-neptune-endpoint`** ***`(endpoint URL)`*** – Der Neptune-Datenbank-Endpunkt, aus dem das Hilfsprogramm das Graphdb-Schema extrahieren soll.

- **`--output-schema-file` *`(file name)`*** – Der Name der Ausgabedatei für das GraphQL-Schema. Falls nicht angegeben, ist `output.schema.graphql` die Standardeinstellung, es sei denn, ein Pipeline-Name wurde mit `--create-update-aws-pipeline-name` festgelegt. In diesem Fall lautet der Standarddateiname *`(pipline name)`*`.schema.graphql`.

- **`--output-source-schema-file` *`(file name)`*** – Der Name der Ausgabedatei für das GraphQL-Schema mit Direktiven. Falls nicht angegeben, ist `output.source.schema.graphql` die Standardeinstellung, es sei denn, ein Pipeline-Name wurde mit `--create-update-aws-pipeline-name` festgelegt. In diesem Fall lautet der Standardname *`(pipeline name)`*`.source.schema.graphql`.

- **`--output-schema-no-mutations`** – Wenn dieses Argument vorhanden ist, generiert das Hilfsprogramm keine Mutationen in der GraphQL-API, sondern nur Abfragen.

- **`--output-neptune-schema-file` *`(file name)`*** – Der Name der Ausgabedatei für das Neptune-Graphdb-Schema, das das Hilfsprogramm entdeckt. Falls nicht angegeben, ist `output.graphdb.json` die Standardeinstellung, es sei denn, ein Pipeline-Name wurde mit `--create-update-aws-pipeline-name` festgelegt. In diesem Fall lautet der Standarddateiname *`(pipeline name)`*`.graphdb.json`.

- **`--output-js-resolver-file` *`(file name)`*** – Der Name der Ausgabedatei für eine Kopie des Resolver-Codes. Falls nicht angegeben, ist `output.resolver.graphql.js` die Standardeinstellung, es sei denn, ein Pipeline-Name wurde mit `--create-update-aws-pipeline-name` festgelegt. In diesem Fall lautet der Dateiname *`(pipeline name)`*`.resolver.graphql.js`.

  Diese Datei ist in dem Codepaket komprimiert, das in die Lambda-Funktion hochgeladen wurde, die den Resolver ausführt.

- **`--output-resolver-query-sdk`** – Dieses Argument gibt an, dass die Lambda-Funktion des Dienstprogramms Neptune mithilfe des Neptune-Daten-SDK abfragen soll, das seit der Neptune-Engine-Version 1.2.1.0.R5 verfügbar ist (dies ist die Standardeinstellung). Wenn das Hilfsprogramm jedoch eine ältere Neptune-Engine-Version erkennt, schlägt es vor, stattdessen die HTTPS-Lambda-Option zu verwenden, die Sie mit dem Argument `--output-resolver-query-https` aufrufen können.

- **`--output-resolver-query-https`** – Dieses Argument gibt an, dass die Lambda-Funktion des Dienstprogramms Neptune mithilfe der Neptune-HTTPS-API abfragen soll.

- **`--create-update-aws-pipeline`**— Dieses Argument löst die Erstellung der AWS Ressourcen aus, die die GraphQL-API verwenden soll, einschließlich der AppSync GraphQL-API und des Lambdas, auf dem der Resolver ausgeführt wird.

- **`--create-update-aws-pipeline-name`** ***`(pipeline name)`***— Dieses Argument legt den Namen für die Pipeline fest, wie die `pipeline-name` API für AppSync oder die `pipeline-name` Funktion für die Lambda-Funktion. Wenn kein Name angegeben ist, verwendet `--create-update-aws-pipeline` den Datenbanknamen `Neptune` verwendet.
- **`--create-update-aws-pipeline-region`** ***`(AWS region)`***— Dieses Argument legt die AWS Region fest, in der die Pipeline für die GraphQL-API erstellt wird. Wenn nicht angegeben, ist die Standardregion entweder `us-east-1` oder die Region, in der sich die Neptun-Datenbank befindet, extrahiert aus dem Datenbankendpunkt.
- **`--create-update-aws-pipeline-neptune-endpoint`** ***`(endpoint URL)`*** – Dieses Argument legt den Endpunkt der Neptune-Datenbank fest, der von der Lambda-Funktion für die Datenbankabfrage verwendet wird. Wenn nicht gesetzt, wird der von festgelegte `--input-graphdb-schema-neptune-endpoint`-Endpunkt verwendet.
- **`--remove-aws-pipeline-name`** ***`(pipeline name)`*** – Dieses Argument entfernt eine Pipeline, die mit erstellt wurde`--create-update-aws-pipeline`. Die zu entfernenden Ressourcen sind in einer Datei aufgeführt, die *`(pipeline name)`*`.resources.json` heißt.
- **`--output-aws-pipeline-cdk`**— Dieses Argument löst die Erstellung einer CDK-Datei aus, mit der die AWS Ressourcen für die GraphQL-API erstellt werden können, einschließlich der GraphQL-API und der AppSync Lambda-Funktion, die den Resolver ausführt.

- **`--output-aws-pipeline-cdk-neptume-endpoint` *`(endpoint URL)`*** – Dieses Argument legt den Endpunkt der Neptune-Datenbank fest, der von der Lambda-Funktion für die Neptune-Datenbankabfrage verwendet wird. Wenn nicht gesetzt, wird der von festgelegte `--input-graphdb-schema-neptune-endpoint`-Endpunkt verwendet.
- **`--output-aws-pipeline-cdk-name` *`(pipeline name)`*** — Dieses Argument legt den Pipeline-Namen für die AppSync API und die zu verwendende Lambda-Pipeline-Name-Funktion fest. Wenn nicht angegeben, wird der Neptune-Datenbankname `--create-update-aws-pipeline` verwendet.
- **`--output-aws-pipeline-cdk-region` *`(AWS region)`*** — Dies legt die AWS Region fest, in der die Pipeline für die GraphQL-API erstellt wird. Wenn nicht angegeben, wird standardmäßig `us-east-1` oder die Region verwendet, in der sich die Neptune-Datenbank befindet, extrahiert aus dem Datenbankendpunkt.
- **`--output-aws-pipeline-cdk-file` *`(file name)`*** – Dadurch wird der CDK-Dateiname festgelegt. Wenn nicht festgelegt, ist der Standardwert *`(pipeline name)`*`-cdk.js`.

Nodestream

Nodestream ist ein Framework für den Umgang mit der semantischen Modellierung von Daten als Graph. Es ist so konzipiert, dass es flexibel und erweiterbar ist, sodass Sie definieren können, wie Daten gesammelt und als Diagramm modelliert werden. Es verwendet einen Pipeline-basierten Ansatz, um zu definieren, wie Daten gesammelt und verarbeitet werden, und es bietet eine Möglichkeit, zu definieren, wie das Diagramm aktualisiert werden soll, wenn sich das Schema ändert. All dies erfolgt mithilfe einer einfachen, für Menschen lesbaren Konfigurationsdatei im Yaml-Format. Um dies zu erreichen, verwendet Nodestream eine Reihe von Kernkonzepten, darunter Pipelines, Extraktoren, Transformatoren, Filter, Interpreter, Interpretationen und Migrationen.

Ab Nodestream 0.12 wird Amazon Neptune sowohl für Neptune Database als auch für Neptune Analytics unterstützt.

Einzelheiten zur Konfiguration und Verwendung von Nodestream mit Neptune finden Sie in der Nodestream-Dokumentation: Nodestream-Unterstützung für Amazon Neptune.

Nodestream with Neptune unterstützt derzeit Standard-ETL-Pipelines sowie Time-to-Live-Pipelines (TTL). ETL-Pipelines ermöglichen die Masseneinspeisung von Daten in Neptune aus einer viel breiteren Palette von Datenquellen und Formaten als dies bisher in Neptune möglich war, darunter:

- Software-Stückliste

- Dateien wie CSV, JSON, JSONL, Parquet, TXT und Yaml
- Kafka
- Athena
- REST-APIs

Nodestream unterstützt die IAM-Authentifizierung vollständig, wenn eine Verbindung zu Amazon Neptune hergestellt wird, sofern die Anmeldeinformationen ordnungsgemäß konfiguriert sind. Weitere Informationen zur korrekten Konfiguration von Anmeldeinformationen finden Sie im Leitfaden für boto3-Anmeldeinformationen.

Der TTL-Mechanismus von Nodestream ermöglicht auch neue Funktionen, die bisher in Neptune nicht verfügbar waren. Durch das Kommentieren von aufgenommenen Diagrammelementen mit Zeitstempeln kann Nodestream Pipelines erstellen, die automatisch ablaufen, und Daten entfernen, die eine konfigurierte Lebensdauer überschritten haben.

Neptune-Servicefehler

Amazon Neptune verfügt über zwei verschiedene Gruppen von Fehlern:

- Die Graph-Engine-Fehler, nur für die Neptune-DB-Cluster-Endpunkte.
- Die Fehler, die mit dem APIs beim Erstellen und Ändern von Neptune-Ressourcen mit dem AWS SDK und AWS Command Line Interface ()AWS CLI verbunden sind.

Themen

- Es sind nicht genügend DB-Instances verfügbar
- Graph-Engine-Fehlermeldungen und -codes
- DB-Clusterverwaltung – API-Fehlermeldungen und -codes
- Neptune Loader-Fehler- und Feed-Nachrichten

Es sind nicht genügend DB-Instances verfügbar

Der Fehler `InsufficientDBInstanceCapacity` kann zurückgegeben werden, wenn Sie versuchen, eine DB-Instance zu erstellen, zu starten oder zu ändern. Er kann auch zurückgegeben werden, wenn Sie versuchen, eine DB-Instance aus einem DB-Snapshot wiederherzustellen. Wenn dieser Fehler zurückgegeben wird, besteht die Ursache häufig darin, dass die spezifische DB-Instance-Klasse in der angeforderten Availability Zone nicht verfügbar ist. Sie können folgende Aktionen ausführen, um zu versuchen, das Problem zu beheben:

- Wiederholen Sie die Anforderung mit einer anderen FB-Instance-Klasse.
- Wiederholen Sie die Anforderung mit einer anderen Availability Zone.
- Wiederholen Sie die Anforderung ohne explizite Angabe einer Availability Zone.

Informationen zur Behebung von Problemen mit der Instance-Kapazität für Amazon EC2 finden Sie unter Unzureichende Instance-Kapazität im EC2 Amazon-Benutzerhandbuch.

Graph-Engine-Fehlermeldungen und -codes

Amazon-Neptune-Endpunkte geben die Standardfehler für Gremlin und SPARQL zurück, wenn diese aufgetreten sind.

Von den gleichen Endpunkten können auch für Neptune spezifische Fehler zurückgegeben werden. Dieser Abschnitt dokumentiert Fehlermeldungen, Codes und empfohlene Aktionen für Neptune.

> **Note**
>
> Diese Fehler beziehen sich nur auf die Neptune-DB-Cluster-Endpunkte. Sie APIs dienen zum Erstellen und Ändern von Neptune-Ressourcen mit dem AWS SDK und AWS CLI haben eine andere Reihe häufiger Fehler. Weitere Informationen zu diesen Fehlern finden Sie unter the section called “API-Fehler”.

Graph-Engine-Fehlerformat

Neptune-Fehlermeldungen geben einen relevanten HTTP-Fehlercode und eine JSON-formatierte Antwort zurück.

```
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."
}
```

Graph-Engine-Abfragefehler

Die folgende Tabelle enthält den Fehlercode, die Meldung und den HTTP-Status.

Außerdem wird angegeben, ob die Anforderung wiederholt werden kann. Im Allgemeinen kann die Anforderung wiederholt werden, wenn die Aussicht besteht, dass sie bei einem neuen Versuch erfolgreich ist.

Neptune-Servicefehlercode	HTTP-Status	Ist ein Wiederholungsversuch erlaubt?	Fehlermeldung
`AccessDeniedException`	403	Nein	Bei der Authentifizierung oder Autorisierung ist ein Fehler aufgetreten.
`BadRequestException`	400	Nein	Die Anforderung konnte nicht abgeschlossen werden.
`BadRequestException`	400	Nein	Die angeforderte Größe übersteigt den maximal zulässigen Wert von 157286400 Bytes.
`CancelledByUserException`	500	Ja	Die Verarbeitung der Anforderung wurde von einem autorisierten Client abgebrochen.
`ConcurrentModificationException`	500	Ja	Die Anforderungsverarbeitung war aufgrund eines Änderungskonflikts nicht erfolgreich. Der Client sollte versuchen, die Anfrage erneut zu senden.
`ConstraintViolationException`	400	Ja	Die Abfrage-Engine stellte während der Ausführung der

Neptune-Servicefehlercode	HTTP-Status	Ist ein Wiederholungsversuch erlaubt?	Fehlermeldung
			Anforderung fest, dass der Abschluss einer Operation unmöglich ist, ohne gegen Einschränkungen der Dateninteggrität, wie z. B. Persistenz von Ein- und Ausgangsvertices beim Hinzufügen einer Edge, zu verstoßen. Bedingungen dieser Art werden in der Regel eingehalten, wenn gleichzeitige Änderungen am Diagramm vorgenommen werden, und sind vorübergehend. Der Client sollte versuchen, die Anfrage erneut zu senden.

Neptune-Servicefehlercode	HTTP-Status	Ist ein Wiederholungsversuch erlaubt?	Fehlermeldung
`FailureByQueryException`	500	Ja	Durch das Aufrufen `fail()` schlug die Anforderungsverarbeitung fehl. Der Client kann die Anforderung erneut versuchen, vorausgesetzt, dass die Logik in der Abfrage bei einem späteren Versuch erfolgreich sein wird. Daher ist die Wahl des erneuten Versuchs abfragespezifisch.
`InternalFailureException`	500	Nein	Die Anforderungsverarbeitung ist fehlgeschlagen.
`InvalidNumericDataException`	400	Nein	Ungültige Verwendung numerischer Daten, die nicht in 64-Bit-Speichergröße dargestellt werden können.
`InvalidParameterException`	400	Nein	Für einen Eingabeparameter wurde ein ungültiger out-of-range Wert oder eine ungültige Syntax in einer bereitgestellten RDF-Datei angegeben.

Neptune-Servicefehlercode	HTTP-Status	Ist ein Wiederholungsversuch erlaubt?	Fehlermeldung
`MalformedQueryException`	400	Nein	Die Anforderung wurde abgelehnt, da sie eine Abfrage enthält, die syntaktisch falsch ist oder keine zusätzliche Validierung übergibt.
`MemoryLimitExceededException`	500	Ja	Die Anforderungsverarbeitung war nicht erfolgreich, weil nicht genügend Arbeitsspeicher vorhanden ist. Sie kann aber wiederholt werden, wenn der Server nicht mehr so stark ausgelastet ist.
`MethodNotAllowedException`	405	Nein	Die Anforderung wurde abgelehnt, da die ausgewählte HTTP-Methode vom verwendeten Endpunkt nicht unterstützt wird.
`MissingParameterException`	400	Nein	Ein erforderlicher Parameter für die festgelegte Aktion ist nicht angegeben.

Neptune-Servicefehlercode	HTTP-Status	Ist ein Wiederholungsversuch erlaubt?	Fehlermeldung
`QueryLimitExceededException`	500	Ja	Die Anforderungsverarbeitung war nicht erfolgreich, weil eine begrenzte Ressource fehlte. Sie kann aber wiederholt werden, wenn der Server nicht mehr so stark ausgelastet ist.
`QueryLimitException`	400	Nein	Die Größe der Abfrage übersteigt das Systemlimit.
`QueryTooLargeException`	400	Nein	Die Anforderung wurde abgelehnt, da der Text zu lang ist.
`ReadOnlyViolationException`	400	Nein	Die Anforderung wurde abgelehnt, da sie einige schreibgeschützte Einschränkungen verletzt, wie z. B. eine Bezeichnung eines Replicas als schreibgeschützt.
`ThrottlingException`	500	Ja	Die Anforderungsrate übersteigt den maximalen Durchsatz. OK, um es erneut zu versuchen.

Neptune-Servicefehlercode	HTTP-Status	Ist ein Wiederholungsversuch erlaubt?	Fehlermeldung
`TimeLimitExceededException`	500	Vielleicht	Die Anforderungsverarbeitung hat das Zeitlimit überschritten.
`TooManyRequestsException`	429	Ja	Die Anzahl der Anfragen übersteigt den maximalen Durchsatz. OK, um es erneut zu versuchen.
`UnsupportedOperationException`	400	Nein	Die Anforderung verwendet eine derzeit nicht unterstützte Funktion oder ein Konstrukt.

> **Important**
>
> Ob ein Fehler erneut versucht werden soll oder nicht, ist oft mit wichtigen Nuancen verbunden, die mit der Art des Fehlers, der Arbeitslast und den beobachteten Fehlermustern zusammenhängen. Eine ausführliche Anleitung zur Entwicklung einer praktischen Wiederholungsstrategie finden Sie unter. the section called “Ausnahmen und Wiederholungen”

IAM-Authentifizierungsfehler

Diese Fehler gelten speziell für Cluster, bei denen die IAM-Authentifizierung aktiviert ist.

Die folgende Tabelle enthält den Fehlercode, die Meldung und den HTTP-Status.

Neptune-Servicefehlercode	HTTP-Status	Fehlermeldung
Falsche(r) IAM-Benutzer/Richtlinie	403	Sie haben keinen ausreichenden Zugriff zum Durchführen dieser Aktion.
Falsche oder fehlende Region	403	Die Anmeldeinformationen sollten auf eine gültige Region beschränkt sein, nicht auf ". *region*
Falscher oder fehlender Service-Name	403	Die Anmeldeinformationen müssen sich auf den korrekten Service beziehen: „`neptune-db` “.
Falscher oder fehlender Host-Header/Ungültige Signatur	403	Die berechnete Anforderungssignatur entspricht nicht der angegebenen Signatur. Überprüfen Sie Ihren AWS geheimen Zugriffsschlüssel und Ihre Signaturmethode. Weitergehende Informationen finden Sie in der Service-Dokumentation. Der Host-Header fehlt oder ist falsch.
`X-Amz-Security-Token` fehlt	403	'`x-amz-security-token` ' wird als `SignedHeader` genannt, ist in der HTTP-Anforderung aber nicht vorhanden
Fehlender Autorisierungsheader	403	Die Anforderung schloss die erforderliche Autorisierungskopfzeile nicht ein oder sie war falsch formatiert.

Neptune-Servicefehlercode	HTTP-Status	Fehlermeldung
Fehlendes Authentifizierungstoken	403	Fehlendes Authentifizierungstoken.
Altes Datum	403	Die Signatur *20181011T213907Z* ist abgelaufen: ist jetzt vor *20181011T213915Z* (*20181011T214415Z -5 min*.)
Datum in der Zukunft	403	Signatur noch nicht aktuell: *20500224T213559Z* ist immer noch später als *20181108T225925Z* (*20181108T225425Z +5 min*.)
Ungültiges Datumsformat	403	Das Datum muss im „Basisformat“ nach ISO-8601 angegeben werden. Habe '*date*'. Siehe https://en.wikipedia.org/wiki/ISO_8601.
Unbekannter/Fehlender Zugriffsschlüssel oder Sitzungs-Token	403	Das Sicherheits-Token der Anfrage ist ungültig.
Unbekannter/Fehlender Geheimnisschlüssel	403	Die berechnete Anforderungssignatur entspricht nicht der angegebenen Signatur. Überprüfen Sie Ihren AWS geheimen Zugriffsschlüssel und Ihre Signaturmethode. Weitergehende Informationen finden Sie in der Service-Dokumentation. Der Host-Header fehlt oder ist falsch.

Neptune-Servicefehlercode	HTTP-Status	Fehlermeldung
`TooManyRequestsException`	429	Die Rate der Anfragen übersteigt den maximalen Durchsatz. OK, um es erneut zu versuchen.

DB-Clusterverwaltung – API-Fehlermeldungen und -codes

Diese Amazon Neptune Neptune-Fehler stehen im Zusammenhang mit dem APIs für das Erstellen und Ändern von Neptune-Ressourcen mit dem SDK und. AWS AWS CLI

Die folgende Tabelle enthält den Fehlercode, die Meldung und den HTTP-Status.

Neptune-Servicefehlercode	HTTP-Status	Fehlermeldung
`AccessDeniedException`	403	Sie haben keinen ausreichenden Zugriff zum Durchführen dieser Aktion.
`IncompleteSignature`	400	Die Signatur der Anfrage entspricht nicht den Standards. AWS
`InternalFailure`	500	Die Anforderungsverarbeitung ist fehlgeschlagen, da ein unbekannter Fehler, eine Ausnahme oder ein Fehler aufgetreten ist.
`InvalidAction`	400	Die angeforderte Aktion oder Operation ist ungültig. Überprüfen Sie, ob die Aktion ordnungsgemäß eingegeben wurde.
`InvalidClientTokenId`	403	Das angegebene X.509-Zertifikat oder die angegebene

Neptune-Servicefehlercode	HTTP-Status	Fehlermeldung
		AWS Zugriffsschlüssel-ID ist in unseren Aufzeichnungen nicht vorhanden.
`InvalidParameterCombination`	400	Parameter, die nicht gemeinsam verwendet werden dürfen, wurden gemeinsam verwendet.
`InvalidParameterValue`	400	Für den Eingabeparameter wurde ein ungültiger out-of-range Wert oder angegeben.
`InvalidQueryParameter`	400	Für den Eingabeparameter wurde ein ungültiger out-of-range Wert oder angegeben.
`MalformedQueryString`	400	Die Abfragezeichenfolge enthält einen Syntaxfehler.
`MissingAction`	400	In der Anforderung fehlt ein Aktions- oder ein erforderlicher Parameter.
`MissingAuthenticationToken`	403	Die Anfrage muss entweder eine gültige (registrierte) AWS Zugriffsschlüssel-ID oder ein X.509-Zertifikat enthalten.
`MissingParameter`	400	Ein erforderlicher Parameter für die festgelegte Aktion ist nicht angegeben.
`OptInRequired`	403	Für die AWS Zugriffsschlüssel-ID ist ein Abonnement für den Dienst erforderlich.

Neptune-Servicefehlercode	HTTP-Status	Fehlermeldung
`RequestExpired`	400	Die Anfrage erreichte den Service mehr als 15 Minuten nach dem Datumsstempel auf der Anfrage oder mehr als 15 Minuten nach dem Ablaufdat um der Anfrage (z. B. bei vorsignierter Anfrage URLs), oder der Datumsstempel auf der Anfrage liegt mehr als 15 Minuten in der future.
`ServiceUnavailable`	503	Die Anforderung ist aufgrund eines temporären Fehlers des Servers fehlgeschlagen.
`ThrottlingException`	500	Die Anforderung wurde aufgrund der Drosselung von Anforderungen abgelehnt.
`ValidationError`	400	Die Eingabe erfüllt nicht die von einem AWS Dienst angegebenen Einschrän kungen.

Neptune Loader-Fehler- und Feed-Nachrichten

Die folgenden Nachrichten werden vom `status`-Endpunkt des Neptune-Loaders ausgegeben. Weitere Informationen finden Sie unter Get-Status API.

Die folgende Tabelle zeigt den Loader Feed-Code und die zugehörige Beschreibung.

Fehler- oder Feed-Code	Beschreibung
LOAD_NOT_STARTED	Der Ladevorgang wurde aufgezeichnet, aber nicht gestartet.

Fehler- oder Feed-Code	Beschreibung
LOAD_IN_PROGRESS	Zeigt an, dass der Ladevorgang läuft, und gibt die Anzahl der Dateien an, die gerade geladen werden. Wenn der Loader eine Datei analysiert, erstellt er einen oder mehrere Blöcke, die parallel geladen werden. Da eine einzelne Datei mehrere Blöcke erzeugen kann, ist die Anzahl der in dieser Nachricht enthaltenen Dateien normalerweise geringer als die Anzahl der Threads, die beim Massenladevorgang verwendet werden.
LOAD_COMPLETED	Der Ladevorgang wurde ohne Fehler oder mit Fehlern innerhalb eines akzeptablen Bereichs abgeschlossen.
LOAD_CANCELLED_BY_USER	Der Ladevorgang wurde vom Benutzer abgebrochen.
LOAD_CANCELLED_DUE_TO_ERRORS	Der Ladevorgang wurde vom System aufgrund von Fehlern abgebrochen.
LOAD_UNEXPECTED_ERROR	Der Ladevorgang schlug aufgrund eines unerwarteten Fehlers fehl.
LOAD_FAILED	Load aufgrund einer oder mehrerer Fehler fehlgeschlagen.
LOAD_S3_READ_ERROR	Feed ist aufgrund zeitweiliger oder vorüberge hender Amazon S3-Konnektivitätsprobleme fehlgeschlagen. Wenn einer der Feeds diesen Fehler erhält, wird der gesamte Ladestatus auf LOAD_FAILED festgelegt.

Fehler- oder Feed-Code	Beschreibung
LOAD_S3_ACCESS_DENIED_ERROR	Der Zugriff auf den S3-Bucket wurde abgelehnt . Wenn einer der Feeds diesen Fehler erhält, wird der gesamte Ladestatus auf LOAD_FAIL ED festgelegt.
LOAD_COMMITTED_W_WRITE_CONFLICTS	Für die geladenen Daten wurde ein Commit mit ungelösten Schreibkonflikten durchgeführt. Der Loader versucht, die Schreibkonflikte in separaten Transaktionen aufzulösen und den Feed-Status mit fortschreitendem Ladevorga ng zu aktualisieren. Wenn der endgültige Feed-Status LOAD_COMMITTED_W_WRITE_CONF LICTS lautet, versuchen Sie, den Ladevorga ng fortzusetzen. Er wird wahrscheinlich ohne Konflikte erfolgreich ausgeführt. Ein Schreibko nflikt tritt in der Regel nicht im Zusammenhang mit ungültigen Eingabedaten auf, aber doppelte Daten erhöhen die Wahrscheinlichkeit von Schreibkonflikten.
LOAD_DATA_DEADLOCK	Der Ladevorgang wurde aufgrund eines Deadlocks automatisch zurückgesetzt.
LOAD_DATA_FAILED_DUE_TO_FEE D_MODIFIED_OR_DELETED	Feed ist fehlgeschlagen, weil die Datei nach dem Start der Ladung gelöscht oder aktualisiert wurde.
LOAD_FAILED_BECAUSE_DEPENDE NCY_NOT_SATISFIED	Die Ladeanforderung wurde nicht ausgeführt, da ihre Abhängigkeitsprüfung fehlgeschlagen ist.
LOAD_IN_QUEUE	Die Ladeanforderung wurde in die Warteschl ange gestellt und wartet darauf, ausgeführt zu werden.

Fehler- oder Feed-Code	Beschreibung
LOAD_FAILED_INVALID_REQUEST	Das Laden ist fehlgeschlagen, weil die Anforderung ungültig war (z. B. ist die angegebene Datei source/bucket möglicher weise nicht vorhanden oder das Dateiformat ist ungültig).

Engine-Versionen für Amazon Neptune

Amazon Neptune veröffentlicht regelmäßig Updates der Engine.

Mit der Instance-Status-API können Sie ermitteln, welche Engine-Version Sie derzeit installiert haben. Die Versionsnummer gibt an, ob Sie eine ursprüngliche Hauptversion, eine Nebenversion oder eine Patch-Version ausführen. Weitere Informationen zur Nummerierung der Versionen finden Sie unter Engine-Versionsnummern.

Weitere allgemeine Informationen zu Updates finden Sie unter Clusterwartung.

Ab der Engine-Version 1.3.0.0 haben die Engine-Versionen die in der folgenden Tabelle dargestellte Struktur. Bei der Nebenversionsnummer handelt es sich um die Nummer, die für die `AutoMinorVersionUpgrade`-Verarbeitung ausgewertet wird.

Version	Prod rsion	Haup ion	Unterversion	Patch-Version	Status	Freigegeben	Ende der Lebensdauer	Aktualisieren zu:
1.4.7.0	1	4	7	0	aktiv	2026-03-03	2027-06-03	–
1.4.6.3	1	4	6	3	aktiv	18.12.2025	2027-06-03	1.4.7.0
1.4.6.2	1	4	6	2	aktiv	18.11.2025	2027-06-03	1.4.7.0
1.4.6.1	1	4	6	1	aktiv	18.09.2025	2027-03-06	1.4.6.2
1.4.6.0	1	4	6	0	aktiv	2025-09-02	2027-03-06	1.4.6.1
1.4.5.1	1	4	5	1	aktiv	30.06.2025	2027-03-06	1.4.6.0
1.4.5.0	1	4	5	0	aktiv	09.04.2025	2027-03-06	1.4.5.1

Version	Prod rsion	Haup ion	Unterversion	Patch-Version	Status	Freigegeben	Ende der Lebensdauer	Aktualisieren zu:
1.4.4.0	1	4	4	0	aktiv	24.02.2025	2027-03-06	1.4.5.0
1.4.3.0	1	4	3	0	aktiv	2025-01-21	2027-03-06	1.4.4.0
1.4.2.0	1	4	2	0	aktiv	2024-12-19	2027-03-06	1.4.3.0
1.4.1.0	1	4	1	0	aktiv	21. November 2024-	2027-03-06	1.4.2.0
1.4.0.0	1	4	0	0	aktiv	2024-11-06	2027-03-06	1.4.1.0
1.3.4.0	1	3	4	0	aktiv	2024-10-01	2027-03-06	1.4.0.0
1.3.3.0	1	3	3	0	aktiv	2024-08-05	2027-03-06	1.3.4.0
1.3.2.1	1	3	2	1	aktiv	20.06.2024	2027-03-06	1.3.3.0
1.3.2.0	1	3	2	0	aktiv	2024-06-10	2027-03-06	1.3.2.1
1.3.1.0	1	3	1	0	aktiv	2024-03-06	2027-03-06	1.3.2.1
1.3.0.0	1	3	0	0	aktiv	15.11.2023-23	2027-03-06	1.3.2.1

In der folgenden Tabelle sind alle Engine-Versionen seit 1.0.1.0 zusammen mit Informationen zur Version aufgeführt. end-of-life Sie können die Daten in dieser Tabelle verwenden, um Ihre Test- und Upgrade-Zyklen zu planen.

Version	Hauptv ion	Unterve ion	Status	Freigegeben	Ende der Lebensdauer	Aktualisieren zu:
1.2.1.2	1.2	1.2	aktiv	2024-08-05	2026-06-30	1.3.0.0
1.2.1.1	1.2	1.1	aktiv	2024-03-11	2026-06-30	1.3.0.0
1.2.1.0	1.2	1,0	aktiv	2023-03-08	2026-06-30	1.3.0.0
1.2.0.2	1.2	0.2	aktiv	16.11.2022	2026-06-30	1.3.0.0
1.2.0.1	1.2	0.1	aktiv	26.10.2022	2026-06-30	1.3.0.0
1.2.0.0	1.2	0.0	aktiv	21.07.2022	2026-06-30	1.3.0.0
1.1.1.0	1.1	1,0	aktiv	19.04.2022	2026-06-30	1.2.1.0
1.1.0.0	1.1	0.0	veraltet	19.11.2021	15.03.2025	1.1.1.0
1.0.5.1	1,0	5.1	veraltet	01.10.2021	2023-01-30	1.1.0.0
1.0.5.0	1,0	5.0	veraltet	27.07.2021	2023-01-30	1.1.0.0
1.0.4.2	1,0	4.2	veraltet	01.06.2021	2023-01-30	1.1.0.0
1.0.4.1	1,0	4.1	veraltet	08.12.2020	2023-01-30	1.1.0.0
1.0.4.0	1,0	4,0	veraltet	10.10.2020	2023-01-30	1.1.0.0
1.0.3.0	1,0	3.0	veraltet	03.08.2020	2023-01-30	1.1.0.0
1.0.2.2	1,0	2.2	veraltet	09.03.2020	29.07.2022	1.0.3.0
1.0.2.1	1,0	2.1	veraltet	22.11.2019	29.07.2022	1.0.3.0
1.0.2.0	1,0	2.0	veraltet	08.11.2019	19.05.2020	1.0.3.0

Version	Hauptv ion	Unterve ion	Status	Freigegeben	Ende der Lebensdauer	Aktualisieren zu:
1.0.1.2	1,0	1.2	veraltet	2019-10-15	—	—
1.0.1.1	1,0	1.1	veraltet	13.08.2019	—	—
1.0.1.0.*	1,0	1.0.*	veraltet	07.02.2019 und davor	—	—

Planung der wichtigsten Engine-Versionen end-of-life

Neptun-Engine-Versionen erreichen fast immer am Ende eines Kalenderquartals das Ende ihres Lebenszyklus. Ausnahmen treten nur auf, wenn wichtige Sicherheits- oder Verfügbarkeitsprobleme auftreten.

Wenn eine Engine-Version das Ende ihres Lebenszyklus de erreicht, müssen Sie Ihre Neptune-Datenbank auf eine neuere Version aktualisieren.

Im Allgemeinen sind Neptune-Engine-Versionen weiterhin wie folgt verfügbar:

- Engine-Nebenversionen: Engine-Nebenversionen bleiben nach ihrer Veröffentlichung mindestens 6 Monate lang verfügbar.
- Engine-Hauptversionen: Engine-Hauptversionen bleiben nach ihrer Veröffentlichung mindestens 12 Monate lang verfügbar.

Mindestens 3 Monate, bevor eine Engine-Version ihr Lebensende erreicht, AWS sendet eine automatische E-Mail-Benachrichtigung an die mit Ihrem AWS Konto verknüpfte E-Mail-Adresse und veröffentlicht dieselbe Nachricht in Ihrem AWS Health Dashboard. So haben Sie Zeit, das Upgrade zu planen und vorzubereiten.

Wenn eine Engine-Version das Ende ihres Lebenszyklus erreicht, können Sie mit dieser Version keine neuen Cluster oder Instances mehr erstellen und auch Autoscaling kann keine Instances mit dieser Version erstellen.

Eine Engine-Version, die das Ende ihres Lebenszyklus erreicht hat, wird während eines Wartungszeitfensters automatisch aktualisiert. Die Nachricht, die Sie 3 Monate vor dem Ende des Lebenszyklus der Engine-Version erhalten, enthält Informationen darüber, was dieses automatische

Update beinhalten würde, einschließlich der Version, auf die Sie automatisch aktualisiert werden würden, der Auswirkungen auf Ihre DB-Cluster und der von uns empfohlenen Maßnahmen.

> **Important**
>
> Sie sind dafür verantwortlich, Ihre Datenbank-Engine-Versionen auf dem neuesten Stand zu halten. AWS fordert alle Kunden nachdrücklich auf, ihre Datenbanken auf die jeweils neueste Engine-Version zu aktualisieren, um von den aktuellsten Sicherheits-, Datenschutz- und Verfügbarkeitsvorkehrungen zu profitieren. Wenn Sie Ihre Datenbank nach dem Verfallsdatum auf einer Engine oder Software betreiben, die nicht mehr unterstützt wird („Legacy Engine“), ist die Wahrscheinlichkeit von Sicherheits-, Datenschutz- und Betriebsrisiken, einschließlich Ausfallzeiten, höher.
> Der Betrieb Ihrer Datenbank auf einer beliebigen Engine unterliegt der Vereinbarung, die Ihre Nutzung der AWS Dienste regelt. Ältere Engines sind nicht allgemein verfügbar. AWS bietet keinen Support mehr für die Legacy Engine an und AWS kann den Zugriff auf oder die Nutzung einer Legacy Engine jederzeit einschränken, wenn AWS festgestellt wird, dass die Legacy Engine ein Sicherheits- oder Haftungsrisiko oder ein Schadensrisiko für die Dienste AWS, ihre verbundenen Unternehmen oder Dritte darstellt. Ihre Entscheidung, Ihre Inhalte weiterhin in einer Legacy Engine auszuführen, könnte dazu führen, dass Ihre Inhalte nicht mehr verfügbar, beschädigt oder nicht wiederherstellbar sind. Für Datenbanken, die auf einer Legacy Engine ausgeführt werden, gelten Ausnahmen für Service Level Agreements (SLA). DATENBANKEN UND ZUGEHÖRIGE SOFTWARE, DIE AUF EINER ÄLTEREN ENGINE AUSGEFÜHRT WERDEN, ENTHALTEN BUGS, FEHLER, DEFEKTE UND AND/OR SCHÄDLICHE KOMPONENTEN. DEMENTSPRECHEND UND UNGEACHTET ANDERSLAUTENDER BESTIMMUNGEN IN DER VEREINBARUNG ODER DEN SERVICEBEDINGUNGEN AWS WIRD DIE ÄLTERE ENGINE „WIE BESEHEN“ BEREITGESTELLT.

Amazon Neptune Engine, Version 1.4.7.0 (03.03.2020)

Ab 2026-03-03 wird die Engine-Version 1.4.7.0 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

Neue Features in dieser Engine-Version

- OpenCypher liest aus der S3-Unterstützung für Parquet- und CSV-Dateien über OC. Weitere Informationen finden Sie in der Dokumentation. the section called “neptune.read ()”

- OpenCypher-Funktionen zur Geodatenabfrage. Diese Version enthält 12 Funktionen für räumliche Typen, die auf dem Standard ISO/IEC 13249-3:2016 basieren, einen neuen Geometrie-Eigenschaftstyp für POINT, der in einem neuen geopetrischen Index für einen schnellen Abruf gespeichert ist, und Unterstützung für das Textformat (WKT). Well-Known Räumliche DatenGeofunktionenWeitere Informationen finden Sie in der und in der Dokumentation.

Verbesserungen in dieser Engine-Version

- Verbesserte Abfrageleistung für SPARQL-Unterabfragen, die kleine Ergebnismengen zurückgeben, einschließlich solcher mit kleinen LIMIT-Werten
- Verbesserte Abfrageleistung in Fällen, in denen Variablen durch eine sehr große Anzahl konstanter Werte eingeschränkt sind (z. B. durch eine SPARQL VALUES-Klausel oder eine UNWIND-Klausel) OpenCypher
- Verbesserungen bei Einfügeabfragen mit niedriger Latenz durch einige Optimierungen bei Wörterbucheinfügungen
- Der DFE-Engine wurden neue Gremlin-Sprachschritte hinzugefügt (siehe Erläuterung der Gremlin-Schritte in DFE).
  - Pfad- und Durchquerungsschritte: `order(local)`
  - Schritte zum Aggregieren und Sammeln: `dedup(local)`
- Leistungsverbesserung für OpenCypher Abfragen, einschließlich`COLLECT(DISTINCT ...)`. Das unter REWRITING COLLECT (DISTINCT...) -Abfragen beschriebene Umschreiben ist nicht mehr erforderlich, wenn Sie die Engine-Version 1.4.7.0 oder höher verwenden.

In diesem Engine-Version behobene Fehler

Allgemeine Korrekturen:

- Das Problem, dass beim Laden einer großen Anzahl von Edge-Dateien das Laden von Massenladungen nicht mehr reagierte, wurde behoben
- Behebt ein globales Datenbankcluster-Patching-Problem, das sekundäre Cluster-Updates der Versionen 1.4.0.0, 1.4.1.0 und 1.4.2.0 betraf.

Query-Language In dieser Version unterstützte Versionen

Stellen Sie vor dem Upgrade eines DB-Clusters auf Version 1.4.7.0 sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprachen kompatibel ist:

- Die älteste unterstützte Version von Gremlin: `3.7.1`
- Die neueste unterstützte Version von Gremlin: `3.7.1`
- openCypher-Version: `Neptune-9.0.20190305-1.0`
- SPARQL-Version: `1.1`

Upgrade-Pfade auf Engine-Version 1.4.7.0

Sie können von der Engine-Version 1.2.0.0 oder höher auf diese Version aktualisieren.

Upgrade auf diesen Release

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.4.7.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.4.7.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Statt `--apply-immediately` können Sie `--no-apply-immediately` angeben. Für die Durchführung eines Hauptversions-Upgrades ist der allow-major-version-upgrade-Parameter

erforderlich. Stellen Sie außerdem sicher, dass Sie die Engine-Version angeben, da Ihre Engine sonst möglicherweise auf eine andere Version aktualisiert wird.

Wenn Ihr Cluster eine benutzerdefinierte Cluster-Parametergruppe verwendet, müssen Sie diesen Parameter einschließen, um ihn zu anzugeben:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Ebenso sollte für Instances im Cluster, die eine benutzerdefinierte DB-Parametergruppe verwenden, dieser Parameter eingeschlossen werden, um ihn zu spezifizieren:

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
>   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.
> ```
>
> Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine, Version 1.4.6.3 (18.12.2025)

Ab dem 18.12.2025 wird die Engine-Version 1.4.6.3 allgemein eingesetzt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

In dieser Engine-Version (1.4.6.3) behobene Fehler

Allgemeine Korrekturen

- Einige Ausnahmen, die fälschlicherweise als Interner Serverfehler eingestuft wurden, werden jetzt während der Ausführung der Data Flow Engine (DFE) korrekt als Ausnahmen wegen unzureichenden Speichers gemeldet.

- Es wurde ein Problem behoben, das dazu führen konnte, dass Neptune Database-Instances beim Start fehlschlugen, wenn die Inline-Edge-ID-Validierung auf unerwartete Wörterbucheintragstypen stieß. Die Inline-Edge-ID-Validierung wird jetzt nur für Konfigurationen der Gremlin-Abfrage-Engine ausgeführt und verarbeitet Wörterbucheinträge ohne URI problemlos.
- Es wurde ein Problem behoben, bei dem der Neptune-Bulk-Loader in bestimmten Regionen aufgrund einer falschen Endpunktauflösung keine Verbindung zu S3 herstellen konnte.

OpenCypher-Korrekturen

- Es wurde ein Engine-Absturz behoben, der manchmal bei Abfragen mit auftrat. `CALL`
- Der Timeout und die Behandlung von Stornierungen bei Mutationsabfragen wurden behoben.
- Es wurde ein Fehler behoben, bei dem die `MERGE` Klausel falsche Ergebnisse lieferte, wenn einer der übergebenen Eigenschaftswerte einen Nullwert `MERGE` hatte.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.4.6.3 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit diesen Versionen in Abfragesprachen kompatibel ist:

- Die älteste unterstützte Version von Gremlin: `3.7.1`
- Die neueste unterstützte Version von Gremlin: `3.7.1`
- openCypher-Version: `Neptune-9.0.20190305-1.0`
- SPARQL-Version: `1.1`

Upgrade-Pfade auf Engine-Version 1.4.6.3

Sie können von der Engine-Version 1.2.0.0 oder höher auf diese Version aktualisieren.

Upgrade auf diesen Release

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.4.6.3 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.4.6.3 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Statt `--apply-immediately` können Sie `--no-apply-immediately` angeben. Um ein Upgrade einer Hauptversion durchzuführen, ist der allow-major-version-upgrade Parameter erforderlich. Stellen Sie außerdem sicher, dass Sie die Engine-Version angeben, da Ihre Engine sonst möglicherweise auf eine andere Version aktualisiert wird.

Wenn Ihr Cluster eine benutzerdefinierte Cluster-Parametergruppe verwendet, müssen Sie diesen Parameter einschließen, um ihn zu anzugeben:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Ebenso sollte für Instances im Cluster, die eine benutzerdefinierte DB-Parametergruppe verwenden, dieser Parameter eingeschlossen werden, um ihn zu spezifizieren:

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
> 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.
> ```
>
> Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine, Version 1.4.6.2 (18.11.2025)

Ab dem 18.11.2025 wird die Engine-Version 1.4.6.2 allgemein eingesetzt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

In dieser Engine-Version wurden Fehler behoben

Allgemeine Korrekturen

- Einige Ausnahmen, die fälschlicherweise als Internal Failure Exceptions (IFEs) klassifiziert wurden, werden jetzt während der Ausführung der Data Flow Engine (DFECMEs) korrekt als Concurrent Modification Exceptions () gemeldet.

Korrekturen für Gremlin

- Die Engine-Stabilität für Gremlin gegenüber DFE wurde verbessert

OpenCypher-Korrekturen

- Es wurde ein Problem behoben, das in der Neptune-Engine-Version 1.4.6.0 eingeführt wurde, bei dem OpenCypher MERGE-Abfragen eingehende Beziehungsrichtungen falsch behandelten (←). Abfragen, die Muster wie MERGE (n) ← [:type] - (m) verwenden, erstellen jetzt Beziehungen mit der richtigen Richtung.
- Ein fehlgeschlagener Commit-Vorgang bei den Writer-Transaktionen, die nicht ordnungsgemäß zurückgesetzt wurden, wurde behoben.
- Optimierte Abfragepläne werden jetzt für MERGE-Abfragen unterstützt, die auf parametrisierte Werte verweisen.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.4.6.2 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit diesen Versionen in Abfragesprachen kompatibel ist:

- Die älteste unterstützte Version von Gremlin: `3.7.1`
- Die neueste unterstützte Version von Gremlin: `3.7.1`
- openCypher-Version: `Neptune-9.0.20190305-1.0`
- SPARQL-Version: `1.1`

Upgrade-Pfade auf Engine-Version 1.4.6.2

Sie können von der Engine-Version 1.2.0.0 oder höher auf diese Version aktualisieren.

Upgrade auf diesen Release

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.4.6.2 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.4.6.2 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Statt `--apply-immediately` können Sie `--no-apply-immediately` angeben. Um ein Hauptversions-Upgrade durchzuführen, ist der allow-major-version-upgrade Parameter erforderlich. Stellen Sie außerdem sicher, dass Sie die Engine-Version angeben, da Ihre Engine sonst möglicherweise auf eine andere Version aktualisiert wird.

Wenn Ihr Cluster eine benutzerdefinierte Cluster-Parametergruppe verwendet, müssen Sie diesen Parameter einschließen, um ihn zu anzugeben:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Ebenso sollte für Instances im Cluster, die eine benutzerdefinierte DB-Parametergruppe verwenden, dieser Parameter eingeschlossen werden, um ihn zu spezifizieren:

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
>   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.
> ```
>
> Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWSPremium-Support zur Verfügung.

Amazon Neptune Engine Version 1.4.6.1 (18.09.2025)

Ab dem 18.09.2025 wird die Engine-Version 1.4.6.1 allgemein eingesetzt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

In dieser Engine-Version wurden Fehler behoben

Allgemeine Korrekturen

- Die in Version 1 eingeführten Netzwerkprüfungen für Cluster, die private IP-Bereiche verwenden, die nicht RFC 1918-konform sind, wurden entfernt. Amazon Neptune Engine, Version 1.4.6.0 (02.09.2025)

Korrekturen für Gremlin

- Das Problem mit der Websocket-Verbindungsverwaltung bei Transaktionen wurde behoben.
- Ein seltenes Problem beim Neustart von Instanzen im Gremlin DFE-Modus wurde behoben

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.4.6.1 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit diesen Versionen in Abfragesprachen kompatibel ist:

- Die älteste unterstützte Version von Gremlin: `3.7.1`
- Die neueste unterstützte Version von Gremlin: `3.7.1`
- openCypher-Version: `Neptune-9.0.20190305-1.0`
- SPARQL-Version: `1.1`

Upgrade-Pfade auf Engine-Version 1.4.6.1

Sie können von der Engine-Version 1.2.0.0 oder höher auf diese Version aktualisieren.

Upgrade auf diesen Release

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.4.6.1 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.4.6.1 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Statt `--apply-immediately` können Sie `--no-apply-immediately` angeben. Um ein Hauptversions-Upgrade durchzuführen, ist der allow-major-version-upgrade Parameter erforderlich.

Stellen Sie außerdem sicher, dass Sie die Engine-Version angeben, da Ihre Engine sonst möglicherweise auf eine andere Version aktualisiert wird.

Wenn Ihr Cluster eine benutzerdefinierte Cluster-Parametergruppe verwendet, müssen Sie diesen Parameter einschließen, um ihn zu anzugeben:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Ebenso sollte für Instances im Cluster, die eine benutzerdefinierte DB-Parametergruppe verwenden, dieser Parameter eingeschlossen werden, um ihn zu spezifizieren:

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
>   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.
> ```
>
> Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine, Version 1.4.6.0 (02.09.2025)

Ab 2025-09-02 wird die Engine-Version 1.4.6.0 allgemein eingesetzt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

> **Warning**
>
> Die Engine-Version 1.4.6.0 beinhaltet neue Netzwerkprüfungen für Cluster, die nicht RFC 1918-konforme private IP-Bereiche für Datenbank-VPC ohne IAM-Authentifizierung verwenden. Wenn Sie über diese VPC- und IAM-Konfiguration verfügen, müssen Sie Ihre Datenbank-VPC so aktualisieren, dass private IP-Bereiche nach RFC 1918 verwendet

werden. and/or Aktivieren Sie die IAM-Authentifizierung, um Fehler bei Abfragen nach dem Upgrade auf 1.4.6.0 zu vermeiden.

Neue Funktionen in dieser Engine-Version

- Stellen Sie über öffentliche Endpunkte eine Connect zu Neptune her. Weitere Informationen finden Sie unter Öffentliche Endpunkte von Neptune.

Verbesserungen in dieser Engine-Version

Allgemeine Verbesserungen

- Verbesserte SPARQL-Leistung für Aktualisierungsvorgänge.
- Verbesserte OpenCypher Leistung für `CREATE`, `MERGE`, und `SET` (Mutationen) -Operationen.
- Verbesserte OpenCypher Leistung für CALL Subquery-Operationen.

Verbesserungen bei OpenCypher

- Es wurde ein neuer Abfragehinweis hinzugefügt, um das Timeout auf Abfrageebene zu unterstützen.

In dieser Engine-Version wurden Fehler behoben

Korrekturen für Gremlin

- Verbindungen zu Gremlin-Sitzungen müssen auf demselben Kanal erfolgen, auf dem sie erstellt wurden. Das bedeutet, dass es nicht möglich ist, mehrere Client-Instanzen mit derselben Sitzung zu verbinden.
- Gremlin-Sitzungen wurden immer geschlossen, wenn der Client geschlossen wurde, aber sie werden jetzt auch geschlossen, wenn ein Server die Verbindung beendet hat, wodurch eine unbeabsichtigte oder unerwartete Wiederverbindung verhindert wird.
- Speicherlecks bei Gremlin-Abfragen, die große Daten vom Typ Blob lesen, wurden behoben.

OpenCypher-Korrekturen

- Die Variablenbehandlung nach einer Unterabfrage wurde behoben. `CALL`
- Es wurde ein Problem mit der `reduce` Funktion zur korrekten Behandlung von arithmetischen Überlaufszenarien behoben.
- Es wurde ein Speicherverlust behoben, der sich auf parametrisierte Abfragen auswirkte, wenn der Abfrageplan-Cache aktiviert war.
- Es wurde ein Problem mit Klauseln behoben, die in komplexen `NOT EXISTS` `WHERE` Klauseln verwendet wurden.
- Es wurde ein Fehler behoben, bei dem Cuncurrent Memory Exception (CMEs) fälschlicherweise als gemeldet wurde. BadRequestException

Fehlerkorrekturen für SPARQL

- Es wurde eine Fehlermeldung für SPARQL behoben, `LOAD/UNLOAD` wenn die Remote-Quelle nicht verfügbar war.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.4.6.0 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit diesen Versionen in Abfragesprachen kompatibel ist:

- Die älteste unterstützte Version von Gremlin: `3.7.1`
- Die neueste unterstützte Version von Gremlin: `3.7.1`
- openCypher-Version: `Neptune-9.0.20190305-1.0`
- SPARQL-Version: `1.1`

Upgrade-Pfade auf Engine-Version 1.4.6.0

Sie können von der Engine-Version 1.2.0.0 oder höher auf diese Version aktualisieren.

Upgrade auf diesen Release

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe

der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.4.6.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.4.6.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Statt `--apply-immediately` können Sie `--no-apply-immediately` angeben. Um ein Hauptversions-Upgrade durchzuführen, ist der allow-major-version-upgrade Parameter erforderlich. Stellen Sie außerdem sicher, dass Sie die Engine-Version angeben, da Ihre Engine sonst möglicherweise auf eine andere Version aktualisiert wird.

Wenn Ihr Cluster eine benutzerdefinierte Cluster-Parametergruppe verwendet, müssen Sie diesen Parameter einschließen, um ihn zu anzugeben:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Ebenso sollte für Instances im Cluster, die eine benutzerdefinierte DB-Parametergruppe verwenden, dieser Parameter eingeschlossen werden, um ihn zu spezifizieren:

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein

Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
> 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.
> ```

Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWSPremium-Support zur Verfügung.

Amazon Neptune Engine, Version 1.4.5.1 (30.06.2025)

Ab dem 30.06.2025 wird die Engine-Version 1.4.5.1 allgemein eingesetzt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

In dieser Engine-Version wurden Fehler behoben

Allgemeine Korrekturen

- Es wurde ein Problem mit der serverlosen Skalierung behoben, das sich auf alle Abfragesprachen auswirkte.
- Es wurde ein Problem behoben, bei dem bei einigen OpenCypher Abfragen bei der Ausführung `collect(distinct())` auf einer Liste ein interner Fehler zurückgegeben wurde.
- Es wurde ein Problem behoben, bei dem eine gleichzeitige FTS-Abfrage dazu führte, dass andere gleichzeitige FTS-Abfragen beendet wurden, wenn Serverless verwendet OpenSearch wurde.

In dieser Version unterstützte Versionen in Abfragesprache

Stellen Sie vor dem Upgrade eines DB-Clusters auf Version 1.4.5.1 sicher, dass Ihr Projekt mit diesen Versionen in Abfragesprachen kompatibel ist:

- Die älteste unterstützte Version von Gremlin: `3.7.1`
- Die neueste unterstützte Version von Gremlin: `3.7.1`
- openCypher-Version: `Neptune-9.0.20190305-1.0`
- SPARQL-Version: `1.1`

Upgrade-Pfade auf Engine-Version 1.4.5.1

Sie können von der Engine-Version 1.2.0.0 oder höher auf diese Version aktualisieren.

Upgrade auf diesen Release

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.4.5.1 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.4.5.1 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Statt `--apply-immediately` können Sie `--no-apply-immediately` angeben. Um ein Hauptversions-Upgrade durchzuführen, ist der allow-major-version-upgrade Parameter erforderlich. Stellen Sie außerdem sicher, dass Sie die Engine-Version angeben, da Ihre Engine sonst möglicherweise auf eine andere Version aktualisiert wird.

Wenn Ihr Cluster eine benutzerdefinierte Cluster-Parametergruppe verwendet, müssen Sie diesen Parameter einschließen, um ihn zu anzugeben:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Ebenso sollte für Instances im Cluster, die eine benutzerdefinierte DB-Parametergruppe verwenden, dieser Parameter eingeschlossen werden, um ihn zu spezifizieren:

```
--db-instance-parameter-group-name (name of the custom instance parameter group)
```

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:

```
  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.
```

Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine, Version 1.4.5.0 (09.04.2025)

Ab dem 09.04.2025 wird die Engine-Version 1.4.5.0 allgemein eingesetzt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

Warning

Aufgrund von Problemen, die in bestimmten serverlosen Konfigurationen auftreten können, haben wir Upgrades auf 1.4.5.0 vorübergehend unterbrochen. Wir empfehlen ein Upgrade auf die Engine-Version 1.4.5.1. Upgrades auf 1.4.5.0 wurden vorübergehend deaktiviert.

Neue Funktionen in dieser Engine-Version

- Der DFE-Engine wurden neue Gremlin-Sprachschritte hinzugefügt.
  - Pfad und Durchlaufschritte: asDate (), DateAdd (), DateDiff (), fail (), Inject (), label (), path (), project (), repeat (), sack (), select (), unfold (), disjunct (), drop (), identity (), intersect (), length (), loops (), barrier (), order (), range (), reverse (), sample (), cap (), split (), filter (), flatMap (), map (), sideEffect (), union (), index ()
  - Aggregations- und Sammelschritte: aggregieren (global), combine (), count (), dedup (global), fold (), group (), groupCount ()

- Mathematische Schritte: max (), mean (), min (), sum ()
- Elementschritte: otherV (), elementMap (), element (), V (), out (), in (), both (), oute (), inE (), boTE (), boTE (), outV (), inV (), beide V (), otherV ()
- Eigenschaftsschritte: properties (), key (), valueMap (), value ()
- Schritte filtern: and (), coalesce (), coin (), is (), local (), none (), not () oder (), where ()
- Schritte zur Bearbeitung von Zeichenketten: concat (), lTrim (), rTrim (), substring (), toLower (), toUpper (), trim ()
- Prädikate:
  - Vergleiche: eq, neq, lt, lte, gt, gte
  - Enthält: innerhalb, ohne
  - TextP: endend mit, enthaltend,,, notStartingWith nichtenthaltend notEndingWith
  - P: und, oder, zwischen, draußen, drinnen

Einzelheiten zu allen verfügbaren Gremlin-Schritten in DFE finden Sie unter. Berichterstattung über Gremlin-Schritte in DFE

Verbesserungen in dieser Engine-Version

Allgemeine Verbesserungen

- Verbesserung der Wartezeit für langsame Abfragen bei Log-Sperren. Die Logs für langsame Abfragen enthalten jetzt Messwerte für Wartezeiten für gemeinsam genutzte und exklusive Sperren. Sie werden als Teil jeder Transaktion gespeichert, falls es zu verzögerter Lese-/Schreibzuweisung kommt. Diese Metriken werden im Abschnitt StorageCounters der Logs für langsame Abfragen angezeigt.
- Die Unterstützung für die folgenden Cipher Suites wurde eingestellt:
  - TLS_ECDHE_RSA_WITH_AES_128_CBC_ SHA256
  - TLS_ECDHE_RSA_WITH_AES_256_CBC_ SHA384
  - TLS_ECDHE_ECDSA_MIT_AES_128_CBC_ SHA256
  - TLS_ECDHE_ECDSA_WITH_AES_256_CBC_ SHA384

Verbesserungen bei Gremlin

- Der Gremlin-Sprache wurden viele neue Schritte hinzugefügt. Weitere Informationen finden Sie unter Berichterstattung über Gremlin-Schritte in DFE.

Verbesserungen bei OpenCypher

- Leistungsverbesserungen bei CREATE, MERGE und SET (Mutationen).
- Leistungsverbesserungen bei CALL Subquery.
- Support für HTTP-Trailing-Header-Unterstützung für mehrteilige OpenCypher-Antworten. Weitere Informationen finden Sie unter Optionale HTTP-Trailing-Header.
- OpenCypher wurde um die Temporalfunktion für Tag, Monat und Jahr erweitert. Weitere Informationen finden Sie unter Temporale Funktionen.

```
RETURN day(datetime('2021-06-03T01:48:14Z'))
{
  "results": [{
      "day(datetime('2021-06-03T01:48:14Z'))": 3
    }]
}
```

In dieser Engine-Version behobene Fehler

Allgemeine Korrekturen

- Das Problem, dass die Audit/SlowQueryLog Protokolldateien gelöscht wurden, wurde behoben.

Korrekturen für Gremlin

- Es wurde ein Problem behoben, bei dem Gremlin-Abfragen mit deaktivierter Ergebnis-Cache-Funktion ausgeführt wurden. Eine Abfrage, die mit iterate () endete, gab Ergebnisse zurück, anstatt eine leere Antwort zurückzugeben.
- Es wurde ein Problem mit dem Gremlin Result Cache behoben, das durch gleichzeitige Abfragen mit demselben Schlüssel verursacht wurde. Eine der gleichzeitig laufenden Abfragen gab fälschlicherweise Ergebnisse zurück, anstatt leere Ergebnisse zurückzugeben.

- Es wurde ein Problem mit Amazon S3 S3-Exportabfragen behoben, das dazu führte, dass eine Abfrage bei einem mehrteiligen Amazon S3-Upload aufgrund von Timeouts oder Stornierungen fehlschlug, indem die Bereinigungszeit verlängert wurde.
- Ein Berechtigungsproblem im Zusammenhang mit dem Gremlin Amazon S3 S3-Export wurde behoben.

Fehlerkorrekturen für SPARQL

- Es wurde ein Problem bei der Behandlung von SPARQL-Abfragen behoben, die mehrere Basisdaten deklarieren IRIs , wodurch nur die ursprüngliche Deklaration verwendet wurde.
- Es wurde ein Problem bei der Verarbeitung von `REPLACE` SPARQL-Funktionen behoben, bei denen ungültige Musterzeichenfolgen verwendet wurden, wodurch ein Fehler zurückgegeben wurde.
- Es wurde ein Problem bei der Verarbeitung von `REPLACE` SPARQL-Funktionen behoben, bei denen das Flag ohne Berücksichtigung der Groß-/Kleinschreibung (`"i"`) bei Unicode-Daten verwendet wurde.
- Es wurde ein Problem beim Parsen von SPARQL-Abfragen mit ungültigen Escape-Sequenzen \u und \U Codepoint-Escape-Sequenzen behoben, das dazu führen konnte, dass keine Antwort zurückgegeben wurde.
- Es wurde ein Problem in der `IRI` SPARQL-Funktion behoben, das relativ IRIs zum aktuellen Basis-IRI nicht immer korrekt aufgelöst wurde.
- Es wurde ein Problem behoben, das dazu führte`SPARQL INSERT DATA`, dass `DELETE DATA` Aktualisierungen, bei denen Namen mit Präfixen verwendet wurden, relativ zum aktuellen IRIs Basis-IRI nicht korrekt aufgelöst wurden.

In dieser Version unterstützte Versionen in Abfragesprache

Stellen Sie vor dem Upgrade eines DB-Clusters auf Version 1.4.5.0 sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprachen kompatibel ist:

- Die älteste unterstützte Version von Gremlin: `3.7.1`
- Die neueste unterstützte Version von Gremlin: `3.7.1`
- openCypher-Version: `Neptune-9.0.20190305-1.0`
- SPARQL-Version: `1.1`

Upgrade-Pfade auf Engine-Version 1.4.5.0

Sie können von der Engine-Version 1.2.0.0 oder höher auf diese Version aktualisieren.

Upgrade auf diesen Release

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.4.5.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.4.5.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Statt `--apply-immediately` können Sie `--no-apply-immediately` angeben. Um ein Hauptversions-Upgrade durchzuführen, ist der allow-major-version-upgrade Parameter erforderlich. Stellen Sie außerdem sicher, dass Sie die Engine-Version angeben, da Ihre Engine sonst möglicherweise auf eine andere Version aktualisiert wird.

Wenn Ihr Cluster eine benutzerdefinierte Cluster-Parametergruppe verwendet, müssen Sie diesen Parameter einschließen, um ihn zu anzugeben:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Ebenso sollte für Instances im Cluster, die eine benutzerdefinierte DB-Parametergruppe verwenden, dieser Parameter eingeschlossen werden, um ihn zu spezifizieren:

```
--db-instance-parameter-group-name (name of the custom instance parameter group)
```

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:

```
  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.
```

Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWSPremium-Support zur Verfügung.

Amazon Neptune Engine, Version 1.4.4.0 (24.02.2025)

Ab dem 24.02.2025 wird die Engine-Version 1.4.4.0 allgemein eingesetzt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

Warning

Der Abfrageplan-Cache wird für den Anwendungsfall der Ausführung parametrisierter Abfragen mit numerischen Parameterwerten vorübergehend nicht unterstützt, da ein Fehler bei der Behandlung doppelter Verwendungen eines numerischen Parameters in der Abfrage aufgetreten ist. Zum Beispiel:

```
MATCH (n:movie) WHERE n.runtime>=$minutes RETURN n
      UNION
      MATCH (n:show) WHERE n.duration>=$minutes RETURN n

      parameters={"minutes":130}
```

Bei Abfragen, die häufig Indexsuchen nach Anweisungen oder Wörterbuchindizes durchführen, konnte es zu einer Leistungseinbuße von 5% kommen. Beispielsweise hätte das keine Auswirkung darauf, die Anzahl aller Scheitelpunkte zu ermitteln oder die Anzahl `id` aller

Scheitelpunkte zu ermitteln. Beim Abrufen aller Eigenschaften aller Scheitelpunkte könnte es zu einer Regression von bis zu 5% kommen.

In dieser Engine-Version wurden Fehler behoben

Allgemeine Korrekturen

- Es wurde ein Problem behoben, bei dem die `WITH` Klausel, die ein Sternchen (`*`) und Aliasausdrücke enthielt, zu einer falschen Abfrageanalyse in OpenCypher führte.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.4.4.0 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit diesen Versionen in Abfragesprachen kompatibel ist:

- Die älteste unterstützte Version von Gremlin: `3.7.1`
- Die neueste unterstützte Version von Gremlin: `3.7.1`
- openCypher-Version: `Neptune-9.0.20190305-1.0`
- SPARQL-Version: `1.1`

Upgrade-Pfade auf Engine-Version 1.4.4.0

Sie können von der Engine-Version 1.2.0.0 oder höher auf diese Version aktualisieren.

Upgrade auf diesen Release

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.4.4.0 \
```

```
    --allow-major-version-upgrade \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.4.4.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Statt `--apply-immediately` können Sie `--no-apply-immediately` angeben. Um ein Hauptversions-Upgrade durchzuführen, ist der allow-major-version-upgrade Parameter erforderlich. Stellen Sie außerdem sicher, dass Sie die Engine-Version angeben, da Ihre Engine sonst möglicherweise auf eine andere Version aktualisiert wird.

Wenn Ihr Cluster eine benutzerdefinierte Cluster-Parametergruppe verwendet, müssen Sie diesen Parameter einschließen, um ihn zu anzugeben:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Ebenso sollte für Instances im Cluster, die eine benutzerdefinierte DB-Parametergruppe verwenden, dieser Parameter eingeschlossen werden, um ihn zu spezifizieren:

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue

Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
> 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.
> ```
>
> Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine, Version 1.4.3.0 (21.01.2025)

Ab dem 21.01.2025 wird die Engine-Version 1.4.3.0 allgemein eingesetzt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

Warning

Der Abfrageplan-Cache wird für den Anwendungsfall der Ausführung parametrisierter Abfragen mit numerischen Parameterwerten vorübergehend nicht unterstützt, da ein Fehler bei der Behandlung doppelter Verwendungen eines numerischen Parameters in der Abfrage aufgetreten ist. Zum Beispiel:

```
MATCH (n:movie) WHERE n.runtime>=$minutes RETURN n
      UNION
      MATCH (n:show) WHERE n.duration>=$minutes RETURN n

      parameters={"minutes":130}
```

Bei Abfragen, die häufig Indexsuchen nach Anweisungen oder Wörterbuchindizes durchführen, konnte es zu einer Leistungseinbuße von 5% kommen. Beispielsweise hätte das keine Auswirkung darauf, die Anzahl aller Scheitelpunkte zu ermitteln oder die Anzahl `id` aller Scheitelpunkte zu ermitteln. Beim Abrufen aller Eigenschaften aller Scheitelpunkte könnte es zu einer Regression von bis zu 5% kommen.

Neue Funktionen in dieser Engine-Version

- Exportieren von Gremlin-Abfrageergebnissen nach Amazon S3. Exportieren von Gremlin-Abfrageergebnissen direkt nach Amazon S3. Mit dieser Funktion können Sie große Abfrageergebnisse effizient verarbeiten, indem Sie sie in einen Amazon S3 S3-Bucket exportieren, anstatt sie als Abfrageantwort zurückzugeben.

```
g.V().
    hasLabel('Comment').
    valueMap().
    call('neptune.query.exportToS3', [
    'destination': 's3://your-bucket/path/result.json',
    'format': 'GraphSONv3',
    'keyArn': 'optional-kms-key-arn'
```

```
])
```

- R7i-Instanzen. Die R7i-Instance-Familie, bis zu 48xlarge, ist jetzt in den folgenden Regionen verfügbar:
  - ap-northeast-1 - Asien-Pazifik (Tokio)
  - ap-northeast-2 - Asien-Pazifik (Seoul)
  - ap-south-1 - Asien-Pazifik (Mumbai)
  - ap-southeast-1 - Asien-Pazifik (Singapur)
  - ap-southeast-2 - Asien-Pazifik (Sydney)
  - ap-southeast-3 - Asien-Pazifik (Jakarta)
  - ca-central-1 - Kanada (Zentral)
  - eu-central-1 - Europa (Frankfurt)
  - eu-north-1 - Europa (Stockholm)
  - eu-south-2 - Europa (Spanien)
  - eu-west-1 - Europa (Irland)
  - eu-west-2 - Europa (London)
  - eu-west-3 - Europa (Paris)
  - us-east-1 - USA Ost (Nord-Virginia)
  - us-east-2 - USA Ost (Ohio)
  - us-west-1 - USA West (Nordkalifornien)
  - us-west-2 - USA West (Oregon)

Verbesserungen in dieser Engine-Version

Allgemeine Verbesserungen

- Unterstützung im Labormodus für die Wörterbuch-Garbage-Collection (GC).

  Wenn diese Option aktiviert ist, werden die ungenutzten Glossareinträge durch einen Hintergrundjob bereinigt. Es reduziert nicht`VolumeBytesUsed`, sondern gibt Platz im Index für neue Einfügungen frei. Die Wachstumsrate von `VolumeBytesUsed` ist wahrscheinlich geringer, wenn die Wörterbuch-GC aktiviert ist, als wenn sie nicht aktiviert ist. Dies funktioniert für Eigenschaftsdiagrammdaten (die über Gremlin oder OpenCypher eingefügt wurden), wenn der

`neptune_streams` Parameter nicht aktiviert ist. Weitere Informationen finden Sie unter Müllabfuhr im Neptun-Wörterbuch.

In dieser Engine-Version wurden Fehler behoben

Allgemeine Korrekturen

- Es wurden zwei Probleme mit Speicherlecks behoben, die sich FreeableMemory bei der Verwendung der DFE-Engine auswirkten.

OpenCypher-Korrekturen

- Das Problem mit MERGE ON MATCH/ON CREATE für doppelte Zeilen wurde behoben.

```
UNWIND [1, 1] AS id
MERGE (n:Person {id: id})
  ON CREATE SET n.p = 5
  ON MATCH SET n.p = 6
```

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.4.3.0 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprachen kompatibel ist:

- Die älteste unterstützte Version von Gremlin: `3.7.1`
- Die neueste unterstützte Version von Gremlin: `3.7.1`
- openCypher-Version: `Neptune-9.0.20190305-1.0`
- SPARQL-Version: `1.1`

Upgrade-Pfade auf Engine-Version 1.4.3.0

Sie können von der Engine-Version 1.2.0.0 oder höher auf diese Version aktualisieren.

Upgrade auf diesen Release

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe

der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.4.3.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.4.3.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Statt `--apply-immediately` können Sie `--no-apply-immediately` angeben. Um ein Hauptversions-Upgrade durchzuführen, ist der allow-major-version-upgrade Parameter erforderlich. Stellen Sie außerdem sicher, dass Sie die Engine-Version angeben, da Ihre Engine sonst möglicherweise auf eine andere Version aktualisiert wird.

Wenn Ihr Cluster eine benutzerdefinierte Cluster-Parametergruppe verwendet, müssen Sie diesen Parameter einschließen, um ihn zu anzugeben:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Ebenso sollte für Instances im Cluster, die eine benutzerdefinierte DB-Parametergruppe verwenden, dieser Parameter eingeschlossen werden, um ihn zu spezifizieren:

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein

Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
> 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.
> ```

Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine, Version 1.4.2.0 (19.12.2024)

Ab 2024-12-19 wird die Engine-Version 1.4.2.0 allgemein eingesetzt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

Warning

Der Abfrageplan-Cache wird für den Anwendungsfall der Ausführung parametrisierter Abfragen mit numerischen Parameterwerten vorübergehend nicht unterstützt, da ein Fehler bei der Behandlung doppelter Verwendungen eines numerischen Parameters in der Abfrage aufgetreten ist. Zum Beispiel:

```
MATCH (n:movie) WHERE n.runtime>=$minutes RETURN n
      UNION
      MATCH (n:show) WHERE n.duration>=$minutes RETURN n

      parameters={"minutes":130}
```

Bei Abfragen, die häufig Indexsuchen nach Anweisungen oder Wörterbuchindizes durchführen, konnte es zu einem Leistungsrückgang von 5% kommen. Beispielsweise hätte das keine Auswirkung darauf, die Anzahl aller Scheitelpunkte zu ermitteln oder die Anzahl `id` aller Scheitelpunkte zu ermitteln. Beim Abrufen aller Eigenschaften aller Scheitelpunkte könnte es zu einer Regression von bis zu 5% kommen.

In dieser Engine-Version wurden Fehler behoben

Allgemeine Korrekturen

- Der Ausführungsplan und die Leistung für Abfragen mit mehreren Eigenschaftszugriffen über dieselbe Variable in OPTIONAL MATCH und Listenverständnis wurden korrigiert. Zum Beispiel:

```
MATCH (n)
  WHERE n.name = 'A'
OPTIONAL MATCH (n)-[:knows]->(m)
  WHERE m.name = 'C' or m.city = 'B'
RETURN m
```

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.4.2.0 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit diesen Versionen in Abfragesprachen kompatibel ist:

- Die älteste unterstützte Version von Gremlin: `3.7.1`
- Die neueste unterstützte Version von Gremlin: `3.7.1`
- openCypher-Version: `Neptune-9.0.20190305-1.0`
- SPARQL-Version: `1.1`

Upgrade-Pfade auf Engine-Version 1.4.2.0

Sie können von der Engine-Version 1.2.0.0 oder höher auf diese Version aktualisieren.

Upgrade auf diesen Release

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
```

```
    --engine-version 1.4.2.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.4.2.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Statt `--apply-immediately` können Sie `--no-apply-immediately` angeben. Um ein Hauptversions-Upgrade durchzuführen, ist der allow-major-version-upgrade Parameter erforderlich. Stellen Sie außerdem sicher, dass Sie die Engine-Version angeben, da Ihre Engine sonst möglicherweise auf eine andere Version aktualisiert wird.

Wenn Ihr Cluster eine benutzerdefinierte Cluster-Parametergruppe verwendet, müssen Sie diesen Parameter einschließen, um ihn zu anzugeben:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Ebenso sollte für Instances im Cluster, die eine benutzerdefinierte DB-Parametergruppe verwenden, dieser Parameter eingeschlossen werden, um ihn zu spezifizieren:

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue

Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
> 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.
> ```
>
> Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine, Version 1.4.1.0 (21.11.2024)

Ab 2024-11-21 wird die Engine-Version 1.4.1.0 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

Warning

Der Abfrageplan-Cache wird für den Anwendungsfall der Ausführung parametrisierter Abfragen mit numerischen Parameterwerten vorübergehend nicht unterstützt, da ein Fehler bei der Behandlung doppelter Verwendungen eines numerischen Parameters in der Abfrage aufgetreten ist. Beispiel:

```
MATCH (n:movie) WHERE n.runtime>=$minutes RETURN n
      UNION
      MATCH (n:show) WHERE n.duration>=$minutes RETURN n

      parameters={"minutes":130}
```

Bei Abfragen, die häufig Indexsuchen nach Anweisungen oder Wörterbuchindizes durchführen, konnte es zu einer Leistungseinbuße von 5% kommen. Beispielsweise hätte das keine Auswirkung darauf, die Anzahl aller Scheitelpunkte zu ermitteln oder die Anzahl `id` aller Scheitelpunkte zu ermitteln. Beim Abrufen aller Eigenschaften aller Scheitelpunkte könnte es zu einer Regression von bis zu 5% kommen.

Neue Funktionen in dieser Engine-Version

- Es wurde Unterstützung für `CALL` Unterabfragen mit einer schreibgeschützten Unterabfrage hinzugefügt, sodass Operationen innerhalb eines definierten Bereichs ausgeführt werden können. Eine `CALL` Unterabfrage wird einmal für jede eingehende Zeile ausgeführt, und die in einer Unterabfrage zurückgegebenen Variablen sind für den äußeren Bereich der umschließenden Abfrage verfügbar. Variablen aus dem äußeren Gültigkeitsbereich können mithilfe einer Importklausel in eine `CALL` Unterabfrage importiert werden. `WITH` Weitere Informationen finden Sie unter Unterstützung von CALL-Unterabfragen in 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
```

- OpenCypher-Funktionen hinzugefügt. Wir führen acht neue Funktionen ein, die bei Zeichenketten, Sammeloperationen und der Sortierung von Sammlungen helfen. Dazu gehören: `textIndexOf`,`collToSet`,`collSubtract`,`collIntersection`, `collSort`,`collSortMaps`,`collSortMulti`, und`collSortNodes`. Beschreibung, Eingabeparameter, Ausgabe und Beispiele finden Sie unter Neptune OpenCypher-Funktionen.

Verbesserungen in dieser Engine-Version

Verbesserungen bei Gremlin

- Neuer Parameter für den Labormodus`AccurateQRCMemoryEstimation`. Wenn der Gremlin-Abfrageergebnis-Cache aktiviert ist, können Abfrageergebnisse in der Datenbank zwischengespeichert werden. Standardmäßig wird eine ungefähre Schätzung verwendet, um die Größe des zwischengespeicherten Ergebnisses zu bestimmen. Wenn dieser Parameter im Labormodus `AccurateQRCMemoryEstimation` aktiviert ist, verwendet die Größenschätzung für zwischengespeicherte Ergebnisse eine genaue Größenschätzung anstelle einer ungefähren Größenschätzung.
- Es wurde ein Problem mit der Filteroptimierung „nicht“ in Gremlin-Abfragen behoben, die auf der Standard-Ausführungs-Engine ausgeführt wurden. Dieses Problem betraf Abfragen, bei denen Kanten mit dem Schritt not () in Kombination mit einem der Schritte outV () /inV () /otherV () gefiltert wurden. Zu den Beispielabfragen gehören:
  - `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()`

Verbesserungen bei OpenCypher

- Verbesserte Leistung bei Abfragen, die große statische Listen oder Maps verwenden. Bei bestimmten Abfragen mit UNWIND über eine große Liste verschachtelter Maps, mit denen ein Knoten mit Eigenschaften eingefügt oder verschoben wird, wurde die Leistung erheblich verbessert.

- Führt einen neuen OpenCypher-Abfragehinweis ein, der die Engine anweist, konsistente Datentypen für die in der Abfrage verwendeten Werte anzunehmen. Einzelheiten AssumeConsistentDataTypeszum neuen OpenCypher-Abfragehinweis finden Sie unter.
- Führt eine Reihe neuer OpenCypher-Funktionen für den Umgang mit Text- und Sammlungswerten ein.

In dieser Engine-Version wurden Fehler behoben

Korrekturen für Gremlin

- Es wurde ein Problem im TinkerPop OSS-Codepfad behoben, bei dem eine Bytecode-Darstellung einer Traversal-Abfrage erstellt wurde, wenn einer der `withStrategies()/withoutStrategies()/with()` Schritte für ein `GraphTraversalSource` „g“ -Objekt verwendet wurde. Das Problem fügte fälschlicherweise neue Anweisungen an Bytecode an, anstatt bestehende Anweisungen für dieselbe Strategie zu ersetzen, und führte dazu, dass bei der Invalidierung des Ergebnis-Caches beim Löschen der gespeicherten Ergebnisse die Cache-Schlüssel nicht übereinstimmten.

OpenCypher-Korrekturen

- Das Verhalten von In-Klauseln wurde korrigiert. `` `~id`match `` CREATE/MERGE/MATCH Bei Verwendung eines ungültigen `` `~id` `` Werts wie Null oder Typen, die keine Zeichenfolge sind, wird jetzt für CREATE/MERGE Klauseln eine korrekte Ausnahme ausgelöst und für eine Klausel wird kein Ergebnis zurückgegeben. `MATCH`
- <string>IFE wurde behoben, wenn der Benutzer einen Wert eines nicht unterstützten Typs mit Aggregationsfunktionen verwendet (d. h. sum ()).
- Es wurde ein Problem behoben, bei dem einige Mutationsanfragen mit niedriger Latenz aus einer großen Anzahl von Abfragen mit einem Fehler fehlschlugen. OutOfMemory

Fehlerkorrekturen für SPARQL

- Es wurde ein Problem mit dem Auditprotokoll bei der Verarbeitung von SPARQL-Abfragen behoben, die das `'%'` Zeichen enthalten.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.4.1.0 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit diesen Versionen in Abfragesprachen kompatibel ist:

- Die älteste unterstützte Version von Gremlin: `3.7.1`
- Die neueste unterstützte Version von Gremlin: `3.7.1`
- openCypher-Version: `Neptune-9.0.20190305-1.0`
- SPARQL-Version: `1.1`

Upgrade-Pfade auf Engine-Version 1.4.1.0

Sie können von der Engine-Version 1.2.0.0 oder höher auf diese Version aktualisieren.

Upgrade auf diesen Release

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.4.1.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.4.1.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Statt `--apply-immediately` können Sie `--no-apply-immediately` angeben. Um ein Upgrade einer Hauptversion durchzuführen, ist der allow-major-version-upgrade Parameter

erforderlich. Stellen Sie außerdem sicher, dass Sie die Engine-Version angeben, da Ihre Engine sonst möglicherweise auf eine andere Version aktualisiert wird.

Wenn Ihr Cluster eine benutzerdefinierte Cluster-Parametergruppe verwendet, müssen Sie diesen Parameter einschließen, um ihn zu anzugeben:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Ebenso sollte für Instances im Cluster, die eine benutzerdefinierte DB-Parametergruppe verwenden, dieser Parameter eingeschlossen werden, um ihn zu spezifizieren:

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
>   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.
> ```
>
> Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWSPremium-Support zur Verfügung.

Amazon Neptune Engine, Version 1.4.0.0 (06.11.2024)

Ab 2024-11-06 wird die Engine-Version 1.4.0.0 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

> **Note**
>
> Mit der Engine-Version 1.3.0.0 wurde ein neues Format für benutzerdefinierte Parametergruppen und benutzerdefinierte Cluster-Parametergruppen eingeführt. Wenn Sie also von einer Engine-Version vor 1.3.0.0 auf Engine-Version 1.3.0.0 oder höher aktualisieren, müssen Sie alle vorhandenen benutzerdefinierten Parametergruppen und benutzerdefinierten Cluster-Parametergruppen mithilfe der Parametergruppenfamilie neu

erstellen. `neptune1.3` In früheren Versionen wurde die Parametergruppenfamilie `neptune1` oder `neptune1.2` verwendet. Diese Parametergruppen funktionieren nicht mit Version 1.3.0.0 und höher. Ebenso sollten Sie Cluster-Parametergruppen der Version 1.4.0.0 für Engine-Versionen 1.4.0.0 und höher verwenden. Weitere Informationen finden Sie unter Amazon-Neptune-Parametergruppen.

Warning

Der Abfrageplan-Cache wird für den Anwendungsfall der Ausführung parametrisierter Abfragen mit numerischen Parameterwerten vorübergehend nicht unterstützt, da ein Fehler bei der Behandlung doppelter Verwendungen eines numerischen Parameters in der Abfrage aufgetreten ist. Beispiel:

```
MATCH (n:movie) WHERE n.runtime>=$minutes RETURN n
      UNION
      MATCH (n:show) WHERE n.duration>=$minutes RETURN n

      parameters={"minutes":130}
```

Bei Abfragen, die häufig Indexsuchen nach Anweisungen oder Wörterbuchindizes durchführen, konnte es zu einer Leistungseinbuße von 5% kommen. Beispielsweise hätte das keine Auswirkung darauf, die Anzahl aller Scheitelpunkte zu ermitteln oder die Anzahl `id` aller Scheitelpunkte zu ermitteln. Beim Abrufen aller Eigenschaften aller Scheitelpunkte könnte es zu einer Regression von bis zu 5% kommen.

Neue Funktionen in dieser Engine-Version

- Wenn einem Eigenschaftsdiagramm eine Kante ohne explizite ID hinzugefügt wird, weist der Server standardmäßig eine UUID-basierte Edge-ID zu, die im Wörterbuch gespeichert wird. Wenn Sie nun einen neuen Clusterparameter festlegen`neptune_enable_server_generated_edge_id = 1`, IDs verwendet der Server eine intern verwaltete 8-Byte-Ganzzahl, ohne dass ein Wörterbuch-Overhead entsteht. Dies führt zu Speichereinsparungen und einer verbesserten Abfrageleistung, ohne dass Änderungen an den Abfragen vorgenommen werden müssen. Diese Funktion wird derzeit nur für Einfügungen über die Gremlin-Abfragesprache unterstützt.

- Unterstützung für die Ausführung von Gremlin Limit () -Schritten in verschachtelten Traversals für die DFE-Engine hinzugefügt.

```
g.V().project("foo").by(out().order().by(T.id).limit(1))
```

Verbesserungen in dieser Engine-Version

Allgemeine Verbesserungen

- Neptune fordert automatisch den Undo-Speicher für große Transaktionen zurück, sobald die Transaktion abgeschlossen ist und keine Protokolle mehr für die Wiederherstellung benötigt werden.
- Support für Global Database Survivable Replica. Diese Funktion ermöglicht es dem sekundären Cluster, während eines Neustarts der Writer-Instanz auf dem primären Cluster weiterhin Leseanforderungen zu bearbeiten. Bisher wurden beim Neustart einer Writer-Instanz auch alle Reader-Instanzen in einem sekundären Cluster neu gestartet. Mit dieser Version bearbeiten sekundäre Cluster-Reader-Instances während des Neustarts einer Writer-Instanz weiterhin Leseanfragen, wodurch die Leseverfügbarkeit im Cluster verbessert wird.
- Audit-Logs werden jetzt synchron geschrieben, sodass garantiert ist, dass jede Abfrage protokolliert wird. Dies kann die Leistung bei besonders großen Abfragen (>100 KB) oder Workloads mit hohem Durchsatz (>1000 qps) beeinträchtigen.

Verbesserungen bei Gremlin

- Das Timeout pro Abfrage wird standardmäßig so festgelegt, dass es kleiner ist als das Timeout auf Clusterebene. In einer früheren Version wurde diese Prüfung eingeführt, musste aber explizit über den Labormodus-Parameter " aktiviert werden. StrictTimeoutValidation In dieser Version ist 'StrictTimeoutValidation' standardmäßig aktiviert und muss explizit deaktiviert werden, um das alte Verhalten beizubehalten.

Verbesserungen bei OpenCypher

- In einer früheren Version haben wir erweiterte Unterstützung für das Datetime-Format eingeführt, die über einen Labormodus-Parameter `DatetimeMillisecond` aktiviert wurde. Diese erweiterte Unterstützung für Datetime-Formate ist jetzt standardmäßig aktiviert.

Verbesserungen bei SPARQL

- Neue explizite IAM-Aktionen für Abfrageberechtigungen.

```
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.
```

In dieser Engine-Version behobene Fehler

Allgemeine Korrekturen

- Es wurde ein Problem mit serverlosen Instanzen behoben, das bei der Skalierung zu einem Neustart der Datenbank führen konnte.
- Es wurde ein Problem im Zusammenhang mit der Verwaltung von Audit-Protokolldateien behoben, das dazu führen konnte, dass auf Protokolldateien für den Download oder die Rotation nicht zugegriffen werden konnte und das in einigen Fällen die CPU-Auslastung erhöhte.
- Es wurde ein Abfrageproblem im Zusammenhang mit der Optimierung behoben, das die Generierung der Kartenausgabe in der DFE-Engine verzögerte.
- Es wurde ein Problem behoben, das dazu führte, dass die Zeitstempel zwischen den Audit-Logs und den Logs für langsame Abfragen nicht übereinstimmten.

Korrekturen für Gremlin

- Es wurde ein Problem in der WebSocket Gremlin-Verbindungsverwaltung behoben, bei dem Abfragen, deren Dauer das Leerlaufzeitlimit der Verbindung überschritt, vorzeitig beendet wurden. Dies betraf insbesondere Python-Gremlin-Clients, die den AIOHTTP-Transport verwenden.

OpenCypher-Korrekturen

- Es wurde ein Problem im Sammelschritt behoben, das zu einer internen Fehlerausnahme führte, wenn während des Abfragekonstrukts collect (distinct (n)) Nullwerte vorhanden waren.
- Es wurde ein Problem behoben, bei dem ein in Abfragen `NullPointerException` auftreten konnte, wenn der Abfrageplan-Cache aktiviert war.
- Es wurde ein Problem behoben, bei dem mehr Daten als nötig ausgewertet wurden, wenn eine Abfrage eine LIMIT-Klausel enthielt.
- Es wurde ein Problem behoben, bei dem die Verwendung von Bereichsoperationen (**<**, **<=**, **>**, **> =**) in einer parametrisierten Abfrage mit einem Abfrageplan-Cache zu doppelten Ergebnissen führte.
- Es wurde ein Problem behoben, bei dem Ergebnisspalten transponiert wurden, wenn die Operationen UNION und UNION ALL mithilfe von Schraubenverbindungen ausgeführt wurden.

In dieser Version unterstützte Versionen in Abfragesprache

Stellen Sie vor dem Upgrade eines DB-Clusters auf Version 1.4.0.0 sicher, dass Ihr Projekt mit diesen Versionen in Abfragesprachen kompatibel ist:

- Die älteste unterstützte Version von Gremlin: `3.7.1`
- Die neueste unterstützte Version von Gremlin: `3.7.1`
- openCypher-Version: `Neptune-9.0.20190305-1.0`
- SPARQL-Version: `1.1`

Upgrade-Pfade auf Engine-Version 1.4.0.0

Sie können von der Engine-Version 1.2.0.0 oder höher auf diese Version aktualisieren.

Upgrade auf diesen Release

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.4.0.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.4.0.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Statt `--apply-immediately` können Sie `--no-apply-immediately` angeben. Um ein Hauptversions-Upgrade durchzuführen, ist der allow-major-version-upgrade Parameter erforderlich. Stellen Sie außerdem sicher, dass Sie die Engine-Version angeben, da Ihre Engine sonst möglicherweise auf eine andere Version aktualisiert wird.

Wenn Ihr Cluster eine benutzerdefinierte Cluster-Parametergruppe verwendet, müssen Sie diesen Parameter einschließen, um ihn zu anzugeben:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Ebenso sollte für Instances im Cluster, die eine benutzerdefinierte DB-Parametergruppe verwenden, dieser Parameter eingeschlossen werden, um ihn zu spezifizieren:

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
> 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.
> ```
>
> Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWSPremium-Support zur Verfügung.

Amazon Neptune Engine version 1.3.4.0 (2024-10-01)

Ab 2024-10-01 wird die Engine-Version 1.3.4.0 allgemein eingesetzt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

> **Note**
>
> Mit der Engine-Version 1.3.0.0 wurde ein neues Format für benutzerdefinierte Parametergruppen und benutzerdefinierte Cluster-Parametergruppen eingeführt. Wenn Sie also von einer Engine-Version vor 1.3.0.0 auf Engine-Version 1.3.0.0 oder höher aktualisieren, müssen Sie alle vorhandenen benutzerdefinierten Parametergruppen und benutzerdefinierten Cluster-Parametergruppen mithilfe der Parametergruppenfamilie neu erstellen. `neptune1.3` In früheren Versionen wurde die Parametergruppenfamilie `neptune1` oder `neptune1.2` verwendet. Diese Parametergruppen funktionieren nicht mit Version 1.3.0.0 und höher. Ebenso sollten Sie Cluster-Parametergruppen der Version 1.4.0.0 für Engine-Versionen 1.4.0.0 und höher verwenden. Weitere Informationen finden Sie unter Amazon-Neptune-Parametergruppen.

> **Warning**
>
> Bei SPARQL 1.1-Aktualisierungsvorgängen wurde ein Problem festgestellt, das unter bestimmten Bedingungen auftreten kann, wenn Aktualisierungsoperatoren mit aktionsbasierten Autorisierungsrichtlinien verwendet werden. Wenn Sie die SPARQL 1.1-Aktualisierungsoperationen mit aktionsbasierten Autorisierungsrichtlinien verwenden, empfehlen wir, auf die neueste Neptune-Minor-Engine-Version (mindestens 1.3.4.0) zu aktualisieren, die eine Lösung für dieses Problem enthält.
> Der Abfrageplan-Cache wurde vorübergehend für parametrisierte Abfragen mit numerischen Parameterwerten deaktiviert, da ein Problem bei der Behandlung doppelter Verwendungen eines numerischen Parameters wie in der folgenden Abfrage aufgetreten ist:
>
> ```
> MATCH (n:movie) WHERE n.runtime>=$minutes RETURN n
> UNION
> ```

```
MATCH (n:show) WHERE n.duration>=$minutes RETURN n

parameters={"minutes":130}
```

Verbesserungen in dieser Engine-Version

- Unterstützung für die Ausführung von Gremlin Limit () -Schritten in verschachtelten Traversals für die DFE-Engine hinzugefügt.
- Es wurden CloudWatch Metriken zum Gremlin-Ergebnis-Cache hinzugefügt, wie unten aufgeführt, die bei der Diagnose und Optimierung der Latenz im Ergebnis-Cache nützlich sein können. Einzelheiten finden Sie unter Neptune Metrics.

```
NumResultCacheHit
NumResultCacheMiss
ResultCacheSizeInBytes
ResultCacheItemCount
ResultCacheOldestItemTimestamp
ResultCacheNewestItemTimestamp
```

In dieser Engine-Version wurden Fehler behoben

Allgemeine Verbesserungen

- Es wurde ein Problem behoben, bei dem die Engine in seltenen Fällen abstürzte, anstatt einen Abfragefehler zurückzugeben.

Korrekturen für Gremlin

- Wir haben die Bearbeitung von Anfragen und die Fehlerberichterstattung verbessert, wenn ein Client oder ein Proxy eine Websocket-Upgrade-Anfrage über eine established/used HTTP-Verbindung sendet (zuvor wurden 400 Antworten mit dem Fehler „Kein Gremlin-Skript bereitgestellt, Code MissingParameterException" zurückgegeben).
- Die Verarbeitung von `mergeV` Schritten mit Aktualisierungen von Eigenschaftswerten mit einfacher Kardinalität wurde optimiert. Zum Beispiel wird die folgende Abfrage jetzt nativ in Neptune unterstützt.

```
g.mergeV([(T.id): 1234]). option(onMatch, ['age': single(20), 'name':
 single('alice'), 'city': set('miami')])
```

- Es wurde ein Problem bei der Auswertung von Gremlin DFE-Abfragen behoben, das dazu führte, dass Abfragen mit einem fehlschlugen. `InternalFailureException` Dieser Fehler trat bei bestimmten Mustern auf, `select` wie im folgenden Beispiel gezeigt:

```
g.V("1").as("a").as("b").select("a","b").dedup()
```

OpenCypher-Korrekturen

- Es wurde ein Problem behoben, bei dem beim Ausführen `collect(distinct())` mit vorhandenen Nullwerten ein Fehler zurückgegeben wurde.
- Es wurde ein Problem behoben, bei dem das Ausführen einer parametrisierten Abfrage mit Bereichsfilter (</<=/>/>= gegen den Parameterwert) zu Ergebnissen führte. duplicate/missing
- Es wurde ein Fehler behoben, bei dem die DFE-Engine mehr Ausgabe als bei Limit-Abfragen angefordert erzeugte, was zu Fehlern aufgrund unzureichenden Speichers führen konnte.

Fehlerkorrekturen für SPARQL

- Es wurde ein Problem behoben, bei dem beim Ausführen einer föderierten SPARQL-Aktualisierungsabfrage auf Clustern mit aktivierter IAM-Authentifizierung ein Fehler zurückgegeben wurde.
- Aktionsbasierte Berechtigungen für SPARQL 1.1-Aktualisierungsvorgänge wurden behoben.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.3.4.0 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit diesen Versionen in Abfragesprachen kompatibel ist:

- Die älteste unterstützte Version von Gremlin: `3.7.1`
- Die neueste unterstützte Version von Gremlin: `3.7.1`
- openCypher-Version: `Neptune-9.0.20190305-1.0`
- SPARQL-Version: `1.1`

Upgrade-Pfade auf Engine-Version 1.3.4.0

Sie können von der Engine-Version 1.2.0.0 oder höher auf diese Version aktualisieren.

Upgrade auf diesen Release

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.3.4.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.3.4.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Statt `--apply-immediately` können Sie `--no-apply-immediately` angeben. Um ein Hauptversions-Upgrade durchzuführen, ist der allow-major-version-upgrade Parameter erforderlich. Stellen Sie außerdem sicher, dass Sie die Engine-Version angeben, da Ihre Engine sonst möglicherweise auf eine andere Version aktualisiert wird.

Wenn Ihr Cluster eine benutzerdefinierte Cluster-Parametergruppe verwendet, müssen Sie diesen Parameter einschließen, um ihn zu anzugeben:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Ebenso sollte für Instances im Cluster, die eine benutzerdefinierte DB-Parametergruppe verwenden, dieser Parameter eingeschlossen werden, um ihn zu spezifizieren:

```
--db-instance-parameter-group-name (name of the custom instance parameter group)
```

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:

```
  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.
```

Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine, Version 1.3.3.0 (05.08.2024)

Ab 2024-08-05 wird die Engine-Version 1.3.3.0 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

Note

Mit der Engine-Version 1.3.0.0 wurde ein neues Format für benutzerdefinierte Parametergruppen und benutzerdefinierte Cluster-Parametergruppen eingeführt. Wenn Sie also von einer Engine-Version vor 1.3.0.0 auf Engine-Version 1.3.0.0 oder höher aktualisieren, müssen Sie alle vorhandenen benutzerdefinierten Parametergruppen und benutzerdefinierten Cluster-Parametergruppen mithilfe der Parametergruppenfamilie neu erstellen. `neptune1.3` In früheren Versionen wurde die Parametergruppenfamilie `neptune1` oder `neptune1.2` verwendet. Diese Parametergruppen funktionieren nicht mit Version 1.3.0.0 und höher. Ebenso sollten Sie Cluster-Parametergruppen der Version 1.4.0.0 für Engine-Versionen 1.4.0.0 und höher verwenden. Weitere Informationen finden Sie unter Amazon-Neptune-Parametergruppen.

> **Warning**
>
> Mit der Engine-Version 1.3.3.0 wurden einige potenzielle Probleme eingeführt, die Sie beachten sollten. Weitere Informationen finden Sie im Behebung von Problemen in Version 1.3.3.0 folgenden Abschnitt.

In dieser Engine-Version behobene Fehler

Allgemeine Verbesserungen

- Es wurde ein Problem behoben, bei dem die Engine instabil wurde, wenn sich eine hohe Anzahl von Prädikaten im Prädikat-Cache befand.

OpenCypher-Korrekturen

- Es wurde ein Problem behoben, bei dem die Abfrageausführung hängen bleiben kann, nachdem eine interne Ausnahme ausgelöst wurde.
- Es wurde ein Problem behoben, bei dem eine Abfrage mit einer internen Ausnahme fehlschlagen kann, wenn der Abfrageplan-Cache verwendet wird.

Fehlerkorrekturen für SPARQL

- Es wurde ein Problem mit dem SPARQL 1.1 Graph Store HTTP Protocol (GSP) behoben, das unter bestimmten Bedingungen auftreten kann, wenn GSP mit aktionsbasierten Autorisierungsrichtlinien verwendet wird.

Behebung von Problemen in Version 1.3.3.0

- Abfragen, die numerische Filterwerte verwenden, können falsche Ergebnisse zurückgeben, wenn der Abfrageplan-Cache verwendet wird. Um das Problem zu vermeiden, verwenden Sie den Abfragehinweis, `QUERY:PLANCACHE "disabled"` um den Abfrageplan-Cache zu überspringen. Verwenden Sie beispielsweise:

```
USING QUERY:PLANCACHE "disabled"
MATCH (n:person)
WHERE n.yearOfBirth > $year
```

```
RETURN n

parameters={"year":1950}
```

- Abfragen, die denselben Parameternamen mehrfach verwenden, können mit dem Fehler fehlschlagen`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`. Überspringen Sie in solchen Fällen entweder den Abfrageplan-Cache wie oben oder duplizieren Sie die Parameter wie in diesem Beispiel:

  ```
  MATCH (n:movie) WHERE n.runtime>=$minutes RETURN n
  UNION
  MATCH (n:show) WHERE n.duration>=$minutes RETURN n

  parameters={"minutes":130}
  ```

  Verwenden Sie den Hinweis `QUERY:PLANCACHE "disabled"` oder ändern Sie die Parameter:

  ```
  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}
  ```

- Abfragen, die mit dem Bolt-Protokoll ausgeführt werden, können zu falschen Ergebnissen führen, wenn es sich bei der Abfrage um eine UNION- oder UNION ALL-Abfrage handelt. Um das Problem zu vermeiden, sollten Sie erwägen, die jeweilige Abfrage mit dem HTTP-Endpunkt auszuführen. Alternativ können Sie jeden Teil der Union separat ausführen, wenn Sie das Bolt-Protokoll verwenden.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.3.3.0 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprache kompatibel ist:

- Die älteste unterstützte Version von Gremlin: `3.7.1`
- Die neueste unterstützte Version von Gremlin: `3.7.1`
- openCypher-Version: `Neptune-9.0.20190305-1.0`

- SPARQL-Version: `1.1`

Upgrade-Pfade auf Engine-Version 1.3.3.0

Sie können von der Engine-Version 1.2.0.0 oder höher auf diese Version aktualisieren.

Upgrade auf diesen Release

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.3.3.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.3.3.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Statt `--apply-immediately` können Sie `--no-apply-immediately` angeben. Um ein Hauptversions-Upgrade durchzuführen, ist der allow-major-version-upgrade Parameter erforderlich. Stellen Sie außerdem sicher, dass Sie die Engine-Version angeben, da Ihre Engine sonst möglicherweise auf eine andere Version aktualisiert wird.

Wenn Ihr Cluster eine benutzerdefinierte Cluster-Parametergruppe verwendet, müssen Sie diesen Parameter einschließen, um ihn zu anzugeben:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Ebenso sollte für Instances im Cluster, die eine benutzerdefinierte DB-Parametergruppe verwenden, dieser Parameter eingeschlossen werden, um ihn zu spezifizieren:

```
--db-instance-parameter-group-name (name of the custom instance parameter group)
```

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
>   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.
> ```
>
> Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine, Version 1.3.2.1 (20.06.2024)

Ab 20.06.2024 wird die Engine-Version 1.3.2.1 allgemein eingesetzt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

> **Note**
>
> Mit der Engine-Version 1.3.0.0 wurde ein neues Format für benutzerdefinierte Parametergruppen und benutzerdefinierte Cluster-Parametergruppen eingeführt. Wenn Sie also von einer Engine-Version vor 1.3.0.0 auf Engine-Version 1.3.0.0 oder höher aktualisieren, müssen Sie alle vorhandenen benutzerdefinierten Parametergruppen und benutzerdefinierten Cluster-Parametergruppen mithilfe der Parametergruppenfamilie neu erstellen. `neptune1.3` In früheren Versionen wurde die Parametergruppenfamilie `neptune1` oder `neptune1.2` verwendet. Diese Parametergruppen funktionieren nicht mit Version 1.3.0.0 und höher. Ebenso sollten Sie Cluster-Parametergruppen der Version 1.4.0.0 für Engine-Versionen 1.4.0.0 und höher verwenden. Weitere Informationen finden Sie unter Amazon-Neptune-Parametergruppen.

> **Warning**
>
> Mit der Engine-Version 1.3.2.1 wurden einige potenzielle Probleme eingeführt, die Sie beachten sollten. Weitere Informationen finden Sie im Behebung von Problemen in Version 1.3.2.1 folgenden Abschnitt.

In dieser Engine-Version behobene Fehler

OpenCypher-Korrekturen

- In der Cache-Funktion für Abfragepläne wurde ein Fehler für parametrisierte Abfragen entdeckt, die eine innere `WITH` Klausel mit `SKIP` und `LIMIT` als Parameter enthalten. Die SKIP/LIMIT Werte waren nicht richtig parametrisiert, sodass nachfolgende Ausführungen desselben zwischengespeicherten Abfrageplans mit unterschiedlichen Parameterwerten immer noch dieselben Ergebnisse wie bei der ersten Ausführung zurückgaben. Das wurde behoben.

```
# 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
  } ]
}%
```

- Es wurde ein Fehler behoben, bei dem parametrisierte Mutationsabfragen einen auslösen, `InternalFailureException` wenn der übergebene Parameter noch nicht in der Datenbank vorhanden ist.
- Es wurde ein Fehler behoben, bei dem parametrisierte Bolt-Abfragen hängen blieben, nachdem sie bei der Bereinigung der Abfrageressourcen auf eine Race-Bedingung gestoßen waren.

Die Änderungen in 1.3.2.1 wurden von 1.3.2.0 übernommen

Die Verbesserungen wurden aus der Engine-Version 1.3.2.0 übernommen

Allgemeine Verbesserungen

- Support für TLS Version 1.3, einschließlich der Verschlüsselungssammlungen TLS_AES_128_GCM_ und SHA256 TLS_AES_256_GCM_. SHA384 TLS 1.3 ist eine Option — TLS 1.2 ist immer noch das Minimum.
- Die erweiterte OpenCypher-Unterstützung für das Dateime-Format befindet sich für diese Version im lab_mode. Wir empfehlen Ihnen, es zu testen.

Gremlin-Verbesserungen

- TinkerPop 3.7.x-Aktualisierung

  - Bietet eine große Erweiterung der Gremlin-Sprache.
    - Neue Schritte für die Verarbeitung von Zeichenketten, Listen und Daten.
    - Neue Syntax für die Angabe der Kardinalität innerhalb des `mergeV()` Schritts.
    - `union()`kann jetzt als Startschritt verwendet werden.
    - Weitere Informationen zu den Änderungen in 3.7.x finden Sie in der TinkerPop Upgrade-Dokumentation.
  - Beachten Sie beim Upgrade der Client-Gremlin-Sprachtreiber für Java, dass die Serializer-Klassen einige Umbenennungen rückgängig gemacht haben. Sie müssen die Paket- und Klassennamen in Ihren Konfigurationsdateien und im Code, falls angegeben, aktualisieren.
- `StrictTimeoutValidation`(nur bei Aktivierung über Labmode `StrictTimeoutValidation` by including`StrictTimeoutValidation=enabled`): Wenn der `StrictTimeoutValidation` Parameter den Wert von hat`enabled`, darf ein als Anforderungsoption oder als Abfragehinweis spezifizierter Timeout-Wert pro Abfrage den global in der Parametergruppe festgelegten Wert nicht überschreiten. In einem solchen Fall wirft Neptune eine. `InvalidParameterException` Diese Einstellung kann in einer Antwort auf dem `/status` Endpunkt bestätigt werden, wenn der Wert lautet`disabled`, und in den Neptun-Versionen 1.3.2.0 und 1.3.2.1 ist der Standardwert dieses Parameters. `Disabled`

Verbesserungen bei OpenCypher

- Die Amazon Neptune Engine Version 1.3.2.0 bietet im Vergleich zu früheren Engine-Versionen einen bis zu 9-mal schnelleren und 10-mal höheren Durchsatz für die OpenCypher-Abfrageleistung.
- Abfragen mit niedriger Latenz und Verbesserung der Durchsatzleistung: Allgemeine Leistungsverbesserungen für OpenCypher-Abfragen mit niedriger Latenz. Die neue Version verbessert auch den Durchsatz für solche Abfragen. Die Verbesserungen sind signifikanter, wenn parametrisierte Abfragen verwendet werden.
- Support für Query Plan Cache: Wenn eine Abfrage an Neptune gesendet wird, wird die Abfragezeichenfolge analysiert, optimiert und in einen Abfrageplan umgewandelt, der dann von der Engine ausgeführt wird. Anwendungen basieren häufig auf gängigen Abfragemustern, die mit unterschiedlichen Werten instanziiert werden. Der Abfrageplan-Cache kann die Gesamtlatenz reduzieren, indem er die Abfragepläne zwischenspeichert und so das Parsen und Optimieren für solche sich wiederholenden Muster vermeidet. Weitere Details finden Sie unter Plan-Cache in Amazon Neptune abfragen.

- Leistungsverbesserung für DISTINCT-Aggregationsabfragen.
- Leistungsverbesserung für Verknüpfungen mit NULL-Variablen.
- Leistungsverbesserung für Abfragen, bei denen das Prädikat „Nicht entspricht ID“ (Knoten/Beziehung) verwendet wird.
- Erweiterte Unterstützung für die Datetime-Funktionalität (Nur im Labormodus aktiviert durch Including). `DatetimeMillisecond` `DatetimeMillisecond=enabled` Weitere Informationen finden Sie unter Temporale Unterstützung in der Neptune OpenCypher-Implementierung (Neptune Analytics und Neptune Database 1.3.2.0 und höher).

Fehlerkorrekturen wurden aus der Engine-Version 1.3.2.0 übernommen

Allgemeine Verbesserungen

- Die NeptuneML-Fehlermeldung bei der Validierung des Zugriffs auf Graphlytics-Buckets wurde aktualisiert.

Korrekturen für Gremlin

- Fehlende Labelinformationen in der DFE-Abfrageübersetzung für Szenarien behoben, in denen Schritte, die nicht zum Pfad beitragen, Beschriftungen enthalten. Zum Beispiel:

```
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')
```

- Es wurde ein `NullPointerException` Fehler bei der Übersetzung von DFE-Abfragen behoben, der auftritt, wenn eine Abfrage in zwei DFE-Fragmenten ausgeführt wird und das erste Fragment für einen unbefriedigenden Knoten optimiert wurde. Zum Beispiel:

```
g.withSideEffect('Neptune#useDFE', true).
  V().
  has('name', 'doesNotExists').
  has("name", TextP.regex("mark.*")).
  inject(1).
  V().
```

```
  out().
  has('name', 'vadas')
```

- Es wurde ein Fehler behoben, durch den Neptune einen auslösen konnte`InternalFailureException`, wenn eine Abfrage den Modulator ValueTraversal inside by () enthält und die Eingabe Map ist. Zum Beispiel:

```
g.V().
  hasLabel("person").
  project("age", "name").by("age").by("name").
  order().by("age")
```

OpenCypher-Korrekturen

- Verbesserte UNWIND-Operationen (z. B. das Erweitern einer Werteliste auf einzelne Werte), um Situationen zu vermeiden, in denen der Arbeitsspeicher knapp wird (OOM). Zum Beispiel:

```
MATCH (n)-->(m)
WITH collect(m) AS list
UNWIND list AS m
RETURN m, list
```

- Die benutzerdefinierte ID-Optimierung bei mehreren MERGE-Operationen, bei denen die ID über UNWIND eingegeben wurde, wurde behoben. Zum Beispiel:

```
UNWIND [{nid: 'nid1', mid: 'mid1'}, {nid: 'nid2', mid: 'mid2'}] as ids
MERGE (n:N {`~id`: ids.nid})
MERGE (m:M {`~id`: ids.mid})
```

- Speicherexplosion bei der Planung komplexer Abfragen mit Eigenschaftszugriff und mehreren Hops mit bidirektionalen Beziehungen behoben. Zum Beispiel:

```
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
```

- Aggregationsabfragen mit Null als Gruppierung nach Variablen behoben. Zum Beispiel:

```
MATCH (n)
RETURN null AS group, sum(n.num) AS result
```

Fehlerkorrekturen für SPARQL

- Der SPARQL-Parser wurde korrigiert, um die Analysezeit für große Abfragen wie INSERT DATA mit vielen Triples und großen Tokens zu verbessern.

Behebung von Problemen in Version 1.3.2.1

- Abfragen, die numerische Filterwerte verwenden, können falsche Ergebnisse zurückgeben, wenn der Abfrageplan-Cache verwendet wird. Um das Problem zu vermeiden, verwenden Sie den Abfragehinweis, `QUERY:PLANCACHE "disabled"` um den Abfrageplan-Cache zu überspringen. Verwenden Sie beispielsweise:

```
USING QUERY:PLANCACHE "disabled"
MATCH (n:person)
WHERE n.yearOfBirth > $year
RETURN n

parameters={"year":1950}
```

- Abfragen, die denselben Parameternamen mehrfach verwenden, können mit dem Fehler fehlschlagen`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`. Überspringen Sie in solchen Fällen entweder den Abfrageplan-Cache wie oben oder duplizieren Sie die Parameter wie in diesem Beispiel:

```
MATCH (n:movie) WHERE n.runtime>=$minutes RETURN n
UNION
MATCH (n:show) WHERE n.duration>=$minutes RETURN n

parameters={"minutes":130}
```

Verwenden Sie den Hinweis `QUERY:PLANCACHE "disabled"` oder ändern Sie die Parameter:

```
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}
```

- Abfragen, die mit dem Bolt-Protokoll ausgeführt werden, können zu falschen Ergebnissen führen, wenn es sich bei der Abfrage um eine UNION- oder UNION ALL-Abfrage handelt. Um das Problem zu vermeiden, sollten Sie erwägen, die jeweilige Abfrage mit dem HTTP-Endpunkt auszuführen. Alternativ können Sie jeden Teil der Union separat ausführen, wenn Sie das Bolt-Protokoll verwenden.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.3.2.1 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit diesen Versionen in Abfragesprachen kompatibel ist:

- Die älteste unterstützte Version von Gremlin: `3.7.1`
- Die neueste unterstützte Version von Gremlin: `3.7.1`
- openCypher-Version: `Neptune-9.0.20190305-1.0`
- SPARQL-Version: `1.1`

Upgrade-Pfade auf Engine-Version 1.3.2.1

Sie können von der Engine-Version 1.2.0.0 oder höher auf diese Version aktualisieren.

Upgrade auf diesen Release

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.3.2.1 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.3.2.1 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Statt `--apply-immediately` können Sie `--no-apply-immediately` angeben. Um ein Hauptversions-Upgrade durchzuführen, ist der allow-major-version-upgrade Parameter erforderlich. Stellen Sie außerdem sicher, dass Sie die Engine-Version angeben, da Ihre Engine sonst möglicherweise auf eine andere Version aktualisiert wird.

Wenn Ihr Cluster eine benutzerdefinierte Cluster-Parametergruppe verwendet, müssen Sie diesen Parameter einschließen, um ihn zu anzugeben:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Ebenso sollte für Instances im Cluster, die eine benutzerdefinierte DB-Parametergruppe verwenden, dieser Parameter eingeschlossen werden, um ihn zu spezifizieren:

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
> 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.
```

Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine, Version 1.3.2.0 (10.06.2020)

Ab 2024-06-10 wird die Engine-Version 1.3.2.0 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

Note

Mit der Engine-Version 1.3.0.0 wurde ein neues Format für benutzerdefinierte Parametergruppen und benutzerdefinierte Cluster-Parametergruppen eingeführt. Wenn Sie also von einer Engine-Version vor 1.3.0.0 auf Engine-Version 1.3.0.0 oder höher aktualisieren, müssen Sie alle vorhandenen benutzerdefinierten Parametergruppen und benutzerdefinierten Cluster-Parametergruppen mithilfe der Parametergruppenfamilie neu erstellen. `neptune1.3` In früheren Versionen wurde die Parametergruppenfamilie `neptune1` oder `neptune1.2` verwendet. Diese Parametergruppen funktionieren nicht mit Version 1.3.0.0 und höher. Ebenso sollten Sie Cluster-Parametergruppen der Version 1.4.0.0 für Engine-Versionen 1.4.0.0 und höher verwenden. Weitere Informationen finden Sie unter Amazon-Neptune-Parametergruppen.

Warning

Mit der Engine-Version 1.3.2.0 wurden einige potenzielle Probleme eingeführt, die Sie beachten sollten. Weitere Informationen finden Sie im Behebung von Problemen in Version 1.3.2.0 folgenden Abschnitt.

Verbesserungen in dieser Engine-Version

Allgemeine Verbesserungen

- Support für TLS Version 1.3, einschließlich der Verschlüsselungssammlungen TLS_AES_128_GCM_ und SHA256 TLS_AES_256_GCM_. SHA384 TLS 1.3 ist eine Option — TLS 1.2 ist immer noch das Minimum.

Gremlin-Verbesserungen

- TinkerPop 3.7.x-Aktualisierung
  - Bietet eine große Erweiterung der Gremlin-Sprache.
    - Neue Schritte für die Verarbeitung von Zeichenketten, Listen und Daten.
    - Neue Syntax für die Angabe der Kardinalität innerhalb des Schritts. `mergeV()`
    - `union()`kann jetzt als Startschritt verwendet werden.
    - Weitere Informationen zu den Änderungen in 3.7.x finden Sie in der TinkerPop Upgrade-Dokumentation.
  - Beachten Sie beim Upgrade der Client-Gremlin-Sprachtreiber für Java, dass die Serializer-Klassen einige Umbenennungen rückgängig gemacht haben. Sie müssen die Paket- und Klassennamen in Ihren Konfigurationsdateien und im Code, falls angegeben, aktualisieren.
- `StrictTimeoutValidation`(nur bei Aktivierung über Labmode `StrictTimeoutValidation` by including`StrictTimeoutValidation=enabled`): Wenn der `StrictTimeoutValidation` Parameter den Wert von hat`enabled`, darf ein als Anforderungsoption oder als Abfragehinweis spezifizierter Timeout-Wert pro Abfrage den global in der Parametergruppe festgelegten Wert nicht überschreiten. In einem solchen Fall wirft Neptune eine. `InvalidParameterException` Diese Einstellung kann in einer Antwort auf dem `/status` Endpunkt bestätigt werden, wenn der Wert lautet`disabled`, und in Neptune Version 1.3.2.0 ist der Standardwert dieses Parameters. `Disabled`

Verbesserungen bei OpenCypher

- Die Amazon Neptune Engine Version 1.3.2.0 bietet im Vergleich zu früheren Engine-Versionen einen bis zu 9-mal schnelleren und 10-mal höheren Durchsatz für die OpenCypher-Abfrageleistung.

- Abfragen mit niedriger Latenz und Verbesserung der Durchsatzleistung: Allgemeine Leistungsverbesserungen für OpenCypher-Abfragen mit niedriger Latenz. Die neue Version verbessert auch den Durchsatz für solche Abfragen. Die Verbesserungen sind signifikanter, wenn parametrisierte Abfragen verwendet werden.
- Support für Query Plan Cache: Wenn eine Abfrage an Neptune gesendet wird, wird die Abfragezeichenfolge analysiert, optimiert und in einen Abfrageplan umgewandelt, der dann von der Engine ausgeführt wird. Anwendungen basieren häufig auf gängigen Abfragemustern, die mit unterschiedlichen Werten instanziiert werden. Der Abfrageplan-Cache kann die Gesamtlatenz reduzieren, indem er die Abfragepläne zwischenspeichert und so das Parsen und Optimieren für solche sich wiederholenden Muster vermeidet.
- Leistungsverbesserung für DISTINCT-Aggregationsabfragen.
- Leistungsverbesserung für Verknüpfungen mit NULL-Variablen.
- Leistungsverbesserung bei Abfragen, bei denen das Prädikat „Nicht entspricht ID“ (Knoten/Beziehung) verwendet wird.
- Erweiterte Unterstützung für die Datetime-Funktionalität (Nur im Labormodus aktiviert durch Including). `DatetimeMillisecond` `DatetimeMillisecond=enabled` Weitere Informationen finden Sie unter Temporale Unterstützung in der Neptune OpenCypher-Implementierung (Neptune Analytics und Neptune Database 1.3.2.0 und höher).

In dieser Engine-Version behobene Fehler

Allgemeine Verbesserungen

- Die NeptuneML-Fehlermeldung bei der Validierung des Zugriffs auf Graphlytics-Buckets wurde aktualisiert.

Korrekturen für Gremlin

- Fehlende Labelinformationen in der DFE-Abfrageübersetzung für Szenarien behoben, in denen Schritte, die nicht zum Pfad beitragen, Beschriftungen enthalten. Zum Beispiel:

```
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')
```

- Es wurde ein `NullPointerException` Fehler bei der Übersetzung von DFE-Abfragen behoben, der auftritt, wenn eine Abfrage in zwei DFE-Fragmenten ausgeführt wird und das erste Fragment für einen unbefriedigenden Knoten optimiert wurde. Zum Beispiel:

```
g.withSideEffect('Neptune#useDFE', true).
  V().
  has('name', 'doesNotExists').
  has("name", TextP.regex("mark.*")).
  inject(1).
  V().
  out().
  has('name', 'vadas')
```

- Es wurde ein Fehler behoben, durch den Neptune einen auslösen konnte`InternalFailureException`, wenn eine Abfrage den Modulator ValueTraversal inside by () enthält und die Eingabe Map ist. Zum Beispiel:

```
g.V().
  hasLabel("person").
  project("age", "name").by("age").by("name").
  order().by("age")
```

OpenCypher-Korrekturen

- Verbesserte UNWIND-Operationen (z. B. das Erweitern einer Werteliste auf einzelne Werte), um Situationen zu vermeiden, in denen der Arbeitsspeicher knapp wird (OOM). Zum Beispiel:

```
MATCH (n)-->(m)
WITH collect(m) AS list
UNWIND list AS m
RETURN m, list
```

- Die benutzerdefinierte ID-Optimierung bei mehreren MERGE-Operationen, bei denen die ID über UNWIND eingegeben wurde, wurde behoben. Zum Beispiel:

```
UNWIND [{nid: 'nid1', mid: 'mid1'}, {nid: 'nid2', mid: 'mid2'}] as ids
MERGE (n:N {`~id`: ids.nid})
MERGE (m:M {`~id`: ids.mid})
```

- Speicherexplosion bei der Planung komplexer Abfragen mit Eigenschaftszugriff und mehreren Hops mit bidirektionalen Beziehungen behoben. Zum Beispiel:

```
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
```

- Aggregationsabfragen mit Null als Gruppierung nach Variablen behoben. Zum Beispiel:

```
MATCH (n)
RETURN null AS group, sum(n.num) AS result
```

Fehlerkorrekturen für SPARQL

- Der SPARQL-Parser wurde korrigiert, um die Analysezeit für große Abfragen wie INSERT DATA mit vielen Triples und großen Tokens zu verbessern.

Behebung von Problemen in Version 1.3.2.0

- Bei Version 1.3.2.0 haben wir ein Problem im Abfrageplan-Cache festgestellt, wenn `skip` oder in einer inneren `WITH` Klausel verwendet `limit` wird und parametrisiert ist. Zum Beispiel:

```
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}
```

In diesem Fall werden die Parameterwerte für Skip und Limit aus dem ersten Plan auch auf nachfolgende Abfragen angewendet, was zu unerwarteten Ergebnissen führt.

Schadensbegrenzung

Um dieses Problem zu vermeiden, fügen Sie den Abfragehinweis hinzu, `QUERY:PLANCACHE "disabled"` wenn Sie eine Abfrage einreichen, die eine parametrisierte and/or Skip-Limit-Unterklausel enthält. Alternativ können Sie die Werte in der Abfrage fest codieren.

Option 1: Verwenden des Abfragehinweises zum Deaktivieren des Plan-Caches:

```
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: Verwendung hartcodierter Werte für Skip und Limit:

```
MATCH (n:Person)
WHERE n.age > $age
WITH n skip 2 LIMIT 3
RETURN n.name, n.age

parameters={"age": 21}
```

- Abfragen, die numerische Filterwerte verwenden, können zu falschen Ergebnissen führen, wenn der Abfrageplan-Cache verwendet wird. Um das Problem zu vermeiden, verwenden Sie den Abfragehinweis, `QUERY:PLANCACHE "disabled"` um den Abfrageplan-Cache zu überspringen. Verwenden Sie beispielsweise:

```
USING QUERY:PLANCACHE "disabled"
MATCH (n:person)
WHERE n.yearOfBirth > $year
RETURN n

parameters={"year":1950}
```

- Abfragen, die denselben Parameternamen mehrfach verwenden, können mit dem Fehler fehlschlagen`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`. Überspringen Sie in solchen Fällen entweder den Abfrageplan-Cache wie oben oder duplizieren Sie die Parameter wie in diesem Beispiel:

```
MATCH (n:movie) WHERE n.runtime>=$minutes RETURN n
UNION
MATCH (n:show) WHERE n.duration>=$minutes RETURN n

parameters={"minutes":130}
```

  Verwenden Sie den Hinweis `QUERY:PLANCACHE "disabled"` oder ändern Sie die Parameter:

```
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}
```

- Abfragen, die mit dem Bolt-Protokoll ausgeführt werden, können zu falschen Ergebnissen führen, wenn es sich bei der Abfrage um eine UNION- oder UNION ALL-Abfrage handelt. Um das Problem zu vermeiden, sollten Sie erwägen, die jeweilige Abfrage mit dem HTTP-Endpunkt auszuführen. Alternativ können Sie jeden Teil der Union separat ausführen, wenn Sie das Bolt-Protokoll verwenden.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.3.2.0 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprache kompatibel ist:

- Die älteste unterstützte Version von Gremlin: `3.7.1`
- Die neueste unterstützte Version von Gremlin: `3.7.1`
- openCypher-Version: `Neptune-9.0.20190305-1.0`
- SPARQL-Version: `1.1`

Upgrade-Pfade auf Engine-Version 1.3.2.0

Sie können von der Engine-Version 1.2.0.0 oder höher auf diese Version aktualisieren.

Upgrade auf diesen Release

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.3.2.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.3.2.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Statt `--apply-immediately` können Sie `--no-apply-immediately` angeben. Um ein Hauptversions-Upgrade durchzuführen, ist der allow-major-version-upgrade Parameter erforderlich. Stellen Sie außerdem sicher, dass Sie die Engine-Version angeben, da Ihre Engine sonst möglicherweise auf eine andere Version aktualisiert wird.

Wenn Ihr Cluster eine benutzerdefinierte Cluster-Parametergruppe verwendet, müssen Sie diesen Parameter einschließen, um ihn zu anzugeben:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Ebenso sollte für Instances im Cluster, die eine benutzerdefinierte DB-Parametergruppe verwenden, dieser Parameter eingeschlossen werden, um ihn zu spezifizieren:

```
--db-instance-parameter-group-name (name of the custom instance parameter group)
```

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:

```
  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.
```

Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine, Version 1.3.1.0 (06.03.2024)

Ab 2024-03-06 wird die Engine-Version 1.3.1.0 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

Note

Mit der Engine-Version 1.3.0.0 wurde ein neues Format für benutzerdefinierte Parametergruppen und benutzerdefinierte Cluster-Parametergruppen eingeführt. Wenn Sie also von einer Engine-Version vor 1.3.0.0 auf Engine-Version 1.3.0.0 oder höher aktualisieren, müssen Sie alle vorhandenen benutzerdefinierten Parametergruppen und benutzerdefinierten Cluster-Parametergruppen mithilfe der Parametergruppenfamilie neu erstellen. `neptune1.3` In früheren Versionen wurde die Parametergruppenfamilie `neptune1` oder `neptune1.2` verwendet. Diese Parametergruppen funktionieren nicht mit Version 1.3.0.0 und höher. Ebenso sollten Sie Cluster-Parametergruppen der Version 1.4.0.0 für Engine-Versionen 1.4.0.0 und höher verwenden. Weitere Informationen finden Sie unter Amazon-Neptune-Parametergruppen.

Verbesserungen in dieser Engine-Version

Allgemeine Verbesserungen

- Neptune hat die in profile/explain angezeigte Warnung verbessert.
- Veraltete NIST-EC-Kurven wurden aus den benannten Standardgruppen entfernt, die bei der TLS-Verhandlung verwendet wurden. Die entfernten Kurven sind sect409k1, sect409r1 und sect571k1.

Gremlin-Verbesserungen

- Die Berechnung der DFE-Statistiken wurde verbessert, um zu viele serverlose Instanzen zu vermeiden. NCUs
- Verbesserung der Gremlin-Leistung für WITHIN.

In dieser Engine-Version wurden Fehler behoben

Korrekturen für Gremlin

- Verschiedene Verbesserungen der Gremlin DFE-Abfragepläne.
- Bugfix für Gremlin-Abfragen mit optionaler Durchquerung, z. B. für Abfragen der Form `G.v () .hasLabel ('person') .group () .by (id ()) .by (id ()) .by (__.in ('friend') .id () .fold ()) `, bei denen keine Personen ohne Freundesgrenzen gruppiert wurden.
- Es wurde ein Fehler behoben, bei dem Gremlin-Abfragen, die Koaleszenzschritte innerhalb von by Modulatoren enthielten, dazu führten, dass ein Fehler zurückgegeben wurde, wenn sie mit der DFE-Engine ausgeführt wurden.
- Es wurde ein Fehler behoben, der verhinderte, dass schreibgeschützte Abfragen, die in einer Gremlin-Sitzung ausgeführt wurden, nicht funktionierten, wenn sie mit einer Read Replica verbunden waren.
- Bugfix, bei dem IAM-ARN nicht im Audit-Log für eine erfolgreiche erste Websocket-Verbindungsanfrage für Gremlin vorhanden war.
- Schritt zusammenführen, Schrittabdeckung mit DFE identifizieren.
- Optimierung der Merkmalssätze für ganze DFE-Pläne.

OpenCypher-Korrekturen

- Fehlerkorrekturen in der OpenCypher SET-Klausel, um die Einstellung von Ausdrücken ohne Variablen zu ermöglichen (d. h.: match (n:Test) set (Fall, wenn n.prop = 2, dann n end) .prop = 3 return n.prop.
- Bugfix für fehlgeschlagene OpenCypher-Abfragen mit Aggregation und Sortierung nach.
- Der UNDOWN einer großen Liste mit statischen Maps wurde verbessert.
- Fehlerkorrektur: OpenCypher MERGE-Abfrage unter Verwendung einer benutzerdefinierten ID mit doppelten Werten.

Fehlerkorrekturen für SPARQL

- Ein SPARQL-Fehler bezüglich des Variablenbereichs in optionalen Mustern wurde behoben.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.3.1.0 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit diesen Versionen in Abfragesprachen kompatibel ist:

- Die älteste unterstützte Version von Gremlin: `3.6.2`
- Die neueste unterstützte Version von Gremlin: `3.6.5`
- openCypher-Version: `Neptune-9.0.20190305-1.0`
- SPARQL-Version: `1.1`

Upgrade-Pfade auf Engine-Version 1.3.1.0

Sie können von der Engine-Version 1.2.0.0 oder höher auf diese Version aktualisieren.

Upgrade auf diesen Release

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.3.1.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.3.1.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Statt `--apply-immediately` können Sie `--no-apply-immediately` angeben. Um ein Hauptversions-Upgrade durchzuführen, ist der allow-major-version-upgrade Parameter erforderlich. Stellen Sie außerdem sicher, dass Sie die Engine-Version angeben, da Ihre Engine sonst möglicherweise auf eine andere Version aktualisiert wird.

Wenn Ihr Cluster eine benutzerdefinierte Cluster-Parametergruppe verwendet, müssen Sie diesen Parameter einschließen, um ihn zu anzugeben:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Ebenso sollte für Instances im Cluster, die eine benutzerdefinierte DB-Parametergruppe verwenden, dieser Parameter eingeschlossen werden, um ihn zu spezifizieren:

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
> 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.
> ```
>
> Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine-Version 1.3.0.0 (15.11.2023)

Seit dem 15.11.2023 wird die Engine-Version 1.3.0.0 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

> **Note**
>
> Mit der Engine-Version 1.3.0.0 wurde ein neues Format für benutzerdefinierte Parametergruppen und benutzerdefinierte Cluster-Parametergruppen eingeführt. Wenn Sie also von einer Engine-Version vor 1.3.0.0 auf Engine-Version 1.3.0.0 oder höher aktualisieren, müssen Sie alle vorhandenen benutzerdefinierten Parametergruppen und benutzerdefinierten Cluster-Parametergruppen mithilfe der Parametergruppenfamilie neu erstellen. `neptune1.3` In früheren Versionen wurde die Parametergruppenfamilie `neptune1` oder `neptune1.2` verwendet. Diese Parametergruppen funktionieren nicht mit Version 1.3.0.0 und höher. Ebenso sollten Sie Cluster-Parametergruppen der Version 1.4.0.0 für Engine-Versionen 1.4.0.0 und höher verwenden. Weitere Informationen finden Sie unter Amazon-Neptune-Parametergruppen.

Neue Features in dieser Engine-Version

- Die Neptune-Daten-API wurde veröffentlicht.

  Die Amazon-Neptune-Daten-API bietet SDK-Unterstützung für über 40 Datenoperationen in Neptune, darunter Laden von Daten, Ausführung von Abfragen, Datenabfragen und Machine Learning. Sie unterstützt die drei Neptune-Abfragesprachen (Gremlin, openCypher und SPARQL) und ist in allen SDK-Sprachen verfügbar. Sie signiert automatisch API-Anfragen und vereinfacht die Integration von Neptune in Anwendungen erheblich.
- Unterstützung für die Integration von OpenSearchServerless mit Neptune hinzugefügt.

Verbesserungen in dieser Engine-Version

Verbesserungen der Neptune-Engine-Updates

Neptune hat die Art und Weise geändert, wie Engine-Updates veröffentlicht werden, sodass Sie mehr Kontrolle über den Aktualisierungsprozess haben. Anstatt Patches für unwichtigere Änderungen zu veröffentlichen, veröffentlicht Neptune nun Nebenversionen, die über das `AutoMinorVersionUpgrade`-Instance-Feld gesteuert werden können und über die Sie Benachrichtigungen erhalten können, indem Sie das `RDS-EVENT-0156`-Ereignis abonnieren.

Weitere Informationen zu diesen Änderungen finden Sie unter Warten eines Amazon-Neptune-DB-Clusters.

Verbesserung der Verschlüsselung während der Übertragung

Neptune unterstützt die folgenden Cipher-Suites nicht mehr:

- `TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA`
- `TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA`

Neptune unterstützt nur die folgenden starken Cipher-Suites mit 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`

Verbesserungen bei Gremlin

- In der DFE-Engine wurde die Unterstützung für die folgenden Gremlin-Schritte hinzugefügt.
  - `FoldStep`
  - `GroupStep`
  - `GroupCountStep`
  - `TraversalMapStep`
  - `UnfoldStep`
  - `LabelStep`
  - `PropertyKeyStep`
  - `PropertyValueStep`
  - `AndStep`

  - `OrStep`
  - `ConstantStep`
  - `CountGlobalStep`
- Optimierte Gremlin-DFE-Abfragepläne, um vollständige Vertex-Scans bei Verwendung der `by()`-Modulation zu vermeiden.
- Deutlich verbesserte Leistung bei Abfragen mit niedriger Kardinalität und geringer Latenz.
- DFE-Unterstützung für Filterprädikate hinzugefügt. TinkerPop `0r`
- Verbesserte DFE-Unterstützung für die Traversierung von Filtern auf demselben Schlüssel bei Abfragen wie der folgenden:

```
g.withSideEffect("Neptune#useDFE", true)
 .V()
 .has('name', 'marko')
 .and(
   or(
     has('name', eq("marko")),
     has('name', eq("vardas"))
   )
 )
```

- Die Fehlerbehandlung für den `fail()`-Schritt wurde verbessert.

Verbesserungen bei OpenCypher

- Deutlich verbesserte Leistung bei Abfragen mit niedriger Kardinalität und geringer Latenz.
- Verbesserte Leistung bei der Abfrageplanung, wenn die Abfrage viele Knotentypen enthält.
- Reduzierte Latenz bei allen VLP-Abfragen.
- Verbesserte Leistung durch das Entfernen redundanter Pipeline-Joins für Abfragen mit einfachen Knotenmustern.
- Verbesserte Leistung für Abfragen, die Multi-Hop-Muster mit Zyklen enthalten, z. B.:

```
MATCH (n)-->()-->()-->(m)
RETURN n m
```

Verbesserungen bei SPARQL

- Es wurde ein neuer SPARQL-Operator eingeführt: `PipelineHashIndexJoin`.
- Verbesserte Leistung bei der URI-Validierung für SPARQL-Abfragen.
- Verbesserte Leistung bei SPARQL-Volltext-Suchabfragen durch Batch-Auflösung von Wörterbuchbegriffen.

In diesem Engine-Version behobene Fehler

Korrekturen für Gremlin

- Es wurde ein Gremlin-Fehler behoben, durch den es zu einem Transaktionsleck kam, wenn der Endpunkt des Gremlin-Abfragestatus auf Abfragen mit Prädikaten in untergeordneten Traversierungen für Schritte überprüft wurde, die nicht nativ in der DFE-Engine verarbeitet werden.
- Es wurde ein Gremlin-Fehler behoben, bei dem `valueMap()` in der DFE-Engine bei `by()`-Traversierungen nicht optimiert wurde.
- Es wurde ein Gremlin-Fehler behoben, durch den ein an `UnionStep` angefügtes Schritt-Label nicht an das letzte Pfadelement seiner untergeordneten Traversierungen weitergegeben wurde.
- Es wurde ein Gremlin-Fehler behoben, bei dem eine Abfrage fehlschlug, weil sie zu viele TinkerPop Schritte enthielt, und dann nicht bereinigt wurde.
- Es wurde ein Gremlin-Fehler behoben, bei dem in den Schritten `mergeV` und `mergeE` eine `NullPointerException` ausgelöst wurde.
- Es wurde ein Gremlin-Fehler behoben, bei dem Zeichenkettenausgaben von `order()` nicht richtig sortiert wurden, wenn einige von ihnen ein Leerzeichen enthielten.
- Es wurde ein Problem mit der Gremlin-Korrektheit behoben, das auftrat, wenn der `valueMap`-Schritt in der DFE-Engine verarbeitet wurde.
- Es wurde ein Problem mit der Gremlin-Korrektheit behoben, das auftrat, wenn `GroupStep` oder `GroupCountStep` in einer Schlüsseltraversierung verschachtelt war.

OpenCypher-Korrekturen

- Es wurde ein OpenCypher-Fehler behoben, der die Fehlerbehandlung bei NULL-Zeichen betraf.
- Es wurde ein Fehler bei der Bolt-Transaktionsverarbeitung von openCypher behoben.

Fehlerkorrekturen für SPARQL

- Es wurde ein SPARQL-Fehler behoben, bei dem Werte in rekursiven Funktionen nicht richtig aufgelöst wurden.
- Es wurde ein SPARQL-Fehler behoben, der beim Einfügen von vielen Werten durch die `VALUES`-Klausel zu Leistungseinbußen führte.
- Es wurde ein SPARQL-Fehler behoben, bei dem ein Aufruf des `REGEX`-Operators in einem Literal mit Sprachmarkierung nie erfolgreich war.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.3.0.0 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit den folgenden Versionen von Abfragesprachen kompatibel ist:

- Die älteste unterstützte Version von Gremlin: `3.6.2`
- Die neueste unterstützte Version von Gremlin: `3.6.4`
- openCypher-Version: `Neptune-9.0.20190305-1.0`
- SPARQL-Version: `1.1`

Upgrade-Pfade zum Engine-Release 1.3.0.0

Sie können von der Engine-Version 1.2.0.0 oder höher auf diese Version aktualisieren.

Upgrade auf diesen Release

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.3.0.0 \
    --allow-major-version-upgrade \
```

```
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.3.0.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Statt `--apply-immediately` können Sie `--no-apply-immediately` angeben. Um ein Hauptversions-Upgrade durchzuführen, ist der allow-major-version-upgrade Parameter erforderlich. Stellen Sie außerdem sicher, dass Sie die Engine-Version angeben, da Ihre Engine sonst möglicherweise auf eine andere Version aktualisiert wird.

Wenn Ihr Cluster eine benutzerdefinierte Cluster-Parametergruppe verwendet, müssen Sie diesen Parameter einschließen, um ihn zu anzugeben:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Ebenso sollte für Instances im Cluster, die eine benutzerdefinierte DB-Parametergruppe verwenden, dieser Parameter eingeschlossen werden, um ihn zu spezifizieren:

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue

Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
> 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.
> ```
>
> Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine Version 1.2.1.2 (05.08.2020)

Ab 2024-08-05 wird die Engine-Version 1.2.1.2 allgemein eingesetzt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

Note

Bei einem Upgrade von einer Engine-Version vor 1.2.0.0:

- Mit der Engine-Version 1.2.0.0 wurde ein neues Format für benutzerdefinierte Parametergruppen und benutzerdefinierte Cluster-Parametergruppen eingeführt. Wenn Sie also von einer Engine-Version vor 1.2.0.0 auf Engine-Version 1.2.0.0 oder höher aktualisieren, müssen Sie alle vorhandenen benutzerdefinierten Parametergruppen und benutzerdefinierten Cluster-Parametergruppen mithilfe der Parametergruppenfamilie `neptune1.2` neu erstellen. In früheren Versionen wurde die Parametergruppenfamilie `neptune1` verwendet und diese Parametergruppen funktionieren nicht mit Version 1.2.0.0 und höher. Weitere Informationen finden Sie unter Amazon-Neptune-Parametergruppen.
- Mit der Engine-Version 1.2.0.0 wurde auch ein neues Format für Undo-Protokolle eingeführt. Daher müssen alle Undo-Logs, die von einer früheren Engine-Version erstellt wurden, gelöscht werden und die `UndoLogsListSize` CloudWatch Metrik muss auf Null fallen, bevor ein Upgrade von einer Version vor 1.2.0.0 gestartet werden kann. Wenn beim Versuch, ein Update zu starten, zu viele Undo-Protokolldatensätze (200.000 oder mehr) vorhanden sind, kann es beim Upgrade-Versuch zu einer Zeitüberschreitung kommen, während auf den Abschluss des Löschvorgangs der Undo-Protokolle gewartet wird.

  Sie können die Löschgeschwindigkeit beschleunigen, indem Sie die Writer-Instance des Clusters aktualisieren, in der das Löschen stattfindet. Wenn Sie dies tun, bevor Sie versuchen, das Upgrade durchzuführen, kann die Anzahl der Undo-Logs vor dem Start reduziert werden. Wenn Sie die Größe des Writers auf einen 24XL-Instance-Typ erhöhen, kann sich Ihre Löschrate auf mehr als eine Million Datensätze pro Stunde erhöhen.

  Wenn die `UndoLogsListSize` CloudWatch Metrik extrem umfangreich ist, kann Ihnen die Eröffnung eines Support-Falls dabei helfen, zusätzliche Strategien zu finden, um sie zu reduzieren.
- Schließlich wurde mit Version 1.2.0.0 eine bahnbrechende Änderung eingeführt, die sich auf früheren Code auswirkt, der das Bolt-Protokoll mit IAM-Authentifizierung verwendete. Ab Version 1.2.0.0 benötigt Bolt einen Ressourcenpfad für die IAM-Signatur. In Java könnte das Festlegen des Ressourcenpfads so aussehen:

`request.setResourcePath("/openCypher"));`. In anderen Sprachen kann `/openCypher` an den Endpunkt-URI angehängt werden. Beispiele finden Sie unter Verwenden des Bolt-Protokolls.

Warning

Es wurde ein Problem mit dem SPARQL 1.1 Graph Store HTTP Protocol (GSP) festgestellt, das unter bestimmten Bedingungen auftreten kann, wenn GSP mit aktionsbasierten Autorisierungsrichtlinien verwendet wird. Wenn Sie das SPARQL 1.1 Graph Store HTTP-Protokoll mit aktionsbasierten Autorisierungsrichtlinien verwenden, empfehlen wir, auf die neueste kleinere Neptune-Engine-Version (mindestens 1.2.1.2) zu aktualisieren, die eine Lösung für dieses Problem enthält.

In diesem Engine-Version behobene Fehler

Fehlerkorrekturen für SPARQL

- Es wurde ein Problem mit dem SPARQL 1.1 Graph Store HTTP Protocol (GSP) behoben, das unter bestimmten Bedingungen auftreten kann, wenn GSP mit aktionsbasierten Autorisierungsrichtlinien verwendet wird.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.2.1.2 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit diesen Versionen in Abfragesprachen kompatibel ist:

- Die älteste unterstützte Version von Gremlin: `3.6.2`
- Die neueste unterstützte Version von Gremlin: `3.6.2`
- openCypher-Version: `Neptune-9.0.20190305-1.0`
- SPARQL-Version: `1.1`

Aktualisieren Sie die Pfade auf Engine-Version 1.2.1.2

Sie können von der Engine-Version 1.2.0.0 oder höher auf diese Version aktualisieren.

Upgrade auf diesen Release

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.1.2 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.1.2 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Statt `--apply-immediately` können Sie `--no-apply-immediately` angeben. Um ein Upgrade auf eine Hauptversion durchzuführen, ist der allow-major-version-upgrade Parameter erforderlich. Stellen Sie außerdem sicher, dass Sie die Engine-Version angeben, da Ihre Engine sonst möglicherweise auf eine andere Version aktualisiert wird.

Wenn Ihr Cluster eine benutzerdefinierte Cluster-Parametergruppe verwendet, müssen Sie diesen Parameter einschließen, um ihn zu anzugeben:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Ebenso sollte für Instances im Cluster, die eine benutzerdefinierte DB-Parametergruppe verwenden, dieser Parameter eingeschlossen werden, um ihn zu spezifizieren:

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
> 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.
```

Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine Version 1.2.1.1 (11.03.2024)

Ab dem 11.03.2022 wird die Engine-Version 1.2.1.1 allgemein eingesetzt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

Note

Bei einem Upgrade von einer Engine-Version vor 1.2.0.0:

- Mit der Engine-Version 1.2.0.0 wurde ein neues Format für benutzerdefinierte Parametergruppen und benutzerdefinierte Cluster-Parametergruppen eingeführt. Wenn Sie also von einer Engine-Version vor 1.2.0.0 auf Engine-Version 1.2.0.0 oder höher aktualisieren, müssen Sie alle vorhandenen benutzerdefinierten Parametergruppen und benutzerdefinierten Cluster-Parametergruppen mithilfe der Parametergruppenfamilie `neptune1.2` neu erstellen. In früheren Versionen wurde die Parametergruppenfamilie `neptune1` verwendet und diese Parametergruppen funktionieren nicht mit Version 1.2.0.0 und höher. Weitere Informationen finden Sie unter Amazon-Neptune-Parametergruppen.
- Mit der Engine-Version 1.2.0.0 wurde auch ein neues Format für Undo-Protokolle eingeführt. Daher müssen alle Undo-Logs, die von einer früheren Engine-Version erstellt wurden, gelöscht werden und die `UndoLogsListSize` CloudWatch Metrik muss auf Null fallen, bevor ein Upgrade von einer Version vor 1.2.0.0 gestartet werden kann. Wenn beim Versuch, ein Update zu starten, zu viele Undo-Protokolldatensätze (200.000 oder mehr) vorhanden sind, kann es beim Upgrade-Versuch zu einer Zeitüberschreitung kommen, während auf den Abschluss des Löschvorgangs der Undo-Protokolle gewartet wird.

Sie können die Löschgeschwindigkeit beschleunigen, indem Sie die Writer-Instance des Clusters aktualisieren, in der das Löschen stattfindet. Wenn Sie dies tun, bevor Sie versuchen, das Upgrade durchzuführen, kann die Anzahl der Undo-Logs vor dem Start reduziert werden. Wenn Sie die Größe des Writers auf einen 24XL-Instance-Typ erhöhen, kann sich Ihre Löschrate auf mehr als eine Million Datensätze pro Stunde erhöhen.

Wenn die `UndoLogsListSize` CloudWatch Metrik extrem umfangreich ist, kann Ihnen die Eröffnung eines Support-Falls dabei helfen, zusätzliche Strategien zu finden, um sie zu reduzieren.

- Schließlich wurde mit Version 1.2.0.0 eine bahnbrechende Änderung eingeführt, die sich auf früheren Code auswirkt, der das Bolt-Protokoll mit IAM-Authentifizierung verwendete. Ab Version 1.2.0.0 benötigt Bolt einen Ressourcenpfad für die IAM-Signatur. In Java könnte das Festlegen des Ressourcenpfads so aussehen: `request.setResourcePath("/openCypher"));`. In anderen Sprachen kann `/openCypher` an den Endpunkt-URI angehängt werden. Beispiele finden Sie unter Verwenden des Bolt-Protokolls.

Verbesserungen in dieser Engine-Version

Allgemeine Verbesserungen

Neptune hat die in profile/explain angezeigte Warnung verbessert.

Verbesserungen bei Gremlin

- Die Berechnung der DFE-Statistiken wurde verbessert, um zu viele serverlose Instanzen zu vermeiden. NCUs
- Verbesserung der Gremlin-Leistung für WITHIN.

In diesem Engine-Version behobene Fehler

Korrekturen für Gremlin

- Fehlerkorrekturen bei der Bestellung des Gremlin DFE-Engine-Abfrageplans.
- Bugfix mit einem out-of-memory Gremlin-Fehler, der ursprünglich als gemeldet wurde. InternalFailureException

- Fehlerkorrektur, bei der der IAM-ARN bei einer erfolgreichen ersten Websocket-Verbindungsanfrage nicht im Auditprotokoll vorhanden war.
- Bugfix für Gremlin-Abfragen mit aktivierter TinkerPop Sitzung, wenn Abfragen in einer Sitzung fehlschlagen, auch wenn sie alle schreibgeschützt sind und eine Verbindung zu einer Reader-Instanz herstellen.

OpenCypher-Korrekturen

- Fehlerkorrekturen in der OpenCypher SET-Klausel, um die Einstellung eines nicht variablen Ausdrucks zu ermöglichen (d. h.: match (n:Test) set (Fall, wenn n.prop = 2, dann n end) .prop = 3 gibt n.prop zurück.
- Bugfix für fehlgeschlagene OpenCypher-Abfragen mit Aggregation und Sortierung nach.
- Der UNDOWN einer großen Liste mit statischen Maps wurde verbessert.
- Fehlerbehebung bei der OpenCypher MERGE-Abfrage, die eine benutzerdefinierte ID mit doppelten Werten verwendet.

Fehlerkorrekturen für SPARQL

- Fehlerkorrekturen im SPARQL DFE-Abfrageplaner.
- Bugfix für SPARQL bei Verwendung mit den Schlüsselwörtern BIND und OPTIONAL.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.2.1.1 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit diesen Versionen in Abfragesprachen kompatibel ist:

- Die älteste unterstützte Version von Gremlin: `3.6.2`
- Die neueste unterstützte Version von Gremlin: `3.6.2`
- openCypher-Version: `Neptune-9.0.20190305-1.0`
- SPARQL-Version: `1.1`

Aktualisieren Sie die Pfade auf Engine-Version 1.2.1.1

Sie können von der Engine-Version 1.2.0.0 oder höher auf diese Version aktualisieren.

Upgrade auf diesen Release

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.1.1 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.1.1 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Statt `--apply-immediately` können Sie `--no-apply-immediately` angeben. Um ein Upgrade auf eine Hauptversion durchzuführen, ist der allow-major-version-upgrade Parameter erforderlich. Stellen Sie außerdem sicher, dass Sie die Engine-Version angeben, da Ihre Engine sonst möglicherweise auf eine andere Version aktualisiert wird.

Wenn Ihr Cluster eine benutzerdefinierte Cluster-Parametergruppe verwendet, müssen Sie diesen Parameter einschließen, um ihn zu anzugeben:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Ebenso sollte für Instances im Cluster, die eine benutzerdefinierte DB-Parametergruppe verwenden, dieser Parameter eingeschlossen werden, um ihn zu spezifizieren:

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
> 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.
```

Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine-Version 1.2.1.0 (08.03.2023)

Ab dem 08.03.2023 wird die Engine-Version 1.2.1.0 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

Note

Bei einem Upgrade von einer Engine-Version vor 1.2.0.0:

- Mit der Engine-Version 1.2.0.0 wurde ein neues Format für benutzerdefinierte Parametergruppen und benutzerdefinierte Cluster-Parametergruppen eingeführt. Wenn Sie also von einer Engine-Version vor 1.2.0.0 auf Engine-Version 1.2.0.0 oder höher aktualisieren, müssen Sie alle vorhandenen benutzerdefinierten Parametergruppen und benutzerdefinierten Cluster-Parametergruppen mithilfe der Parametergruppenfamilie `neptune1.2` neu erstellen. In früheren Versionen wurde die Parametergruppenfamilie `neptune1` verwendet und diese Parametergruppen funktionieren nicht mit Version 1.2.0.0 und höher. Weitere Informationen finden Sie unter Amazon-Neptune-Parametergruppen.
- Mit der Engine-Version 1.2.0.0 wurde auch ein neues Format für Undo-Protokolle eingeführt. Daher müssen alle Undo-Logs, die von einer früheren Engine-Version erstellt wurden, gelöscht werden und die `UndoLogsListSize` CloudWatch Metrik muss auf Null fallen, bevor ein Upgrade von einer Version vor 1.2.0.0 gestartet werden kann. Wenn beim Versuch, ein Update zu starten, zu viele Undo-Protokolldatensätze (200.000 oder mehr) vorhanden sind, kann es beim Upgrade-Versuch zu einer Zeitüberschreitung kommen, während auf den Abschluss des Löschvorgangs der Undo-Protokolle gewartet wird.

Sie können die Löschgeschwindigkeit beschleunigen, indem Sie die Writer-Instance des Clusters aktualisieren, in der das Löschen stattfindet. Wenn Sie dies tun, bevor Sie versuchen, das Upgrade durchzuführen, kann die Anzahl der Undo-Logs vor dem Start reduziert werden. Wenn Sie die Größe des Writers auf einen 24XL-Instance-Typ erhöhen, kann sich Ihre Löschrate auf mehr als eine Million Datensätze pro Stunde erhöhen.

Wenn die `UndoLogsListSize` CloudWatch Metrik extrem umfangreich ist, kann Ihnen die Eröffnung eines Support-Falls dabei helfen, zusätzliche Strategien zu finden, um sie zu reduzieren.

- Schließlich wurde mit Version 1.2.0.0 eine bahnbrechende Änderung eingeführt, die sich auf früheren Code auswirkt, der das Bolt-Protokoll mit IAM-Authentifizierung verwendete. Ab Version 1.2.0.0 benötigt Bolt einen Ressourcenpfad für die IAM-Signatur. In Java könnte das Festlegen des Ressourcenpfads so aussehen: `request.setResourcePath("/openCypher"));`. In anderen Sprachen kann `/openCypher` an den Endpunkt-URI angehängt werden. Beispiele finden Sie unter Verwenden des Bolt-Protokolls.

Nachfolgende Patch-Veröffentlichungen für dieses Version

- Release: 1.2.1.0.R2 (02.05.2023)
- Release: 1.2.1.0.R3 (13.06.2023)
- Release: 1.2.1.0.R4 (10.08.2023)
- Release: 1.2.1.0.R5 (02.09.2023)
- Release: 1.2.1.0.R6 (12.09.2023)
- Release: 1.2.1.0.R7 (06.10.2023)

Neue Features in dieser Engine-Version

- Unterstützung für TinkerPop 3.6.2 hinzugefügt, wodurch viele neue Gremlin-Funktionen wie die neuen Schritte`mergeV()`, `mergeE()element()`, und hinzugefügt werden. `fail()` Die Schritte `mergeV()` und `mergeE()` sind von besonderer Bedeutung, da sie eine lang erwartete deklarative Option für die Ausführung von upsert-ähnlichen Operationen bieten, was bestehende Codemuster erheblich vereinfachen und Gremlin lesbarer machen sollte. In der Version 3.6.x wurden außerdem Regex-Prädikate, eine neue Überladung des `property()`-Schrittes, die eine `Map` annimmt,

und eine umfassende Überarbeitung des `by()`-Modulationsverhaltens hinzugefügt, das in allen Schritten, die es verwenden, viel konsistenter ist.

Im TinkerPopÄnderungsprotokoll und auf der Upgrade-Seite finden Sie Informationen zu den Änderungen in Version 3.6 und zu den Dingen, die Sie beim Upgrade beachten sollten.

Wenn Sie `fold().coalesce(unfold(), <mutate>)` für bedingte Einfügungen arbeiten, empfehlen wir Ihnen, auf die neue `mergeV/E()` Syntax zu migrieren, die hier und hier beschrieben wird. Neptune verwendet ein engeres Sperrmuster für `Merge` als für`Coalesce`, wodurch die Anzahl gleichzeitiger Modifikationsausnahmen () reduziert werden kann. CMEs

Weitere Informationen zu den neuen Funktionen in dieser TinkerPop Version finden Sie in Stephen Mallettes Blog Exploring new features of Apache TinkerPop 3.6.x in Amazon Neptune.

- Es wurde Unterstützung für R6i-Instance-Typen hinzugefügt, die auf skalierbaren Intel-Xeon-Prozessoren der 3. Generation basieren. Diese eignen sich ideal für speicherintensive Workloads und bieten bis zu 15% mehr compute/price Leistung und bis zu 20% mehr Speicherbandbreite pro vCPU als vergleichbare R5-Instance-Typen.
- API-Endpunkte für Graphzusammenfassungen wurden sowohl für Eigenschaftsgraphen als auch für RDF-Graphen hinzugefügt, sodass Sie schnell einen zusammenfassenden Bericht über Ihren Graphen erhalten können.

  Bei Eigenschaftsgraphen (PG-Graphen) bietet die Graphübersichts-API eine schreibgeschützte Liste von Knoten- und Edgebezeichnungen und Eigenschaftsschlüsseln sowie die Anzahl der Knoten, Edges und Eigenschaften. Für RDF-Graphenbietet sie eine Liste von Klassen und Prädikatschlüsseln sowie die Anzahl der Quadrate, Subjekte und Prädikate.

  Die folgenden Änderungen gingen mit der neuen API für die Zusammenfassung von Graphen einher:

  - Eine neue Datenebenenaktion wurde hinzugefügt. GetGraphSummary
  - Es wurde ein neuer `rdf/statistics`-Endpunkt hinzugefügt, der den `sparql/statistics`-Endpunkt ersetzt, der jetzt veraltet ist.
  - Der Name des `summary`-Felds in der Statistikstatusantwort wurde zu `signatureInfo` geändert, um es nicht mit den zusammenfassenden Informationen des Graphen zu verwechseln. Frühere Engine-Versionen werden weiterhin `summary` in der JSON-Antwort verwenden.
  - Die Genauigkeit des `date`-Felds in der Statistikstatusantwort wurde von Minute auf Millisekunde geändert. Das vorherige Format lautete `2020-05-07T23:13Z` (Minutengenauigkeit), das neue Format ist `2023-01-24T00:47:43.319Z` (Millisekundengenauigkeit). Beide sind ISO 8601-

konform, aber diese Änderung kann den vorhandenen Code beschädigen, je nachdem, wie das Datum analysiert wird.

  - In der Workbench wurde ein neuer `%statistics`-Line Magic hinzugefügt, mit dem Sie Statistiken der DFE-Engine abrufen können.
  - In der Workbench wurde ein neuer `%summary`-Line Magic hinzugefügt, mit dem Sie Informationen zur Graphzusammenfassung abrufen können.
- Es wurde eine Protokollierung für langsame Abfragen hinzugefügt, um Abfragen zu protokollieren, deren Ausführung länger dauert als ein bestimmter Schwellenwert. Sie aktivieren und steuern die Protokollierung langsamer Abfragen mithilfe der beiden neuen dynamischen Parameter neptune_enable_slow_query_log und neptune_slow_query_log_threshold.
- Unterstützung für zwei dynamische Parameter wurde hinzugefügt, nämlich die neuen Cluster-Parameter neptune_enable_slow_query_log und neptune_slow_query_log_threshold. Wenn Sie einen dynamischen Parameter ändern, wird dieser Schritt sofort wirksam, ohne dass ein Neustart der Instance erforderlich ist.
- Es wurde eine Neptun-spezifische OpenCypher removeKeyFromMap () -Funktion hinzugefügt, die einen bestimmten Schlüssel aus einer Map entfernt und die resultierende neue Map zurückgibt.

Verbesserungen in dieser Engine-Version

- Die Unterstützung von Gremlin DFE wurde auf `limit`-Schritte mit lokalem Geltungsbereich erweitert.
- `by()`-Modulationsunterstützung für `DedupGlobalStep` für die Gremlin DFE-Engine hinzugefügt.
- DFE-Unterstützung für Gremlin `SelectStep` und `SelectOneStep` hinzugefügt.
- Leistungsverbesserungen und Korrekturkorrekturen für verschiedene Gremlin-Operatoren, darunter `repeat`, `coalesce`, `store` und `aggregate`.
- Verbesserte Leistung von openCypher-Abfragen mit `MERGE` und `OPTIONAL MATCH`.
- Verbesserte Leistung von openCypher-Abfragen, die eine Liste `UNWIND` von Zuordnungen mit Literalwerten beinhalten.
- Verbesserte Leistung von openCypher-Abfragen, die einen `IN`-Filter für `id` haben. Zum Beispiel:

```
MATCH (n) WHERE id(n) IN ['1', '2', '3'] RETURN n
```

- Es wurde die Möglichkeit hinzugefügt, den Basis-IRI für SPARQL-Abfragen mithilfe der BASE-Anweisung anzugeben (siehe Standard-Basis-IRI für Abfragen und Updates).

- Die Wartezeit bei der Ladeverarbeitung für Gremlin- und openCypher-Only-Edge-Massenladungen wurde verkürzt.
- Massenladungen wurden asynchron fortgesetzt, wenn Neptune neu gestartet wird, um lange Wartezeiten aufgrund von Amazon S3-Verbindungsproblemen zu vermeiden, bevor Wiederaufnahmeversuche fehlschlagen.
- Verbesserte Handhabung von SPARQL DESCRIBE-Abfragen, bei denen der DescribeMode-Abfragehinweis auf `"CBD"` (kurze, begrenzte Beschreibung) gesetzt ist und die eine große Anzahl leerer Knoten beinhalten.

In diesem Engine-Version behobene Fehler

- Es wurde ein openCypher-Fehler behoben, bei dem Abfragen die Zeichenfolge, `"null"`, anstelle eines Nullwerts in Bolt und SPARQL-JSON zurückgaben.
- Es wurde ein openCypher-Fehler im Listenverständnis behoben, der anstelle der für die Listenelemente angegebenen Werte einen Nullwert erzeugte.
- Es wurde ein openCypher-Fehler behoben, bei dem Bytewerte nicht korrekt serialisiert wurden.
- Es wurde ein Gremlin-Fehler in `UnionStep` behoben, der auftrat, wenn es sich bei der Eingabe um eine Edge handelte, die zu einem Scheitelpunkt innerhalb eines untergeordneten Traversalen führte.
- Es wurde ein Gremlin-Fehler behoben, der dazu führte, dass ein Schritt-Label, das mit einem Schritt verknüpft war, nicht korrekt auf den letzten Schritt jeder untergeordneten Traversale übertragen wurde.
- Es wurde ein Gremlin-Fehler für den `dedup`-Schritt behoben, bei dem Labels auf einen Schritt folgten, bei dem die an den `repeat`-Schritt angehängten Beschriftungen nicht für die weitere Verwendung in `dedup`-Abfragen verfügbar waren.
- Es wurde ein Gremlin-Fehler behoben, bei dem die Übersetzung des `repeat`-Schritts innerhalb eines `union`-Schritts aufgrund eines internen Fehlers fehlschlug.
- Behebung von Gremlin-Korrektheitsproblemen bei DFE-Abfragen mit `limit` als untergeordneter Traversal von Nicht-Union-Schritten durch Rückgriff auf Tinkerpop. Abfragen in einem Formular wie diesem sind betroffen:

```
g.withSideEffect('Neptune#useDFE', true).V().as("a").select("a").by(out().limit(1))
```

- Es wurde ein SPARQL-Fehler behoben, bei dem `SPARQL GRAPH`-Muster den durch eine `FROM NAMED`-Klausel bereitgestellten Datensatz nicht berücksichtigten.

- Es wurde ein SPARQL-Fehler behoben, bei dem SPARQL `DESCRIBE` mit einigen `FROM` and/or `FROM NAMED` Klauseln Daten aus dem Standarddiagramm nicht immer korrekt verwendete und manchmal eine Ausnahme auslöste. Siehe Verhalten von SPARQL DESCRIBE in Bezug auf das Standarddiagramm.
- Es wurde ein SPARQL-Fehler behoben, sodass die richtige Ausnahmemeldung zurückgegeben wurde, wenn Nullzeichen zurückgewiesen wurden.
- Es wurde ein SPARQL-Erklärungsfehler behoben, der Pläne betraf, die einen Operator enthielten. PipelinedHashIndexJoin
- Es wurde ein Fehler behoben, der dazu führte, dass ein interner Fehler ausgelöst wurde, wenn eine Abfrage, die einen konstanten Wert zurückgibt, gesendet wurde.
- Es wurde ein Problem mit der Deadlock-Detektorlogik behoben, das gelegentlich dazu führte, dass die Engine nicht reagierte.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.2.1.0 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprache kompatibel ist:

- Die älteste unterstützte Version von Gremlin: `3.6.2`
- Die neueste unterstützte Version von Gremlin: `3.6.2`
- openCypher-Version: `Neptune-9.0.20190305-1.1`
- SPARQL-Version: `1.1`

Upgrade-Pfade zum Engine-Release 1.2.1.0

Sie können ein manuelles Upgrade auf diese Version von jedem früheren Neptune-Engine-Release ab 1.1.0.0 durchführen.

> **Note**
>
> Ab Engine-Version 1.2.0.0 `1.2.0.0` müssen alle benutzerdefinierten Parametergruppen und benutzerdefinierten Cluster-Parametergruppen, die Sie mit früheren Engine-Versionen verwendet haben, jetzt mithilfe der Parametergruppenfamilie `neptune1.2` neu erstellt werden. In früheren Versionen wurde die Parametergruppenfamilie `neptune1` verwendet,

und diese Parametergruppen funktionieren ab Version `1.2.0.0` nicht mehr. Weitere Informationen finden Sie unter Amazon-Neptune-Parametergruppen.

Es wird kein automatisches Upgrade auf diesen großen Versions-Release durchgeführt.

Upgrade auf diesen Release

Amazon Neptune 1.2.1.0 ist jetzt allgemein verfügbar.

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.1.0 \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.1.0 ^
    --apply-immediately
```

Updates werden auf alle Instances in einem DB-Cluster gleichzeitig angewendet. Ein Update erfordert einen Datenbankneustart auf diesen Instances. Daher kommt es zu einer Ausfallzeit von 20–30 Sekunden bis zu mehreren Minuten. Anschließend können Sie die Nutzung Ihres DB-Clusters fortsetzen.

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
> 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.
> ```

Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine-Version 1.2.1.0.R7 (06.10.2023)

Seit dem 06.10.2023 wird die Engine-Version 1.2.1.0.R7 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

Note

Bei einem Upgrade von einer Engine-Version vor 1.2.0.0:

- Mit der Engine-Version 1.2.0.0 wurde ein neues Format für benutzerdefinierte Parametergruppen und benutzerdefinierte Cluster-Parametergruppen eingeführt. Wenn Sie also von einer Engine-Version vor 1.2.0.0 auf Engine-Version 1.2.0.0 oder höher aktualisieren, müssen Sie alle vorhandenen benutzerdefinierten Parametergruppen und benutzerdefinierten Cluster-Parametergruppen mithilfe der Parametergruppenfamilie `neptune1.2` neu erstellen. In früheren Versionen wurde die Parametergruppenfamilie `neptune1` verwendet und diese Parametergruppen funktionieren nicht mit Version 1.2.0.0 und höher. Weitere Informationen finden Sie unter Amazon-Neptune-Parametergruppen.
- Mit der Engine-Version 1.2.0.0 wurde auch ein neues Format für Undo-Protokolle eingeführt. Daher müssen alle Undo-Logs, die von einer früheren Engine-Version erstellt wurden, gelöscht werden und die `UndoLogsListSize` CloudWatch Metrik muss auf Null fallen, bevor ein Upgrade von einer Version vor 1.2.0.0 gestartet werden kann. Wenn beim Versuch, ein Update zu starten, zu viele Undo-Protokolldatensätze (200.000 oder mehr) vorhanden sind, kann es beim Upgrade-Versuch zu einer Zeitüberschreitung kommen, während auf den Abschluss des Löschvorgangs der Undo-Protokolle gewartet wird.

  Sie können die Löschgeschwindigkeit beschleunigen, indem Sie die Writer-Instance des Clusters aktualisieren, in der das Löschen stattfindet. Wenn Sie dies tun, bevor Sie versuchen, das Upgrade durchzuführen, kann die Anzahl der Undo-Logs vor dem Start

reduziert werden. Wenn Sie die Größe des Writers auf einen 24XL-Instance-Typ erhöhen, kann sich Ihre Löschrate auf mehr als eine Million Datensätze pro Stunde erhöhen.

Wenn die `UndoLogsListSize` CloudWatch Metrik extrem umfangreich ist, kann Ihnen die Eröffnung eines Support-Falls dabei helfen, zusätzliche Strategien zu finden, um sie zu reduzieren.

- Schließlich wurde mit Version 1.2.0.0 eine bahnbrechende Änderung eingeführt, die sich auf früheren Code auswirkt, der das Bolt-Protokoll mit IAM-Authentifizierung verwendete. Ab Version 1.2.0.0 benötigt Bolt einen Ressourcenpfad für die IAM-Signatur. In Java könnte das Festlegen des Ressourcenpfads so aussehen: `request.setResourcePath("/openCypher"));`. In anderen Sprachen kann `/openCypher` an den Endpunkt-URI angehängt werden. Beispiele finden Sie unter Verwenden des Bolt-Protokolls.

In diesem Engine-Version behobene Fehler

- Es wurde ein Fehler behoben, bei dem in einigen Fällen eine fehlgeschlagene Transaktion nicht korrekt abgeschlossen wurde.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.2.1.0.R7 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprache kompatibel ist:

- Die älteste unterstützte Version von Gremlin: `3.6.2`
- Die neueste unterstützte Version von Gremlin: `3.6.2`
- openCypher-Version: `Neptune-9.0.20190305-1.0`
- SPARQL-Version: `1.1`

Upgrade auf diesen Release

Amazon Neptune 1.2.1.0.R7 ist jetzt allgemein verfügbar.

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe

der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.1.0 \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.1.0 ^
    --apply-immediately
```

Updates werden auf alle Instances in einem DB-Cluster gleichzeitig angewendet. Ein Update erfordert einen Datenbankneustart auf diesen Instances. Daher kommt es zu einer Ausfallzeit von 20–30 Sekunden bis zu mehreren Minuten. Anschließend können Sie die Nutzung Ihres DB-Clusters fortsetzen.

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
> 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.
> ```
>
> Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine-Version 1.2.1.0.R6 (12.09.2023)

Ab dem 12.09.2023 wird die Engine-Version 1.2.1.0.R6 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

> **Note**
>
> Bei einem Upgrade von einer Engine-Version vor 1.2.0.0:
>
> - Mit der Engine-Version 1.2.0.0 wurde ein neues Format für benutzerdefinierte Parametergruppen und benutzerdefinierte Cluster-Parametergruppen eingeführt. Wenn Sie also von einer Engine-Version vor 1.2.0.0 auf Engine-Version 1.2.0.0 oder höher aktualisieren, müssen Sie alle vorhandenen benutzerdefinierten Parametergruppen und benutzerdefinierten Cluster-Parametergruppen mithilfe der Parametergruppenfamilie `neptune1.2` neu erstellen. In früheren Versionen wurde die Parametergruppenfamilie `neptune1` verwendet und diese Parametergruppen funktionieren nicht mit Version 1.2.0.0 und höher. Weitere Informationen finden Sie unter Amazon-Neptune-Parametergruppen.
> - Mit der Engine-Version 1.2.0.0 wurde auch ein neues Format für Undo-Protokolle eingeführt. Daher müssen alle Undo-Logs, die von einer früheren Engine-Version erstellt wurden, gelöscht werden und die `UndoLogsListSize` CloudWatch Metrik muss auf Null fallen, bevor ein Upgrade von einer Version vor 1.2.0.0 gestartet werden kann. Wenn beim Versuch, ein Update zu starten, zu viele Undo-Protokolldatensätze (200.000 oder mehr) vorhanden sind, kann es beim Upgrade-Versuch zu einer Zeitüberschreitung kommen, während auf den Abschluss des Löschvorgangs der Undo-Protokolle gewartet wird.
>
>   Sie können die Löschgeschwindigkeit beschleunigen, indem Sie die Writer-Instance des Clusters aktualisieren, in der das Löschen stattfindet. Wenn Sie dies tun, bevor Sie versuchen, das Upgrade durchzuführen, kann die Anzahl der Undo-Logs vor dem Start reduziert werden. Wenn Sie die Größe des Writers auf einen 24XL-Instance-Typ erhöhen, kann sich Ihre Löschrate auf mehr als eine Million Datensätze pro Stunde erhöhen.
>
>   Wenn die `UndoLogsListSize` CloudWatch Metrik extrem umfangreich ist, kann Ihnen die Eröffnung eines Support-Falls dabei helfen, zusätzliche Strategien zu finden, um sie zu reduzieren.
> - Schließlich wurde mit Version 1.2.0.0 eine bahnbrechende Änderung eingeführt, die sich auf früheren Code auswirkt, der das Bolt-Protokoll mit IAM-Authentifizierung verwendete. Ab Version 1.2.0.0 benötigt Bolt einen Ressourcenpfad für die IAM-Signatur. In Java könnte das Festlegen des Ressourcenpfads so aussehen:

`request.setResourcePath("/openCypher"));`. In anderen Sprachen kann `/openCypher` an den Endpunkt-URI angehängt werden. Beispiele finden Sie unter Verwenden des Bolt-Protokolls.

Neue Features in dieser Engine-Version

- Die Neptune-Daten-API wurde veröffentlicht.

  Die Amazon Neptune-Daten-API bietet SDK-Unterstützung für das Laden von Daten, das Ausführen von Abfragen, das Abrufen von Informationen über die Daten und das Ausführen von Machine-Learning-Operationen. Sie unterstützt die Abfragesprachen Gremlin und openCypher in Neptune und ist in allen SDK-Sprachen verfügbar. Sie signiert automatisch API-Anfragen und vereinfacht die Integration von Neptune in Anwendungen erheblich.

In diesem Engine-Version behobene Fehler

- Es wurde ein Fehler behoben, der bei hoher Auslastung zu CPU-Spitzen führen konnte, wenn Slow Query-Protokolle aktiviert waren.
- Es wurde ein Gremlin-Fehler behoben, durch den beim Hinzufügen einer Edge und ihrer Eigenschaften gefolgt von `inV()` oder `outV()` ein `InternalFailureException` ausgelöst wurde.
- Es wurden mehrere Probleme mit der IAM-Rollenverkettung behoben, die in einigen Fällen zu einer verminderten Massenladerleistung führten.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.2.1.0.R6 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprache kompatibel ist:

- Die älteste unterstützte Version von Gremlin: `3.6.2`
- Die neueste unterstützte Version von Gremlin: `3.6.2`
- openCypher-Version: `Neptune-9.0.20190305-1.0`
- SPARQL-Version: `1.1`

Upgrade auf diesen Release

Amazon Neptune 1.2.1.0.R6 ist jetzt allgemein verfügbar.

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.1.0 \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.1.0 ^
    --apply-immediately
```

Updates werden auf alle Instances in einem DB-Cluster gleichzeitig angewendet. Ein Update erfordert einen Datenbankneustart auf diesen Instances. Daher kommt es zu einer Ausfallzeit von 20–30 Sekunden bis zu mehreren Minuten. Anschließend können Sie die Nutzung Ihres DB-Clusters fortsetzen.

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue

Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
> 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.
> ```
>
> Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine-Version 1.2.1.0.R5 (2023-09-02)

Ab dem 02.09.2023 wird die Engine-Version 1.2.1.0.R5 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

> **Note**
>
> Bei einem Upgrade von einer Engine-Version vor 1.2.0.0:
>
> - Mit der Engine-Version 1.2.0.0 wurde ein neues Format für benutzerdefinierte Parametergruppen und benutzerdefinierte Cluster-Parametergruppen eingeführt. Wenn Sie also von einer Engine-Version vor 1.2.0.0 auf Engine-Version 1.2.0.0 oder höher aktualisieren, müssen Sie alle vorhandenen benutzerdefinierten Parametergruppen und benutzerdefinierten Cluster-Parametergruppen mithilfe der Parametergruppenfamilie `neptune1.2` neu erstellen. In früheren Versionen wurde die Parametergruppenfamilie `neptune1` verwendet und diese Parametergruppen funktionieren nicht mit Version 1.2.0.0 und höher. Weitere Informationen finden Sie unter Amazon-Neptune-Parametergruppen.
> - Mit der Engine-Version 1.2.0.0 wurde auch ein neues Format für Undo-Protokolle eingeführt. Daher müssen alle Undo-Logs, die von einer früheren Engine-Version erstellt wurden, gelöscht werden und die `UndoLogsListSize` CloudWatch Metrik muss auf Null fallen, bevor ein Upgrade von einer Version vor 1.2.0.0 gestartet werden kann. Wenn beim Versuch, ein Update zu starten, zu viele Undo-Protokolldatensätze (200.000 oder mehr) vorhanden sind, kann es beim Upgrade-Versuch zu einer Zeitüberschreitung kommen, während auf den Abschluss des Löschvorgangs der Undo-Protokolle gewartet wird.
>
>   Sie können die Löschgeschwindigkeit beschleunigen, indem Sie die Writer-Instance des Clusters aktualisieren, in der das Löschen stattfindet. Wenn Sie dies tun, bevor Sie versuchen, das Upgrade durchzuführen, kann die Anzahl der Undo-Logs vor dem Start reduziert werden. Wenn Sie die Größe des Writers auf einen 24XL-Instance-Typ erhöhen, kann sich Ihre Löschrate auf mehr als eine Million Datensätze pro Stunde erhöhen.
>
>   Wenn die `UndoLogsListSize` CloudWatch Metrik extrem umfangreich ist, kann Ihnen die Eröffnung eines Support-Falls dabei helfen, zusätzliche Strategien zu finden, um sie zu reduzieren.
> - Schließlich wurde mit Version 1.2.0.0 eine bahnbrechende Änderung eingeführt, die sich auf früheren Code auswirkt, der das Bolt-Protokoll mit IAM-Authentifizierung verwendete. Ab Version 1.2.0.0 benötigt Bolt einen Ressourcenpfad für die IAM-Signatur. In Java könnte das Festlegen des Ressourcenpfads so aussehen:

`request.setResourcePath("/openCypher"));`. In anderen Sprachen kann `/openCypher` an den Endpunkt-URI angehängt werden. Beispiele finden Sie unter Verwenden des Bolt-Protokolls.

Neue Features in dieser Engine-Version

- Die Neptune-Daten-API wurde veröffentlicht.

  Die Amazon Neptune-Daten-API bietet SDK-Unterstützung für das Laden von Daten, das Ausführen von Abfragen, das Abrufen von Informationen über die Daten und das Ausführen von Machine-Learning-Operationen. Sie unterstützt die Abfragesprachen Gremlin und openCypher in Neptune und ist in allen SDK-Sprachen verfügbar. Sie signiert automatisch API-Anfragen und vereinfacht die Integration von Neptune in Anwendungen erheblich.

In diesem Engine-Version behobene Fehler

- Es wurde ein Gremlin-Fehler behoben, durch den beim Hinzufügen einer Edge und ihrer Eigenschaften gefolgt von `inV()` oder `outV()` ein `InternalFailureException` ausgelöst wurde.
- Es wurden mehrere Probleme mit der IAM-Rollenverkettung behoben, die in einigen Fällen zu einer verminderten Massenladerleistung führten.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.2.1.0.R5 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprache kompatibel ist:

- Die älteste unterstützte Version von Gremlin: `3.6.2`
- Die neueste unterstützte Version von Gremlin: `3.6.2`
- openCypher-Version: `Neptune-9.0.20190305-1.0`
- SPARQL-Version: `1.1`

Upgrade auf diesen Release

Amazon Neptune 1.2.1.0.R5 ist jetzt allgemein verfügbar.

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.1.0 \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.1.0 ^
    --apply-immediately
```

Updates werden auf alle Instances in einem DB-Cluster gleichzeitig angewendet. Ein Update erfordert einen Datenbankneustart auf diesen Instances. Daher kommt es zu einer Ausfallzeit von 20–30 Sekunden bis zu mehreren Minuten. Anschließend können Sie die Nutzung Ihres DB-Clusters fortsetzen.

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
> 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.
> ```
>
> Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine-Version 1.2.1.0.R4 (2023-08-10)

Ab dem 10.08.2023 wird die Engine-Version 1.2.1.0.R4 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

Important

Die in dieser Engine-Version eingeführten Änderungen können in einigen Fällen dazu führen, dass Sie eine verringerte Leistung beim Laden von Massenladevorgängen feststellen. Aus diesem Grund wurden die Upgrades dieser Version vorübergehend ausgesetzt, bis das Problem behoben ist.

Note

Bei einem Upgrade von einer Engine-Version vor 1.2.0.0:

- Mit der Engine-Version 1.2.0.0 wurde ein neues Format für benutzerdefinierte Parametergruppen und benutzerdefinierte Cluster-Parametergruppen eingeführt. Wenn Sie also von einer Engine-Version vor 1.2.0.0 auf Engine-Version 1.2.0.0 oder höher aktualisieren, müssen Sie alle vorhandenen benutzerdefinierten Parametergruppen und benutzerdefinierten Cluster-Parametergruppen mithilfe der Parametergruppenfamilie `neptune1.2` neu erstellen. In früheren Versionen wurde die Parametergruppenfamilie `neptune1` verwendet und diese Parametergruppen funktionieren nicht mit Version 1.2.0.0 und höher. Weitere Informationen finden Sie unter Amazon-Neptune-Parametergruppen.
- Mit der Engine-Version 1.2.0.0 wurde auch ein neues Format für Undo-Protokolle eingeführt. Daher müssen alle Undo-Logs, die von einer früheren Engine-Version erstellt wurden, gelöscht werden und die `UndoLogsListSize` CloudWatch Metrik muss auf Null fallen, bevor ein Upgrade von einer Version vor 1.2.0.0 gestartet werden kann. Wenn beim Versuch, ein Update zu starten, zu viele Undo-Protokolldatensätze (200.000 oder mehr) vorhanden sind, kann es beim Upgrade-Versuch zu einer Zeitüberschreitung kommen, während auf den Abschluss des Löschvorgangs der Undo-Protokolle gewartet wird.

  Sie können die Löschgeschwindigkeit beschleunigen, indem Sie die Writer-Instance des Clusters aktualisieren, in der das Löschen stattfindet. Wenn Sie dies tun, bevor Sie versuchen, das Upgrade durchzuführen, kann die Anzahl der Undo-Logs vor dem Start reduziert werden. Wenn Sie die Größe des Writers auf einen 24XL-Instance-Typ erhöhen, kann sich Ihre Löschrate auf mehr als eine Million Datensätze pro Stunde erhöhen.

Wenn die `UndoLogsListSize` CloudWatch Metrik extrem umfangreich ist, kann Ihnen die Eröffnung eines Support-Falls dabei helfen, zusätzliche Strategien zu finden, um sie zu reduzieren.

- Schließlich wurde mit Version 1.2.0.0 eine bahnbrechende Änderung eingeführt, die sich auf früheren Code auswirkt, der das Bolt-Protokoll mit IAM-Authentifizierung verwendete. Ab Version 1.2.0.0 benötigt Bolt einen Ressourcenpfad für die IAM-Signatur. In Java könnte das Festlegen des Ressourcenpfads so aussehen: `request.setResourcePath("/openCypher"));`. In anderen Sprachen kann `/openCypher` an den Endpunkt-URI angehängt werden. Beispiele finden Sie unter Verwenden des Bolt-Protokolls.

Verbesserungen in dieser Engine-Version

- GraphSON-1.0-Unterstützung für Gremlin hinzugefügt. Um GraphSON-1.0 zu verwenden, passieren Sie `Accept header` mit einem Wert von:

```
application/vnd.gremlin-v1.0+json;types=false
```

In diesem Engine-Version behobene Fehler

- Es wurde ein Gremlin-Fehler behoben, bei dem es zu einem Transaktionsleck kam, wenn der Endpunkt des Gremlin-Abfragestatus auf Abfragen mit Prädikaten in untergeordneten Durchläufen für Schritte überprüft wurde, die nicht nativ verarbeitet wurden.
- Es wurde ein openCypher-Fehler bei der Transaktionsverarbeitung von Bolt behoben.
- Es wurde ein Parallelitätsproblem auf der Speicherebene behoben, das zu einem Absturz führen konnte.
- Es wurde ein Fehler in Protokollen für langsame Abfragen behoben, um sicherzustellen, dass sie nicht aktiv sind, wenn sie deaktiviert sind.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.2.1.0.R4 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprache kompatibel ist:

- Die älteste unterstützte Version von Gremlin: `3.6.2`
- Die neueste unterstützte Version von Gremlin: `3.6.5`
- openCypher-Version: `Neptune-9.0.20190305-1.0`
- SPARQL-Version: `1.1`

Upgrade-Pfade zum Engine-Release 1.2.1.0.R4

Upgrade auf diesen Release

Amazon Neptune 1.2.1.0.R4 ist jetzt allgemein verfügbar.

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.1.0 \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.1.0 ^
    --apply-immediately
```

Updates werden auf alle Instances in einem DB-Cluster gleichzeitig angewendet. Ein Update erfordert einen Datenbankneustart auf diesen Instances. Daher kommt es zu einer Ausfallzeit von 20–30 Sekunden bis zu mehreren Minuten. Anschließend können Sie die Nutzung Ihres DB-Clusters fortsetzen.

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein

Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
> 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.
> ```

Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine-Version 1.2.1.0.R3 (13.06.2023)

Ab dem 13.06.2023 wird die Engine-Version 1.2.1.0.R3 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

Important

Die in dieser Engine-Version eingeführten Änderungen können in einigen Fällen dazu führen, dass Sie eine verringerte Leistung beim Laden von Massenladevorgängen feststellen. Aus diesem Grund wurden die Upgrades dieser Version vorübergehend ausgesetzt, bis das Problem behoben ist.

Note

Bei einem Upgrade von einer Engine-Version vor 1.2.0.0:

- Mit der Engine-Version 1.2.0.0 wurde ein neues Format für benutzerdefinierte Parametergruppen und benutzerdefinierte Cluster-Parametergruppen eingeführt. Wenn Sie also von einer Engine-Version vor 1.2.0.0 auf Engine-Version 1.2.0.0 oder höher aktualisieren, müssen Sie alle vorhandenen benutzerdefinierten Parametergruppen und benutzerdefinierten Cluster-Parametergruppen mithilfe der Parametergruppenfamilie `neptune1.2` neu erstellen. In früheren Versionen wurde die Parametergruppenfamilie `neptune1` verwendet und diese Parametergruppen funktionieren nicht mit Version 1.2.0.0 und höher. Weitere Informationen finden Sie unter Amazon-Neptune-Parametergruppen.
- Mit der Engine-Version 1.2.0.0 wurde auch ein neues Format für Undo-Protokolle eingeführt. Daher müssen alle Undo-Logs, die von einer früheren Engine-Version erstellt wurden, gelöscht werden und die `UndoLogsListSize` CloudWatch Metrik muss auf Null

fallen, bevor ein Upgrade von einer Version vor 1.2.0.0 gestartet werden kann. Wenn beim Versuch, ein Update zu starten, zu viele Undo-Protokolldatensätze (200.000 oder mehr) vorhanden sind, kann es beim Upgrade-Versuch zu einer Zeitüberschreitung kommen, während auf den Abschluss des Löschvorgangs der Undo-Protokolle gewartet wird.

Sie können die Löschgeschwindigkeit beschleunigen, indem Sie die Writer-Instance des Clusters aktualisieren, in der das Löschen stattfindet. Wenn Sie dies tun, bevor Sie versuchen, das Upgrade durchzuführen, kann die Anzahl der Undo-Logs vor dem Start reduziert werden. Wenn Sie die Größe des Writers auf einen 24XL-Instance-Typ erhöhen, kann sich Ihre Löschrate auf mehr als eine Million Datensätze pro Stunde erhöhen.

Wenn die `UndoLogsListSize` CloudWatch Metrik extrem umfangreich ist, kann Ihnen die Eröffnung eines Support-Falls dabei helfen, zusätzliche Strategien zu finden, um sie zu reduzieren.

- Schließlich wurde mit Version 1.2.0.0 eine bahnbrechende Änderung eingeführt, die sich auf früheren Code auswirkt, der das Bolt-Protokoll mit IAM-Authentifizierung verwendete. Ab Version 1.2.0.0 benötigt Bolt einen Ressourcenpfad für die IAM-Signatur. In Java könnte das Festlegen des Ressourcenpfads so aussehen: `request.setResourcePath("/openCypher"));`. In anderen Sprachen kann `/openCypher` an den Endpunkt-URI angehängt werden. Beispiele finden Sie unter Verwenden des Bolt-Protokolls.

Neue Features in dieser Engine-Version

- Unterstützung für kontoübergreifendes Massenladen mithilfe der IAM-Rollenverkettung wurde hinzugefügt.

Verbesserungen in dieser Engine-Version

- Der `fail()`-Schritt von Gremlin wurde verbessert, um die erzeugte Ausnahme von einer generischen Ausnahme zu unterscheiden `InternalFailureException` und sicherzustellen, dass alle vom Benutzer bereitgestellten Nachrichten an den Aufrufer zurückgesendet wurden.
- Die Optimierungen der Gremlin-Abfrage-Engine für `store`, `aggregate`, `cap`, `limit` und `hasLabel` wurden verbessert.
- Unterstützung für trignometrische Features von openCypher hinzugefügt:

  - `acos()`
  - `asin()`
  - `atan()`
  - `atan2()`
  - `cos()`
  - `cot()`
  - `degrees()`
  - `pi()`
  - `radians()`
  - `sin()`
  - `tan()`
- Unterstützung für mehrere Aggregationsfunktionen von openCypher hinzugefügt:
  - `percentileDisc()`
  - `stDev()`
- Unterstützung für die openCypher-`epochmillis()`-Funktion hinzugefügt, die ein `datetime` in `epochmillis` umwandelt. Zum Beispiel:

```
MATCH (n) RETURN epochMillis(n.someDateTime)
1698972364782
```

- Unterstützung für den openCypher-Operator modulo (`%`) hinzugefügt.
- Unterstützung für das openCypher Static Debug-Erklärungs-Tool hinzugefügt.
- Unterstützung für die openCypher-`randomUUID()`-Funktion hinzugefügt.
- Verbesserung der Leistung von openCypher:
  - Der Parser und der Abfrageplaner wurden verbessert.
  - Verbesserung der CPU-Auslastung in der DFE-Engine.
  - Die Leistung von Abfragen mit mehreren Aktualisierungsklauseln, die dieselben Variablen wiederverwenden, wurde verbessert. Beispiele sind:

```
MERGE (n {name: 'John'})
  or
MERGE (m {name: 'Jim'})
  or
```

```
MERGE (n)-[:knows {since: 2023}]#(m)
```

- Optimierte Abfragepläne für Multi-Hop-Abfragemuster wie:

```
MATCH (n)-->()-->()-->(m)
RETURN n m
```

- Die Leistung der Listen- und Map-Injection durch parametrisierte Abfragen wurde verbessert. Zum Beispiel:

```
UNWIND $idList as id MATCH (n {`~id`: id})
RETURN n.name
```

- Die Ausführung von Abfragen mit `WITH` wurde verbessert, indem es zu einer geeigneten Barriere gemacht wurde.
- Optimiert, um eine redundante Materialisierung von Werten in `Unfold` und Aggregationsfunktionen zu vermeiden.

- Die Leistung von SPARQL-Abfragen, die eine große Anzahl statischer Eingaben in der `VALUES`-Klausel enthalten, wie z. B.:

```
SELECT ?n WHERE { VALUES (?name) { ("John") ("Jim") ... many values ... } ?n a ?
n_type . ?n ?name . }
```

- Verbesserung der SPARQL CBD-Abfrageleistung.

In diesem Engine-Version behobene Fehler

- Es wurde ein Gremlin-Fehler behoben, bei dem lange Abfragen mit tiefer Verschachtelung zu einer hohen CPU-Auslastung und Abfrage-Timeouts während der Abfrageplanungsphase führten.
- Es wurde ein Gremlin-Fehler behoben, durch den bei Verwendung von `mergeV` oder `mergeE` ein ungültiges `NullPointerException` ausgegeben werden konnte.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.2.1.0.R3 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprache kompatibel ist:

- Die älteste unterstützte Version von Gremlin: `3.6.2`

- Die neueste unterstützte Version von Gremlin: `3.6.2`
- openCypher-Version: `Neptune-9.0.20190305-1.0`
- SPARQL-Version: `1.1`

Upgrade-Pfade zum Engine-Release 1.2.1.0.R3

Upgrade auf diesen Release

Amazon Neptune 1.2.1.0.R3 ist jetzt allgemein verfügbar.

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.1.0 \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.1.0 ^
    --apply-immediately
```

Updates werden auf alle Instances in einem DB-Cluster gleichzeitig angewendet. Ein Update erfordert einen Datenbankneustart auf diesen Instances. Daher kommt es zu einer Ausfallzeit von 20–30 Sekunden bis zu mehreren Minuten. Anschließend können Sie die Nutzung Ihres DB-Clusters fortsetzen.

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein

Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
> 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.
> ```

Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine-Version 1.2.1.0.R2 (02.05.2023)

Ab dem 02.05.2023 wird die Engine-Version 1.2.1.0.R2 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

Note

Bei einem Upgrade von einer Engine-Version vor 1.2.0.0:

- Mit der Engine-Version 1.2.0.0 wurde ein neues Format für benutzerdefinierte Parametergruppen und benutzerdefinierte Cluster-Parametergruppen eingeführt. Wenn Sie also von einer Engine-Version vor 1.2.0.0 auf Engine-Version 1.2.0.0 oder höher aktualisieren, müssen Sie alle vorhandenen benutzerdefinierten Parametergruppen und benutzerdefinierten Cluster-Parametergruppen mithilfe der Parametergruppenfamilie `neptune1.2` neu erstellen. In früheren Versionen wurde die Parametergruppenfamilie `neptune1` verwendet und diese Parametergruppen funktionieren nicht mit Version 1.2.0.0 und höher. Weitere Informationen finden Sie unter Amazon-Neptune-Parametergruppen.
- Mit der Engine-Version 1.2.0.0 wurde auch ein neues Format für Undo-Protokolle eingeführt. Daher müssen alle Undo-Logs, die von einer früheren Engine-Version erstellt wurden, gelöscht werden und die `UndoLogsListSize` CloudWatch Metrik muss auf Null fallen, bevor ein Upgrade von einer Version vor 1.2.0.0 gestartet werden kann. Wenn beim Versuch, ein Update zu starten, zu viele Undo-Protokolldatensätze (200.000 oder mehr) vorhanden sind, kann es beim Upgrade-Versuch zu einer Zeitüberschreitung kommen, während auf den Abschluss des Löschvorgangs der Undo-Protokolle gewartet wird.

  Sie können die Löschgeschwindigkeit beschleunigen, indem Sie die Writer-Instance des Clusters aktualisieren, in der das Löschen stattfindet. Wenn Sie dies tun, bevor Sie versuchen, das Upgrade durchzuführen, kann die Anzahl der Undo-Logs vor dem Start

reduziert werden. Wenn Sie die Größe des Writers auf einen 24XL-Instance-Typ erhöhen, kann sich Ihre Löschrate auf mehr als eine Million Datensätze pro Stunde erhöhen.

Wenn die `UndoLogsListSize` CloudWatch Metrik extrem umfangreich ist, kann Ihnen die Eröffnung eines Support-Falls dabei helfen, zusätzliche Strategien zu finden, um sie zu reduzieren.

- Schließlich wurde mit Version 1.2.0.0 eine bahnbrechende Änderung eingeführt, die sich auf früheren Code auswirkt, der das Bolt-Protokoll mit IAM-Authentifizierung verwendete. Ab Version 1.2.0.0 benötigt Bolt einen Ressourcenpfad für die IAM-Signatur. In Java könnte das Festlegen des Ressourcenpfads so aussehen: `request.setResourcePath("/openCypher"));`. In anderen Sprachen kann `/openCypher` an den Endpunkt-URI angehängt werden. Beispiele finden Sie unter Verwenden des Bolt-Protokolls.

Verbesserungen in dieser Engine-Version

- Allen Neptune ML wurde ein `enableInterContainerTrafficEncryption` Parameter hinzugefügt APIs, mit dem Sie die Verschlüsselung des Datenverkehrs zwischen Containern in Trainings- oder Hyperparameter-Tuning-Jobs aktivieren und deaktivieren können.
- Unterstützung für mehrere Labels für Gremlin `mergeV()` und `mergeE()` hinzugefügt.

In diesem Engine-Version behobene Fehler

- Es wurde ein openCypher-Fehler behoben, durch den Aktualisierungs- und Rückgabeanfragen `orderBy`, `limit` oder `skip` nicht richtig behandelt haben.
- Es wurde ein openCypher-Fehler behoben, durch den Parameter, die in einer Anfrage enthalten waren, durch Parameter in einer anderen gleichzeitigen Anfrage überschrieben werden konnten.
- Es wurde ein openCypher-Fehler behoben, bei dem langsame Abfrageprotokolle nicht die korrekten Abfragezeiten enthielten.
- Es wurde ein Gremlin-Fehler behoben, durch den ein Transaktionsleck auftreten konnte, wenn eine Abfrage mit `GroupCountStep` als Zeichenfolge gesendet wurde.
- Es wurde ein Gremlin-Fehler behoben, bei dem WebSocket Abfragen fehlschlugen, wenn langsame Abfrageprotokolle aktiviert waren.

- Es wurde ein Gremlin-Fehler behoben, bei dem Storage-Counter-Debug-Logs in den langsamen Abfrageprotokollen für Anfragen fehlten. WebSocket
- Es wurden mehrere Gremlin-Fehler mit `mergeV()` und `mergeE()` behoben.
- Es wurde ein SPARQL-Fehler behoben, bei dem die Kosten für benannte Graph-Abfragen falsch geschätzt wurden, was zu suboptimalen Abfrageplänen und Fehlern führte. out-of-memory
- Es wurde ein Fehler behoben, der sich auf die Autorisierung von Gremlin- und openCypher-Abfragen auf einem IAM-fähigen Cluster auswirkte.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.2.1.0.R2 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprache kompatibel ist:

- Die älteste unterstützte Version von Gremlin: `3.6.2`
- Die neueste unterstützte Version von Gremlin: `3.6.2`
- openCypher-Version: `Neptune-9.0.20190305-1.0`
- SPARQL-Version: `1.1`

Upgrade-Pfade zum Engine-Release 1.2.1.0.R2

Upgrade auf diesen Release

Amazon Neptune 1.2.1.0.R2 ist jetzt allgemein verfügbar.

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.1.0 \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.1.0 ^
    --apply-immediately
```

Updates werden auf alle Instances in einem DB-Cluster gleichzeitig angewendet. Ein Update erfordert einen Datenbankneustart auf diesen Instances. Daher kommt es zu einer Ausfallzeit von 20–30 Sekunden bis zu mehreren Minuten. Anschließend können Sie die Nutzung Ihres DB-Clusters fortsetzen.

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune

einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
> 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.
> ```
>
> Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine-Version 1.2.0.2 (20.11.2022)

Ab dem 20.11.2022 wird die Engine-Version 1.2.0.2 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

> **Note**
>
> Bei einem Upgrade von einer Engine-Version vor 1.2.0.0:
>
> - Mit der Engine-Version 1.2.0.0 wurde ein neues Format für benutzerdefinierte Parametergruppen und benutzerdefinierte Cluster-Parametergruppen eingeführt. Wenn Sie also von einer Engine-Version vor 1.2.0.0 auf Engine-Version 1.2.0.0 oder höher aktualisieren, müssen Sie alle vorhandenen benutzerdefinierten Parametergruppen und benutzerdefinierten Cluster-Parametergruppen mithilfe der Parametergruppenfamilie `neptune1.2` neu erstellen. In früheren Versionen wurde die Parametergruppenfamilie

`neptune1` verwendet und diese Parametergruppen funktionieren nicht mit Version 1.2.0.0 und höher. Weitere Informationen finden Sie unter Amazon-Neptune-Parametergruppen.

- Mit der Engine-Version 1.2.0.0 wurde auch ein neues Format für Undo-Protokolle eingeführt. Daher müssen alle Undo-Logs, die von einer früheren Engine-Version erstellt wurden, gelöscht werden und die `UndoLogsListSize` CloudWatch Metrik muss auf Null fallen, bevor ein Upgrade von einer Version vor 1.2.0.0 gestartet werden kann. Wenn beim Versuch, ein Update zu starten, zu viele Undo-Protokolldatensätze (200.000 oder mehr) vorhanden sind, kann es beim Upgrade-Versuch zu einer Zeitüberschreitung kommen, während auf den Abschluss des Löschvorgangs der Undo-Protokolle gewartet wird.

  Sie können die Löschgeschwindigkeit beschleunigen, indem Sie die Writer-Instance des Clusters aktualisieren, in der das Löschen stattfindet. Wenn Sie dies tun, bevor Sie versuchen, das Upgrade durchzuführen, kann die Anzahl der Undo-Logs vor dem Start reduziert werden. Wenn Sie die Größe des Writers auf einen 24XL-Instance-Typ erhöhen, kann sich Ihre Löschrate auf mehr als eine Million Datensätze pro Stunde erhöhen.

  Wenn die `UndoLogsListSize` CloudWatch Metrik extrem umfangreich ist, kann Ihnen die Eröffnung eines Support-Falls dabei helfen, zusätzliche Strategien zu finden, um sie zu reduzieren.
- Schließlich wurde mit Version 1.2.0.0 eine bahnbrechende Änderung eingeführt, die sich auf früheren Code auswirkt, der das Bolt-Protokoll mit IAM-Authentifizierung verwendete. Ab Version 1.2.0.0 benötigt Bolt einen Ressourcenpfad für die IAM-Signatur. In Java könnte das Festlegen des Ressourcenpfads so aussehen: `request.setResourcePath("/openCypher"));`. In anderen Sprachen kann `/openCypher` an den Endpunkt-URI angehängt werden. Beispiele finden Sie unter Verwenden des Bolt-Protokolls.

Nachfolgende Patch-Veröffentlichungen für dieses Version

- Release: 1.2.0.2.R2 (15.12.2022)
- Release: 1.2.0.2.R3 (27.03.2023)
- Release: 1.2.0.2.R4 (08.05.2023)
- Release: 1.2.0.2.R5 (16.08.2023)
- Release: 1.2.0.2.R6 (12.09.2023)

Neue Features in dieser Engine-Version

- Einführung der induktiven Echtzeit-Inferenz für Gremlin in Neptune ML.
- Es wurde eine OpenCypher-Erweiterung eingeführt, die die Angabe benutzerdefinierter ID-Werte für Entitäten unterstützt, anstatt die UUIDs , die Neptune sonst generiert. Die Möglichkeit, benutzerdefinierte Zuweisungen vorzunehmen, IDs erleichtert die Migration von Neo4j zu Neptune.

  > **Warning**
  >
  > Diese Erweiterung der openCypher-Spezifikation ist nicht abwärtskompatibel, da `~id` jetzt als reservierter Eigenschaftsname gilt. Wenn Sie in Ihren Daten und Abfragen bereits `~id` als Eigenschaft verwenden, müssen Sie die Eigenschaft auf einen neuen `~id`-Eigenschaftsschlüssel migrieren, bevor Sie auf diesen Release aktualisieren.

- Es wurden mehrere neue `DESCRIBE` SPARQL-Modi zusammen mit Abfragehinweisen hinzugefügt, um sie zu konfigurieren.

Verbesserungen in dieser Engine-Version

- Die Leistung von openCypher wurde verbessert, insbesondere für VLP-Abfragen.
- Verbesserte DFE-Leistung für Gremlin-Abfragen mit Nicht-Terminal-Beschränkungen, wie z. B.:

```
g.withSideEffect('Neptune#useDFE',true).V().hasLabel('Student').limit(5).out('takesCourse')
```

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.2.0.2 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprache kompatibel ist:

- Die älteste unterstützte Version von Gremlin: `3.5.2`
- Die neueste unterstützte Version von Gremlin: `3.5.4`
- openCypher-Version: `Neptune-9.0.20190305-1.0`
- SPARQL-Version: `1.1`

Upgrade-Pfade zum Engine-Release 1.2.0.2

Upgrade auf diesen Release

Amazon Neptune 1.2.0.2 ist jetzt allgemein verfügbar.

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.2 \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.2 ^
    --apply-immediately
```

Updates werden auf alle Instances in einem DB-Cluster gleichzeitig angewendet. Ein Update erfordert einen Datenbankneustart auf diesen Instances. Daher kommt es zu einer Ausfallzeit von 20–30 Sekunden bis zu mehreren Minuten. Anschließend können Sie die Nutzung Ihres DB-Clusters fortsetzen.

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
> 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.
> ```
>
> Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine-Version 1.2.0.2.R6 (12.09.2023)

Ab dem 12.09.2023 wird die Engine-Version 1.2.0.2.R6 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

> **Note**
>
> Bei einem Upgrade von einer Engine-Version vor 1.2.0.0:
>
> - Mit der Engine-Version 1.2.0.0 wurde ein neues Format für benutzerdefinierte Parametergruppen und benutzerdefinierte Cluster-Parametergruppen eingeführt. Wenn Sie also von einer Engine-Version vor 1.2.0.0 auf Engine-Version 1.2.0.0 oder höher aktualisieren, müssen Sie alle vorhandenen benutzerdefinierten Parametergruppen und benutzerdefinierten Cluster-Parametergruppen mithilfe der Parametergruppenfamilie `neptune1.2` neu erstellen. In früheren Versionen wurde die Parametergruppenfamilie `neptune1` verwendet und diese Parametergruppen funktionieren nicht mit Version 1.2.0.0 und höher. Weitere Informationen finden Sie unter Amazon-Neptune-Parametergruppen.
> - Mit der Engine-Version 1.2.0.0 wurde auch ein neues Format für Undo-Protokolle eingeführt. Daher müssen alle Undo-Logs, die von einer früheren Engine-Version erstellt wurden, gelöscht werden und die `UndoLogsListSize` CloudWatch Metrik muss auf Null fallen, bevor ein Upgrade von einer Version vor 1.2.0.0 gestartet werden kann. Wenn beim Versuch, ein Update zu starten, zu viele Undo-Protokolldatensätze (200.000 oder mehr) vorhanden sind, kann es beim Upgrade-Versuch zu einer Zeitüberschreitung kommen, während auf den Abschluss des Löschvorgangs der Undo-Protokolle gewartet wird.
>
>   Sie können die Löschgeschwindigkeit beschleunigen, indem Sie die Writer-Instance des Clusters aktualisieren, in der das Löschen stattfindet. Wenn Sie dies tun, bevor Sie versuchen, das Upgrade durchzuführen, kann die Anzahl der Undo-Logs vor dem Start reduziert werden. Wenn Sie die Größe des Writers auf einen 24XL-Instance-Typ erhöhen, kann sich Ihre Löschrate auf mehr als eine Million Datensätze pro Stunde erhöhen.
>
>   Wenn die `UndoLogsListSize` CloudWatch Metrik extrem umfangreich ist, kann Ihnen die Eröffnung eines Support-Falls dabei helfen, zusätzliche Strategien zu finden, um sie zu reduzieren.

- Schließlich wurde mit Version 1.2.0.0 eine bahnbrechende Änderung eingeführt, die sich auf früheren Code auswirkt, der das Bolt-Protokoll mit IAM-Authentifizierung verwendete. Ab Version 1.2.0.0 benötigt Bolt einen Ressourcenpfad für die IAM-Signatur. In Java könnte das Festlegen des Ressourcenpfads so aussehen: `request.setResourcePath("/openCypher"));`. In anderen Sprachen kann `/openCypher` an den Endpunkt-URI angehängt werden. Beispiele finden Sie unter Verwenden des Bolt-Protokolls.

In diesem Engine-Version behobene Fehler

- Es wurde ein SPARQL-Fehler behoben, bei dem der `REGEX`-Operator nie erfolgreich war, wenn er für ein mit einer Sprache markiertes Literal aufgerufen wurde.
- Es wurde ein Problem behoben, durch das die Leistung beim Massenladen beeinträchtigt wurde.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.2.0.2.R6 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprache kompatibel ist:

- Die älteste unterstützte Version von Gremlin: `3.5.2`
- Die neueste unterstützte Version von Gremlin: `3.5.5`
- openCypher-Version: `Neptune-9.0.20190305-1.0`
- SPARQL-Version: `1.1`

Upgrade-Pfade zum Engine-Release 1.2.0.2.R6

Ihr Cluster wird während des nächsten Wartungsfensters automatisch auf diese Patch-Version aktualisiert, wenn Sie die Modulversion `1.2.0.2` ausführen.

Upgrade auf diesen Release

Amazon Neptune 1.2.0.2.R6 ist jetzt allgemein verfügbar.

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe

der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.2 \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.2 ^
    --apply-immediately
```

Updates werden auf alle Instances in einem DB-Cluster gleichzeitig angewendet. Ein Update erfordert einen Datenbankneustart auf diesen Instances. Daher kommt es zu einer Ausfallzeit von 20–30 Sekunden bis zu mehreren Minuten. Anschließend können Sie die Nutzung Ihres DB-Clusters fortsetzen.

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
> 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.
> ```
>
> Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine-Version 1.2.0.2.R5 (2023-08-16)

Ab dem 16.08.2023 wird die Engine-Version 1.2.0.2.R5 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

Important

Die in dieser Engine-Version eingeführten Änderungen können in einigen Fällen dazu führen, dass Sie eine verringerte Leistung beim Laden von Massenladevorgängen feststellen. Aus diesem Grund wurden die Upgrades dieser Version vorübergehend ausgesetzt, bis das Problem behoben ist.

Note

Bei einem Upgrade von einer Engine-Version vor 1.2.0.0:

- Mit der Engine-Version 1.2.0.0 wurde ein neues Format für benutzerdefinierte Parametergruppen und benutzerdefinierte Cluster-Parametergruppen eingeführt. Wenn Sie also von einer Engine-Version vor 1.2.0.0 auf Engine-Version 1.2.0.0 oder höher aktualisieren, müssen Sie alle vorhandenen benutzerdefinierten Parametergruppen und benutzerdefinierten Cluster-Parametergruppen mithilfe der Parametergruppenfamilie `neptune1.2` neu erstellen. In früheren Versionen wurde die Parametergruppenfamilie `neptune1` verwendet und diese Parametergruppen funktionieren nicht mit Version 1.2.0.0 und höher. Weitere Informationen finden Sie unter Amazon-Neptune-Parametergruppen.
- Mit der Engine-Version 1.2.0.0 wurde auch ein neues Format für Undo-Protokolle eingeführt. Daher müssen alle Undo-Logs, die von einer früheren Engine-Version erstellt wurden, gelöscht werden und die `UndoLogsListSize` CloudWatch Metrik muss auf Null fallen, bevor ein Upgrade von einer Version vor 1.2.0.0 gestartet werden kann. Wenn beim Versuch, ein Update zu starten, zu viele Undo-Protokolldatensätze (200.000 oder mehr) vorhanden sind, kann es beim Upgrade-Versuch zu einer Zeitüberschreitung kommen, während auf den Abschluss des Löschvorgangs der Undo-Protokolle gewartet wird.

  Sie können die Löschgeschwindigkeit beschleunigen, indem Sie die Writer-Instance des Clusters aktualisieren, in der das Löschen stattfindet. Wenn Sie dies tun, bevor Sie versuchen, das Upgrade durchzuführen, kann die Anzahl der Undo-Logs vor dem Start reduziert werden. Wenn Sie die Größe des Writers auf einen 24XL-Instance-Typ erhöhen, kann sich Ihre Löschrate auf mehr als eine Million Datensätze pro Stunde erhöhen.

Wenn die `UndoLogsListSize` CloudWatch Metrik extrem umfangreich ist, kann Ihnen die Eröffnung eines Support-Falls dabei helfen, zusätzliche Strategien zu finden, um sie zu reduzieren.

- Schließlich wurde mit Version 1.2.0.0 eine bahnbrechende Änderung eingeführt, die sich auf früheren Code auswirkt, der das Bolt-Protokoll mit IAM-Authentifizierung verwendete. Ab Version 1.2.0.0 benötigt Bolt einen Ressourcenpfad für die IAM-Signatur. In Java könnte das Festlegen des Ressourcenpfads so aussehen: `request.setResourcePath("/openCypher"));`. In anderen Sprachen kann `/openCypher` an den Endpunkt-URI angehängt werden. Beispiele finden Sie unter Verwenden des Bolt-Protokolls.

In diesem Engine-Version behobene Fehler

- Es wurde ein Gremlin-Fehler behoben, bei dem Zeichenkettenausgaben von `order()` nicht richtig sortiert wurden, wenn einige von ihnen ein Leerzeichen enthielten.
- Es wurde ein Gremlin-Fehler behoben, bei dem es zu einem Transaktionsleck kam, wenn der Endpunkt des Gremlin-Abfragestatus auf Abfragen mit Prädikaten in untergeordneten Durchläufen für Schritte überprüft wurde, die nicht nativ verarbeitet wurden.
- Es wurde ein openCypher-Fehler bei der Transaktionsverarbeitung von Bolt behoben.
- Es wurde ein Parallelitätsproblem auf der Speicherebene behoben, das zu einem Absturz führen konnte.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.2.0.2.R5 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprache kompatibel ist:

- Die älteste unterstützte Version von Gremlin: `3.5.2`
- Die neueste unterstützte Version von Gremlin: `3.5.5`
- openCypher-Version: `Neptune-9.0.20190305-1.0`
- SPARQL-Version: `1.1`

Upgrade-Pfade zum Engine-Release 1.2.0.2.R5

Ihr Cluster wird während des nächsten Wartungsfensters automatisch auf diese Patch-Version aktualisiert, wenn Sie die Modulversion `1.2.0.2` ausführen.

Upgrade auf diesen Release

Amazon Neptune 1.2.0.2.R5 ist jetzt allgemein verfügbar.

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.2 \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.2 ^
    --apply-immediately
```

Updates werden auf alle Instances in einem DB-Cluster gleichzeitig angewendet. Ein Update erfordert einen Datenbankneustart auf diesen Instances. Daher kommt es zu einer Ausfallzeit von 20–30 Sekunden bis zu mehreren Minuten. Anschließend können Sie die Nutzung Ihres DB-Clusters fortsetzen.

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
> 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.
> ```
>
> Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine-Version 1.2.0.2.R4 (2023-05-08)

Ab dem 08.05.2023 wird die Engine-Version 1.2.0.2.R4 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

> **Note**
>
> Bei einem Upgrade von einer Engine-Version vor 1.2.0.0:
>
> - Mit der Engine-Version 1.2.0.0 wurde ein neues Format für benutzerdefinierte Parametergruppen und benutzerdefinierte Cluster-Parametergruppen eingeführt. Wenn Sie also von einer Engine-Version vor 1.2.0.0 auf Engine-Version 1.2.0.0 oder höher aktualisieren, müssen Sie alle vorhandenen benutzerdefinierten Parametergruppen und benutzerdefinierten Cluster-Parametergruppen mithilfe der Parametergruppenfamilie `neptune1.2` neu erstellen. In früheren Versionen wurde die Parametergruppenfamilie `neptune1` verwendet und diese Parametergruppen funktionieren nicht mit Version 1.2.0.0 und höher. Weitere Informationen finden Sie unter Amazon-Neptune-Parametergruppen.
> - Mit der Engine-Version 1.2.0.0 wurde auch ein neues Format für Undo-Protokolle eingeführt. Daher müssen alle Undo-Logs, die von einer früheren Engine-Version erstellt wurden, gelöscht werden und die `UndoLogsListSize` CloudWatch Metrik muss auf Null fallen, bevor ein Upgrade von einer Version vor 1.2.0.0 gestartet werden kann. Wenn beim Versuch, ein Update zu starten, zu viele Undo-Protokolldatensätze (200.000 oder mehr) vorhanden sind, kann es beim Upgrade-Versuch zu einer Zeitüberschreitung kommen, während auf den Abschluss des Löschvorgangs der Undo-Protokolle gewartet wird.
>
>   Sie können die Löschgeschwindigkeit beschleunigen, indem Sie die Writer-Instance des Clusters aktualisieren, in der das Löschen stattfindet. Wenn Sie dies tun, bevor Sie versuchen, das Upgrade durchzuführen, kann die Anzahl der Undo-Logs vor dem Start reduziert werden. Wenn Sie die Größe des Writers auf einen 24XL-Instance-Typ erhöhen, kann sich Ihre Löschrate auf mehr als eine Million Datensätze pro Stunde erhöhen.
>
>   Wenn die `UndoLogsListSize` CloudWatch Metrik extrem umfangreich ist, kann Ihnen die Eröffnung eines Support-Falls dabei helfen, zusätzliche Strategien zu finden, um sie zu reduzieren.

- Schließlich wurde mit Version 1.2.0.0 eine bahnbrechende Änderung eingeführt, die sich auf früheren Code auswirkt, der das Bolt-Protokoll mit IAM-Authentifizierung verwendete. Ab Version 1.2.0.0 benötigt Bolt einen Ressourcenpfad für die IAM-Signatur. In Java könnte das Festlegen des Ressourcenpfads so aussehen: `request.setResourcePath("/openCypher"));`. In anderen Sprachen kann `/openCypher` an den Endpunkt-URI angehängt werden. Beispiele finden Sie unter Verwenden des Bolt-Protokolls.

In diesem Engine-Version behobene Fehler

- Es wurde ein SPARQL-Fehler behoben, bei dem eine große Anzahl von Werten, die durch die `VALUES`-Klausel eingefügt wurden, zu Leistungseinbußen führen konnte.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.2.0.2.R4 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprache kompatibel ist:

- Die älteste unterstützte Version von Gremlin: `3.5.2`
- Die neueste unterstützte Version von Gremlin: `3.5.6`
- openCypher-Version: `Neptune-9.0.20190305-1.0`
- SPARQL-Version: `1.1`

Upgrade-Pfade zum Engine-Release 1.2.0.2.R4

Ihr Cluster wird während des nächsten Wartungsfensters automatisch auf diese Patch-Version aktualisiert, wenn Sie die Modulversion `1.2.0.2` ausführen.

Upgrade auf diesen Release

Amazon Neptune 1.2.0.2.R4 ist jetzt allgemein verfügbar.

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.2 \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.2 ^
    --apply-immediately
```

Updates werden auf alle Instances in einem DB-Cluster gleichzeitig angewendet. Ein Update erfordert einen Datenbankneustart auf diesen Instances. Daher kommt es zu einer Ausfallzeit von 20–30 Sekunden bis zu mehreren Minuten. Anschließend können Sie die Nutzung Ihres DB-Clusters fortsetzen.

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
>   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.
> ```
>
> Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine-Version 1.2.0.2.R3 (27.03.2023)

Ab dem 27.03.2023 wird die Engine-Version 1.2.0.2.R3 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

> **Note**
>
> Bei einem Upgrade von einer Engine-Version vor 1.2.0.0:

- Mit der Engine-Version 1.2.0.0 wurde ein neues Format für benutzerdefinierte Parametergruppen und benutzerdefinierte Cluster-Parametergruppen eingeführt. Wenn Sie also von einer Engine-Version vor 1.2.0.0 auf Engine-Version 1.2.0.0 oder höher aktualisieren, müssen Sie alle vorhandenen benutzerdefinierten Parametergruppen und benutzerdefinierten Cluster-Parametergruppen mithilfe der Parametergruppenfamilie `neptune1.2` neu erstellen. In früheren Versionen wurde die Parametergruppenfamilie `neptune1` verwendet und diese Parametergruppen funktionieren nicht mit Version 1.2.0.0 und höher. Weitere Informationen finden Sie unter Amazon-Neptune-Parametergruppen.
- Mit der Engine-Version 1.2.0.0 wurde auch ein neues Format für Undo-Protokolle eingeführt. Daher müssen alle Undo-Logs, die von einer früheren Engine-Version erstellt wurden, gelöscht werden und die `UndoLogsListSize` CloudWatch Metrik muss auf Null fallen, bevor ein Upgrade von einer Version vor 1.2.0.0 gestartet werden kann. Wenn beim Versuch, ein Update zu starten, zu viele Undo-Protokolldatensätze (200.000 oder mehr) vorhanden sind, kann es beim Upgrade-Versuch zu einer Zeitüberschreitung kommen, während auf den Abschluss des Löschvorgangs der Undo-Protokolle gewartet wird.

  Sie können die Löschgeschwindigkeit beschleunigen, indem Sie die Writer-Instance des Clusters aktualisieren, in der das Löschen stattfindet. Wenn Sie dies tun, bevor Sie versuchen, das Upgrade durchzuführen, kann die Anzahl der Undo-Logs vor dem Start reduziert werden. Wenn Sie die Größe des Writers auf einen 24XL-Instance-Typ erhöhen, kann sich Ihre Löschrate auf mehr als eine Million Datensätze pro Stunde erhöhen.

  Wenn die `UndoLogsListSize` CloudWatch Metrik extrem umfangreich ist, kann Ihnen die Eröffnung eines Support-Falls dabei helfen, zusätzliche Strategien zu finden, um sie zu reduzieren.
- Schließlich wurde mit Version 1.2.0.0 eine bahnbrechende Änderung eingeführt, die sich auf früheren Code auswirkt, der das Bolt-Protokoll mit IAM-Authentifizierung verwendete. Ab Version 1.2.0.0 benötigt Bolt einen Ressourcenpfad für die IAM-Signatur. In Java könnte das Festlegen des Ressourcenpfads so aussehen: `request.setResourcePath("/openCypher"));`. In anderen Sprachen kann `/openCypher` an den Endpunkt-URI angehängt werden. Beispiele finden Sie unter Verwenden des Bolt-Protokolls.

Verbesserungen in dieser Engine-Version

- Für serverlose DB-Cluster wurde die Einstellung für die Mindestkapazität auf 1,0 NCU und die niedrigste gültige Maximaleinstellung auf 2,5 geändert. NCUs Siehe Kapazitätsskalierung in einem Neptune-Serverless-DB-Cluster
- Allen Neptune ML wurde ein `enableInterContainerTrafficEncryption` Parameter hinzugefügt APIs, mit dem Sie die Verschlüsselung des Datenverkehrs zwischen Containern in Trainings- oder Hyperparameter-Tuning-Jobs aktivieren und deaktivieren können.

In diesem Engine-Version behobene Fehler

- Es wurde ein Gremlin-Fehler behoben, bei dem die Gremlin-Syntax nicht `option(Predicate)` als gültige Gremlin-Syntax erkannt wurde.
- Es wurde ein Gremlin-Fehler behoben, der dazu führte, dass Abfragen nicht richtig bereinigt wurden, wenn sie fehlschlugen, weil sie zu viele Schritte enthielten.
- Es wurde ein Problem mit der Gremlin-Korrektheit behoben, das DFE-Abfragen mit `limit` untergeordnetem Traversal von Nicht-Union-Schritten betraf, indem auf Tinkerpop zurückgegriffen wurde. Hier ist ein Beispiel für eine solche Abfrage:

```
g.withSideEffect('Neptune#useDFE', true).V().as("a").select("a").by(out().limit(1))
```

- Es wurde ein potenzielles Gremlin-Transaktionsleck behoben, wenn eine als Zeichenfolge eingereichte Abfrage `GroupCountStep` enthält.
- Es wurde ein openCypher-Fehler behoben, bei dem der Typ des Parameterwerts in einer Liste oder einer Liste von Maps nicht korrekt abgeleitet wurde.
- Es wurde ein openCypher-Fehler behoben, durch den Aktualisierungs- und Rückgabeanfragen `orderBy`, `limit` oder `skip` nicht richtig behandelt haben.
- Es wurde ein openCypher-Fehler behoben, durch den Parameter, die in einer Anfrage enthalten waren, durch Parameter in einer anderen gleichzeitigen Anfrage überschrieben werden konnten.
- Es wurde ein SPARQL-Fehler behoben, bei dem eine große Anzahl von Werten, die in eine `VALUES`-Klausel eingefügt wurden, zu Leistungseinbußen führen konnte.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.2.0.2.R3 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprache kompatibel ist:

- Die älteste unterstützte Version von Gremlin: `3.5.2`
- Die neueste unterstützte Version von Gremlin: `3.5.6`
- openCypher-Version: `Neptune-9.0.20190305-1.0`
- SPARQL-Version: `1.1`

Upgrade-Pfade zum Engine-Release 1.2.0.2.R3

Ihr Cluster wird während des nächsten Wartungsfensters automatisch auf diese Patch-Version aktualisiert, wenn Sie die Modulversion `1.2.0.2` ausführen.

Upgrade auf diesen Release

Amazon Neptune 1.2.0.2.R3 ist jetzt allgemein verfügbar.

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.2 \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.2 ^
    --apply-immediately
```

Updates werden auf alle Instances in einem DB-Cluster gleichzeitig angewendet. Ein Update erfordert einen Datenbankneustart auf diesen Instances. Daher kommt es zu einer Ausfallzeit von

20–30 Sekunden bis zu mehreren Minuten. Anschließend können Sie die Nutzung Ihres DB-Clusters fortsetzen.

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:

```
  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.
```

Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine-Version 1.2.0.2.R2 (15.12.2022)

Ab dem 15.12.2022 wird die Engine-Version 1.2.0.2.R2 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

Note

Bei einem Upgrade von einer Engine-Version vor 1.2.0.0:

- Mit der Engine-Version 1.2.0.0 wurde ein neues Format für benutzerdefinierte Parametergruppen und benutzerdefinierte Cluster-Parametergruppen eingeführt. Wenn Sie also von einer Engine-Version vor 1.2.0.0 auf Engine-Version 1.2.0.0 oder höher aktualisieren, müssen Sie alle vorhandenen benutzerdefinierten Parametergruppen und benutzerdefinierten Cluster-Parametergruppen mithilfe der Parametergruppenfamilie `neptune1.2` neu erstellen. In früheren Versionen wurde die Parametergruppenfamilie `neptune1` verwendet und diese Parametergruppen funktionieren nicht mit Version 1.2.0.0 und höher. Weitere Informationen finden Sie unter Amazon-Neptune-Parametergruppen.
- Mit der Engine-Version 1.2.0.0 wurde auch ein neues Format für Undo-Protokolle eingeführt. Daher müssen alle Undo-Logs, die von einer früheren Engine-Version erstellt wurden, gelöscht werden und die `UndoLogsListSize` CloudWatch Metrik muss auf Null fallen, bevor ein Upgrade von einer Version vor 1.2.0.0 gestartet werden kann. Wenn beim Versuch, ein Update zu starten, zu viele Undo-Protokolldatensätze (200.000 oder mehr)

vorhanden sind, kann es beim Upgrade-Versuch zu einer Zeitüberschreitung kommen, während auf den Abschluss des Löschvorgangs der Undo-Protokolle gewartet wird.

Sie können die Löschgeschwindigkeit beschleunigen, indem Sie die Writer-Instance des Clusters aktualisieren, in der das Löschen stattfindet. Wenn Sie dies tun, bevor Sie versuchen, das Upgrade durchzuführen, kann die Anzahl der Undo-Logs vor dem Start reduziert werden. Wenn Sie die Größe des Writers auf einen 24XL-Instance-Typ erhöhen, kann sich Ihre Löschrate auf mehr als eine Million Datensätze pro Stunde erhöhen.

Wenn die `UndoLogsListSize` CloudWatch Metrik extrem umfangreich ist, kann Ihnen die Eröffnung eines Support-Falls dabei helfen, zusätzliche Strategien zu finden, um sie zu reduzieren.

- Schließlich wurde mit Version 1.2.0.0 eine bahnbrechende Änderung eingeführt, die sich auf früheren Code auswirkt, der das Bolt-Protokoll mit IAM-Authentifizierung verwendete. Ab Version 1.2.0.0 benötigt Bolt einen Ressourcenpfad für die IAM-Signatur. In Java könnte das Festlegen des Ressourcenpfads so aussehen: `request.setResourcePath("/openCypher"));`. In anderen Sprachen kann `/openCypher` an den Endpunkt-URI angehängt werden. Beispiele finden Sie unter Verwenden des Bolt-Protokolls.

Verbesserungen in dieser Engine-Version

- Verbesserte Leistung von openCypher-Abfragen mit `MERGE` und `OPTIONAL MATCH`.
- Verbesserte Leistung von openCypher-Abfragen, die eine Liste `UNWIND` von Zuordnungen mit Literalwerten beinhalten.
- Verbesserte Leistung von openCypher-Abfragen, die einen `IN`-Filter für `id` haben. Zum Beispiel:

```
MATCH (n) WHERE id(n) IN ['1', '2', '3'] RETURN n
```

- Leistungsverbesserungen und Korrekturkorrekturen für verschiedene Gremlin-Operatoren, darunter `repeat`, `coalesce`, `store` und `aggregate`.

In diesem Engine-Version behobene Fehler

- Es wurde ein openCypher-Fehler behoben, bei dem Abfragen die Zeichenfolge, `"null"`, anstelle eines Nullwerts in Bolt und SPARQL-JSON zurückgaben.

- Es wurde ein Gremlin-Fehler behoben, der dazu führte, dass ein mit einem Schritt verknüpftes Label `UnionStep` nicht an das letzte Pfadelement seiner untergeordneten Traversalen weitergegeben wurde.
- Es wurde ein Gremlin-Fehler behoben, der dazu führte, dass `valueMap()` bei einem `by()`-Traversal in der DFE-Engine nicht optimiert wurde.
- Es wurde ein Gremlin-Fehler behoben, bei dem Leseabfragen, die als Teil einer längeren Gremlin-Transaktion ausgeführt wurden, die Zeilen nicht sperrten.
- Es wurde ein Fehler im Auditprotokoll behoben, der dazu führte, dass unnötige Informationen protokolliert wurden und bestimmte Felder in den Protokollen fehlten.
- Es wurde ein Auditprotokollfehler behoben, bei dem der IAM-ARN von HTTP-Anfragen an einen IAM-fähigen DB-Cluster nicht aufgezeichnet wurde.
- Es wurde ein Fehler im Lookup-Cache behoben, sodass der inkrementelle Speicher, der für Schreibvorgänge in den Cache verwendet wurde, begrenzt wurde.
- Es wurde ein Lookup-Cache-Fehler behoben, bei dem für den Lookup-Cache der Nur-Lese-Modus eingestellt wurde, wenn Schreibvorgänge fehlschlugen.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.2.0.2.R2 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprache kompatibel ist:

- Die älteste unterstützte Version von Gremlin: `3.5.2`
- Die neueste unterstützte Version von Gremlin: `3.5.4`
- openCypher-Version: `Neptune-9.0.20190305-1.0`
- SPARQL-Version: `1.1`

Upgrade-Pfade zum Engine-Release 1.2.0.2.R2

Ihr Cluster wird während des nächsten Wartungsfensters automatisch auf diese Patch-Version aktualisiert, wenn Sie die Modulversion `1.2.0.2` ausführen.

Upgrade auf diesen Release

Amazon Neptune 1.2.0.2.R2 ist jetzt allgemein verfügbar.

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.2 \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.2 ^
    --apply-immediately
```

Updates werden auf alle Instances in einem DB-Cluster gleichzeitig angewendet. Ein Update erfordert einen Datenbankneustart auf diesen Instances. Daher kommt es zu einer Ausfallzeit von 20–30 Sekunden bis zu mehreren Minuten. Anschließend können Sie die Nutzung Ihres DB-Clusters fortsetzen.

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
> 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.
> ```
>
> Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine-Version 1.2.0.1 (26.10.2022)

Ab dem 26.10.2022 wird die Engine-Version 1.2.0.1 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

Note

Bei einem Upgrade von einer Engine-Version vor 1.2.0.0:

- Mit der Engine-Version 1.2.0.0 wurde ein neues Format für benutzerdefinierte Parametergruppen und benutzerdefinierte Cluster-Parametergruppen eingeführt. Wenn Sie also von einer Engine-Version vor 1.2.0.0 auf Engine-Version 1.2.0.0 oder höher aktualisieren, müssen Sie alle vorhandenen benutzerdefinierten Parametergruppen und benutzerdefinierten Cluster-Parametergruppen mithilfe der Parametergruppenfamilie `neptune1.2` neu erstellen. In früheren Versionen wurde die Parametergruppenfamilie `neptune1` verwendet und diese Parametergruppen funktionieren nicht mit Version 1.2.0.0 und höher. Weitere Informationen finden Sie unter Amazon-Neptune-Parametergruppen.
- Mit der Engine-Version 1.2.0.0 wurde auch ein neues Format für Undo-Protokolle eingeführt. Daher müssen alle Undo-Logs, die von einer früheren Engine-Version erstellt wurden, gelöscht werden und die `UndoLogsListSize` CloudWatch Metrik muss auf Null fallen, bevor ein Upgrade von einer Version vor 1.2.0.0 gestartet werden kann. Wenn beim Versuch, ein Update zu starten, zu viele Undo-Protokolldatensätze (200.000 oder mehr) vorhanden sind, kann es beim Upgrade-Versuch zu einer Zeitüberschreitung kommen, während auf den Abschluss des Löschvorgangs der Undo-Protokolle gewartet wird.

  Sie können die Löschgeschwindigkeit beschleunigen, indem Sie die Writer-Instance des Clusters aktualisieren, in der das Löschen stattfindet. Wenn Sie dies tun, bevor Sie versuchen, das Upgrade durchzuführen, kann die Anzahl der Undo-Logs vor dem Start reduziert werden. Wenn Sie die Größe des Writers auf einen 24XL-Instance-Typ erhöhen, kann sich Ihre Löschrate auf mehr als eine Million Datensätze pro Stunde erhöhen.

  Wenn die `UndoLogsListSize` CloudWatch Metrik extrem umfangreich ist, kann Ihnen die Eröffnung eines Support-Falls dabei helfen, zusätzliche Strategien zu finden, um sie zu reduzieren.
- Schließlich wurde mit Version 1.2.0.0 eine bahnbrechende Änderung eingeführt, die sich auf früheren Code auswirkt, der das Bolt-Protokoll mit IAM-Authentifizierung verwendete. Ab Version 1.2.0.0 benötigt Bolt einen Ressourcenpfad für die IAM-Signatur. In Java könnte das Festlegen des Ressourcenpfads so aussehen:

`request.setResourcePath("/openCypher"));`. In anderen Sprachen kann `/openCypher` an den Endpunkt-URI angehängt werden. Beispiele finden Sie unter Verwenden des Bolt-Protokolls.

Nachfolgende Patch-Veröffentlichungen für dieses Version

- Wartungs-Release: 1.2.0.1.R2 (13.12.2022)
- Wartungs-Release: 1.2.0.1.R3 (27.09.2023)

Neue Features in dieser Engine-Version

- Einführung von Amazon Neptune Serverless, einer On-Demand-Konfiguration für automatische Skalierung, die Ihren DB-Cluster hochskaliert, um dem steigenden Verarbeitungsbedarf gerecht zu werden, und dann wieder herunterskaliert, wenn der Bedarf sinkt.

Verbesserungen in dieser Engine-Version

- Verbesserte Leistung von Gremlin-`order-by`-Abfragen. Gremlin-Abfragen mit einem `order-by` am Ende eines `NeptuneGraphQueryStep` verwenden jetzt eine größere Chunk-Größe für bessere Leistung. Dies gilt nicht für `order-by` auf einem internen (Nicht-Wurzel-)Knoten des Abfrageplans.
- Verbesserte Leistung von Gremlin-Update-Abfragen. Scheitelpunkte und Kanten müssen jetzt beim Hinzufügen von Edges oder Eigenschaften gegen Löschen gesperrt werden. Durch diese Änderung werden doppelte Sperren innerhalb einer Transaktion beseitigt, wodurch die Leistung verbessert wird.
- Die Leistung von Gremlin-Abfragen, die `dedup()` innerhalb einer `repeat()` Unterabfrage verwendet werden, wurde verbessert, indem sie auf die native `dedup` Ausführungsebene verschoben wurden.
- Es wurden benutzerfreundliche Fehlermeldungen für IAM-Authentifizierungsfehler hinzugefügt. Diese Nachrichten zeigen jetzt den ARN Ihres IAM-Benutzers oder Ihrer Rolle, den Ressourcen-ARN und eine Liste der nicht autorisierten Aktionen für die Anfrage. Anhand der Liste der nicht autorisierten Aktionen können Sie erkennen, was in der von Ihnen verwendeten IAM-Richtlinie möglicherweise fehlt oder ausdrücklich verweigert wurde.

In diesem Engine-Version behobene Fehler

- Es wurde ein Gremlin-Fehler behoben, bei dem die Verwendung `PartitionStrategy` nach dem Upgrade auf TinkerPop 3.5 fälschlicherweise zu einem Fehler mit der Meldung „PartitionStrategy Funktioniert nicht mit anonymen Durchläufen“ führte, wodurch die Ausführung des Traversals verhindert wurde.
- Es wurde ein Gremlin-Korrekturfehler im Zusammenhang mit `WherePredicateStep`-Übersetzungen behoben, bei dem die Abfrage-Engine von Neptune falsche Ergebnisse für Abfragen mit `where(P.neq('x'))` und Varianten davon lieferte.
- Ein openCypher-Fehler in der `MERGE`-Klausel wurde behoben, der in einigen Fällen zur doppelten Erstellung von Knoten und Edges führte.
- Es wurde ein SPARQL-Fehler bei der Behandlung von Abfragen behoben, die `(NOT) EXISTS` innerhalb einer `OPTIONAL`-Klausel enthalten waren, wodurch in einigen Fällen Abfrageergebnisse fehlten.
- Es wurde ein Massenladerfehler behoben, der bei hohen Einfügebelatungen zu Leistungsregressionen führte.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.2.0.1 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprache kompatibel ist:

- Die älteste unterstützte Version von Gremlin: `3.5.2`
- Die neueste unterstützte Version von Gremlin: `3.5.4`
- openCypher-Version: `Neptune-9.0.20190305-1.0`
- SPARQL-Version: `1.1`

Upgrade-Pfade zum Engine-Release 1.2.0.1

Upgrade auf diesen Release

Amazon Neptune 1.2.0.1 ist jetzt allgemein verfügbar.

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe

der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.1 \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.1 ^
    --apply-immediately
```

Updates werden auf alle Instances in einem DB-Cluster gleichzeitig angewendet. Ein Update erfordert einen Datenbankneustart auf diesen Instances. Daher kommt es zu einer Ausfallzeit von 20–30 Sekunden bis zu mehreren Minuten. Anschließend können Sie die Nutzung Ihres DB-Clusters fortsetzen.

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
> 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.
> ```
>
> Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine-Version 1.2.0.1.R3 (27.09.2023)

Ab dem 27.09.2023 wird die Engine-Version 1.2.0.1.R3 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

> **Note**
>
> Bei einem Upgrade von einer Engine-Version vor 1.2.0.0:
>
> - Mit der Engine-Version 1.2.0.0 wurde ein neues Format für benutzerdefinierte Parametergruppen und benutzerdefinierte Cluster-Parametergruppen eingeführt. Wenn Sie also von einer Engine-Version vor 1.2.0.0 auf Engine-Version 1.2.0.0 oder höher aktualisieren, müssen Sie alle vorhandenen benutzerdefinierten Parametergruppen und benutzerdefinierten Cluster-Parametergruppen mithilfe der Parametergruppenfamilie `neptune1.2` neu erstellen. In früheren Versionen wurde die Parametergruppenfamilie `neptune1` verwendet und diese Parametergruppen funktionieren nicht mit Version 1.2.0.0 und höher. Weitere Informationen finden Sie unter Amazon-Neptune-Parametergruppen.
> - Mit der Engine-Version 1.2.0.0 wurde auch ein neues Format für Undo-Protokolle eingeführt. Daher müssen alle Undo-Logs, die von einer früheren Engine-Version erstellt wurden, gelöscht werden und die `UndoLogsListSize` CloudWatch Metrik muss auf Null fallen, bevor ein Upgrade von einer Version vor 1.2.0.0 gestartet werden kann. Wenn beim Versuch, ein Update zu starten, zu viele Undo-Protokolldatensätze (200.000 oder mehr) vorhanden sind, kann es beim Upgrade-Versuch zu einer Zeitüberschreitung kommen, während auf den Abschluss des Löschvorgangs der Undo-Protokolle gewartet wird.
>
>   Sie können die Löschgeschwindigkeit beschleunigen, indem Sie die Writer-Instance des Clusters aktualisieren, in der das Löschen stattfindet. Wenn Sie dies tun, bevor Sie versuchen, das Upgrade durchzuführen, kann die Anzahl der Undo-Logs vor dem Start reduziert werden. Wenn Sie die Größe des Writers auf einen 24XL-Instance-Typ erhöhen, kann sich Ihre Löschrate auf mehr als eine Million Datensätze pro Stunde erhöhen.
>
>   Wenn die `UndoLogsListSize` CloudWatch Metrik extrem umfangreich ist, kann Ihnen die Eröffnung eines Support-Falls dabei helfen, zusätzliche Strategien zu finden, um sie zu reduzieren.
> - Schließlich wurde mit Version 1.2.0.0 eine bahnbrechende Änderung eingeführt, die sich auf früheren Code auswirkt, der das Bolt-Protokoll mit IAM-Authentifizierung verwendete. Ab Version 1.2.0.0 benötigt Bolt einen Ressourcenpfad für die IAM-Signatur. In Java könnte das Festlegen des Ressourcenpfads so aussehen:

`request.setResourcePath("/openCypher"));`. In anderen Sprachen kann `/openCypher` an den Endpunkt-URI angehängt werden. Beispiele finden Sie unter Verwenden des Bolt-Protokolls.

Verbesserungen in dieser Engine-Version

- Allen Neptune ML wurde ein `enableInterContainerTrafficEncryption` Parameter hinzugefügt APIs, mit dem Sie die Verschlüsselung des Datenverkehrs zwischen Containern in Trainings- oder Hyperparameter-Tuning-Jobs aktivieren und deaktivieren können.
- Für serverlose DB-Cluster wurde die Einstellung für die Mindestkapazität auf 1,0 NCU und die niedrigste gültige Maximaleinstellung auf 2,5 geändert. NCUs Siehe Kapazitätsskalierung in einem Neptune-Serverless-DB-Cluster *(((before release, this change needs to be reflected in the serverless page too)))*.

In diesem Engine-Version behobene Fehler

- Es wurde ein OpenCypher-Fehler behoben, bei dem update-and-return Abfragen nicht oder nicht richtig behandelt `orderBy` wurden. `limit skip`
- Es wurde ein openCypher-Fehler behoben, durch den Parameter, die in einer Anfrage enthalten waren, durch Parameter in einer anderen gleichzeitigen Anfrage überschrieben werden konnten.
- Es wurde ein openCypher-Fehler bei der Transaktionsverarbeitung von Bolt behoben.
- Behebung von Gremlin-Korrektheitsproblemen bei DFE-Abfragen mit `limit` als untergeordneter Traversal von Nicht-Union-Schritten durch Rückgriff auf Tinkerpop. Zum Beispiel für Abfragen wie diese:

```
g.withSideEffect('Neptune#useDFE', true)
 .V()
 .as("a")
 .select("a")
 .by(out()
 .limit(1))
```

- Es wurde ein Gremlin-Fehler behoben, bei dem eine Abfrage fehlschlug, weil sie zu viele TinkerPop Schritte enthielt, und dann nicht bereinigt wurde.
- Es wurde ein Gremlin-Fehler behoben, bei dem Zeichenkettenausgaben von `order()` nicht richtig sortiert wurden, wenn einige von ihnen ein Leerzeichen enthielten.

- Es wurde ein Gremlin-Fehler behoben, durch den ein Transaktionsleck auftreten konnte, wenn eine Abfrage als Zeichenfolge gesendet wurde und `GroupCountStep` enthalten hat.
- Es wurde ein Gremlin-Fehler behoben, bei dem es zu einem Transaktionsleck kam, wenn der Endpunkt des Gremlin-Abfragestatus auf Abfragen mit Prädikaten in untergeordneten Durchläufen für Schritte überprüft wurde, die nicht nativ verarbeitet wurden.
- Es wurde ein Gremlin-Fehler behoben, durch den beim Hinzufügen einer Edge und ihrer Eigenschaften gefolgt von `inV()` oder `outV()` ein `InternalFailureException` verursacht wurde.
- Ein Problem mit der Parallelität in der Speicherebene wurde behoben.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.2.0.1.R3 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprache kompatibel ist:

- Die älteste unterstützte Version von Gremlin: `3.5.2`
- Die neueste unterstützte Version von Gremlin: `3.5.6`
- openCypher-Version: `Neptune-9.0.20190305-1.0`
- SPARQL-Version: `1.1`

Upgrade-Pfade zum Engine-Release 1.2.0.1.R3

Ihr Cluster wird während des nächsten Wartungsfensters automatisch auf diese Patch-Version aktualisiert, wenn Sie die Engine-Version 1.2.0.1 ausführen.

Upgrade auf diesen Release

Amazon Neptune 1.2.0.1.R3 ist jetzt allgemein verfügbar.

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
```

```
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.1 \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.1 ^
    --apply-immediately
```

Updates werden auf alle Instances in einem DB-Cluster gleichzeitig angewendet. Ein Update erfordert einen Datenbankneustart auf diesen Instances. Daher kommt es zu einer Ausfallzeit von 20–30 Sekunden bis zu mehreren Minuten. Anschließend können Sie die Nutzung Ihres DB-Clusters fortsetzen.

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
> 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.
> ```
>
> Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine-Version 1.2.0.1.R2 (13.12.2022)

Ab dem 13.12.2022 wird die Engine-Version 1.2.0.1.R2 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

> **Note**
>
> Bei einem Upgrade von einer Engine-Version vor 1.2.0.0:
>
> - Mit der Engine-Version 1.2.0.0 wurde ein neues Format für benutzerdefinierte Parametergruppen und benutzerdefinierte Cluster-Parametergruppen eingeführt. Wenn Sie also von einer Engine-Version vor 1.2.0.0 auf Engine-Version 1.2.0.0 oder höher

aktualisieren, müssen Sie alle vorhandenen benutzerdefinierten Parametergruppen und benutzerdefinierten Cluster-Parametergruppen mithilfe der Parametergruppenfamilie `neptune1.2` neu erstellen. In früheren Versionen wurde die Parametergruppenfamilie `neptune1` verwendet und diese Parametergruppen funktionieren nicht mit Version 1.2.0.0 und höher. Weitere Informationen finden Sie unter Amazon-Neptune-Parametergruppen.

- Mit der Engine-Version 1.2.0.0 wurde auch ein neues Format für Undo-Protokolle eingeführt. Daher müssen alle Undo-Logs, die von einer früheren Engine-Version erstellt wurden, gelöscht werden und die `UndoLogsListSize` CloudWatch Metrik muss auf Null fallen, bevor ein Upgrade von einer Version vor 1.2.0.0 gestartet werden kann. Wenn beim Versuch, ein Update zu starten, zu viele Undo-Protokolldatensätze (200.000 oder mehr) vorhanden sind, kann es beim Upgrade-Versuch zu einer Zeitüberschreitung kommen, während auf den Abschluss des Löschvorgangs der Undo-Protokolle gewartet wird.

  Sie können die Löschgeschwindigkeit beschleunigen, indem Sie die Writer-Instance des Clusters aktualisieren, in der das Löschen stattfindet. Wenn Sie dies tun, bevor Sie versuchen, das Upgrade durchzuführen, kann die Anzahl der Undo-Logs vor dem Start reduziert werden. Wenn Sie die Größe des Writers auf einen 24XL-Instance-Typ erhöhen, kann sich Ihre Löschrate auf mehr als eine Million Datensätze pro Stunde erhöhen.

  Wenn die `UndoLogsListSize` CloudWatch Metrik extrem umfangreich ist, kann Ihnen die Eröffnung eines Support-Falls dabei helfen, zusätzliche Strategien zu finden, um sie zu reduzieren.
- Schließlich wurde mit Version 1.2.0.0 eine bahnbrechende Änderung eingeführt, die sich auf früheren Code auswirkt, der das Bolt-Protokoll mit IAM-Authentifizierung verwendete. Ab Version 1.2.0.0 benötigt Bolt einen Ressourcenpfad für die IAM-Signatur. In Java könnte das Festlegen des Ressourcenpfads so aussehen: `request.setResourcePath("/openCypher"));`. In anderen Sprachen kann `/openCypher` an den Endpunkt-URI angehängt werden. Beispiele finden Sie unter Verwenden des Bolt-Protokolls.

Verbesserungen in dieser Engine-Version

- Die Leistung von openCypher-Abfragen, die `UNWIND` mit einer Liste von Zuordnungen mit Literalwerten beinhalten, wurde verbessert.
- Leistungsverbesserungen und Korrekturkorrekturen für verschiedene Gremlin-Operatoren, darunter `repeat`, `coalesce`, `store` und `aggregate`.

In diesem Engine-Version behobene Fehler

- Es wurde ein openCypher-Fehler behoben, bei dem Abfragen die Zeichenfolge, `"null"`, anstelle eines Nullwerts in Bolt und SPARQL-JSON zurückgaben.
- Es wurde ein Fehler im Auditprotokoll behoben, der dazu führte, dass unnötige Informationen protokolliert wurden und bestimmte Felder in den Protokollen fehlten.
- Es wurde ein Fehler im Lookup-Cache behoben, sodass der inkrementelle Speicher, der für Schreibvorgänge in den Cache verwendet wurde, begrenzt wurde.
- Es wurde ein Lookup-Cache-Fehler behoben, bei dem für den Lookup-Cache der Nur-Lese-Modus eingestellt wurde, wenn Schreibvorgänge fehlschlugen.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.2.0.1.R2 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprache kompatibel ist:

- Die älteste unterstützte Version von Gremlin: `3.5.2`
- Die neueste unterstützte Version von Gremlin: `3.5.4`
- openCypher-Version: `Neptune-9.0.20190305-1.0`
- SPARQL-Version: `1.1`

Upgrade-Pfade zum Engine-Release 1.2.0.1.R2

Ihr Cluster wird während des nächsten Wartungsfensters automatisch auf diese Patch-Version aktualisiert, wenn Sie die Engine-Version 1.2.0.1 ausführen.

Upgrade auf diesen Release

Amazon Neptune 1.2.0.1.R2 ist jetzt allgemein verfügbar.

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
```

```
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.1 \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.1 ^
    --apply-immediately
```

Updates werden auf alle Instances in einem DB-Cluster gleichzeitig angewendet. Ein Update erfordert einen Datenbankneustart auf diesen Instances. Daher kommt es zu einer Ausfallzeit von 20–30 Sekunden bis zu mehreren Minuten. Anschließend können Sie die Nutzung Ihres DB-Clusters fortsetzen.

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
> 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.
> ```
>
> Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine-Version 1.2.0.0 (21.07.2022)

Ab dem 21.07.2022 wird die Engine-Version 1.2.0.0 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

> **Note**
>
> Bei einem Upgrade von einer Engine-Version vor 1.2.0.0:
>
> - Mit der Engine-Version 1.2.0.0 wurde ein neues Format für benutzerdefinierte Parametergruppen und benutzerdefinierte Cluster-Parametergruppen eingeführt. Wenn Sie also von einer Engine-Version vor 1.2.0.0 auf Engine-Version 1.2.0.0 oder höher

aktualisieren, müssen Sie alle vorhandenen benutzerdefinierten Parametergruppen und benutzerdefinierten Cluster-Parametergruppen mithilfe der Parametergruppenfamilie `neptune1.2` neu erstellen. In früheren Versionen wurde die Parametergruppenfamilie `neptune1` verwendet und diese Parametergruppen funktionieren nicht mit Version 1.2.0.0 und höher. Weitere Informationen finden Sie unter Amazon-Neptune-Parametergruppen.

- Mit der Engine-Version 1.2.0.0 wurde auch ein neues Format für Undo-Protokolle eingeführt. Daher müssen alle Undo-Logs, die von einer früheren Engine-Version erstellt wurden, gelöscht werden und die `UndoLogsListSize` CloudWatch Metrik muss auf Null fallen, bevor ein Upgrade von einer Version vor 1.2.0.0 gestartet werden kann. Wenn beim Versuch, ein Update zu starten, zu viele Undo-Protokolldatensätze (200.000 oder mehr) vorhanden sind, kann es beim Upgrade-Versuch zu einer Zeitüberschreitung kommen, während auf den Abschluss des Löschvorgangs der Undo-Protokolle gewartet wird.

  Sie können die Löschgeschwindigkeit beschleunigen, indem Sie die Writer-Instance des Clusters aktualisieren, in der das Löschen stattfindet. Wenn Sie dies tun, bevor Sie versuchen, das Upgrade durchzuführen, kann die Anzahl der Undo-Logs vor dem Start reduziert werden. Wenn Sie die Größe des Writers auf einen 24XL-Instance-Typ erhöhen, kann sich Ihre Löschrate auf mehr als eine Million Datensätze pro Stunde erhöhen.

  Wenn die `UndoLogsListSize` CloudWatch Metrik extrem umfangreich ist, kann Ihnen die Eröffnung eines Support-Falls dabei helfen, zusätzliche Strategien zu finden, um sie zu reduzieren.

- Schließlich wurde mit Version 1.2.0.0 eine bahnbrechende Änderung eingeführt, die sich auf früheren Code auswirkt, der das Bolt-Protokoll mit IAM-Authentifizierung verwendete. Ab Version 1.2.0.0 benötigt Bolt einen Ressourcenpfad für die IAM-Signatur. In Java könnte das Festlegen des Ressourcenpfads so aussehen: `request.setResourcePath("/openCypher"));`. In anderen Sprachen kann `/openCypher` an den Endpunkt-URI angehängt werden. Beispiele finden Sie unter Verwenden des Bolt-Protokolls.

Nachfolgende Patch-Veröffentlichungen für dieses Version

- Release: 1.2.0.0.R2 (14.10.2022)
- Release: 1.2.0.0.R3 (15.12.2022)
- Release: 1.2.0.0.R4 (29.09.2023)

Neue Features in dieser Engine-Version

- Unterstützung für globale Datenbanken hinzugefügt. Eine globale Neptune-Datenbank umfasst mehrere AWS-Regionen Datenbanken und besteht aus einem primären DB-Cluster in einer Region und bis zu fünf sekundären DB-Clustern in anderen Regionen.
- Unterstützung für eine detailliertere Zugriffskontrolle in Neptune IAM-Richtlinien als bisher hinzugefügt, basierend auf Aktionen auf Datenebene. Dies ist insofern eine signifikante Änderung, als bestehende IAM-Richtlinien, die auf der veralteten `connect`-Aktion basieren, angepasst werden müssen, um die detaillierteren Aktionen auf der Datenebene zu verwenden. Siehe Arten von IAM-Richtlinien.
- Verbesserte Verfügbarkeit von Reader-Instances Zuvor wurden, als eine Writer-Instance neu gestartet wurde, alle Reader-Instances in einem Neptune-Cluster automatisch neu gestartet. Ab Engine-Version 1.2.0.0 bleiben Reader-Instances auch nach einem Writer-Neustart aktiv, was die Verfügbarkeit der Reader verbessert. Reader-Instances können separat neu gestartet werden, um Änderungen an Parametergruppen zu übernehmen. Siehe Neustarten einer DB-Instance in Amazon Neptune.
- Es wurde ein neuer DB-Cluster-Parameter neptune_streams_expiry_days hinzugefügt, mit dem Sie festlegen können, wie viele Tage Stream-Datensätze auf dem Server aufbewahrt werden, bevor sie gelöscht werden. Der Bereich ist 1 bis 90 und der Standard ist 7.

Verbesserungen in dieser Engine-Version

- Die Leistung der Gremlin-Serialisierung für Abfragen wurde verbessert. ByteCode
- Neptune verarbeitet Textprädikate jetzt mithilfe der DFE-Engine, um die Leistung zu verbessern.
- Neptune verarbeitet `limit()` Gremlin-Schritte jetzt mithilfe der DFE-Engine, einschließlich Grenzwerten für Nichtterminal- und untergeordneten Traversal-Grenzwerte.
- Die DFE-Behandlung des Gremlin-`union()`-Schritts wurde geändert, sodass er mit anderen neuen Features funktioniert. Das bedeutet, dass Referenzknoten wie erwartet in Abfrageprofilen angezeigt werden.
- Die Leistung einiger teurer Join-Operationen innerhalb von DFE wurde um den Faktor 5 verbessert, indem sie parallelisiert wurden.
- `by()` Modulationsunterstützung für `OrderGlobalStep order(global)` für die Gremlin DFE-Engine hinzugefügt.
- Die Anzeige eingefügter statischer Werte wurde in den Erläuterungsdetails für DFE hinzugefügt.

- Die Leistung beim Löschen doppelter Muster wurde verbessert.
- Unterstützung für die Beibehaltung von Aufträgen in der Gremlin DFE-Engine hinzugefügt.
- Die Leistung von Gremlin-Abfragen mit leeren Filtern wie diesen wurde verbessert:

```
g.V().hasId(P.within([]))
```

```
g.V().hasId([])
```

- Fehlermeldungen wurden verbessert, wenn eine SPARQL-Abfrage einen numerischen Wert verwendet, der für Neptune zu groß ist, um ihn intern darzustellen.
- Die Leistung beim Löschen von Scheitelpunkten mit zugehörigen Edges wurde verbessert, indem die Anzahl der Indexsuchen reduziert wurde, wenn Streams deaktiviert sind.
- Die DFE-Unterstützung wurde um mehr Varianten des `has()` Schritts erweitert, insbesondere um die Prädikate to `hasKey()``hasLabel()`, und Range for within. strings/URIs `has()` Dies betrifft Abfragen wie die folgende:

```
// 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)"))
```

- Es wurde eine Neptune-spezifische openCypher-`join()`Funktion hinzugefügt, die Zeichenketten in einer Liste zu einer einzigen Zeichenkette verkettet.
- Die von Neptune verwalteten Richtlinien wurden aktualisiert und enthalten nun Datenzugriffsberechtigungen und Berechtigungen für die neue globale Datenbank. APIs

In diesem Engine-Version behobene Fehler

- Es wurde ein Fehler behoben, bei dem eine HTTP-Anfrage ohne angegebenen Inhaltstyp automatisch fehlschlug.

- Es wurde ein SPARQL-Fehler im Abfrageoptimierer behoben, der die Verwendung eines Serviceaufrufs innerhalb einer Abfrage verhinderte.
- Es wurde ein SPARQL-Fehler im Turtle RDF-Parser behoben, bei dem eine bestimmte Kombination von Unicode-Daten zu einem Ausfall führte.
- Es wurde ein SPARQL-Fehler behoben, bei dem eine bestimmte Kombination von `GRAPH`- und `SELECT`-Klauseln zu falschen Abfrageergebnissen führte.
- Es wurde ein Gremlin-Fehler behoben, der zu einem Korrektheitsproblem bei Abfragen führte, die einen beliebigen Filterschritt innerhalb eines Union-Schritts verwendeten, wie zum Beispiel den folgenden:

```
g.V("1").union(hasLabel("person"), out())
```

- Es wurde ein Gremlin-Fehler behoben, bei dem `count()` von `both().simplePath()` zu einer doppelten Anzahl von Ergebnissen führen würde, die ohne `count()` zurückgegeben wurden.
- Es wurde ein openCypher-Fehler behoben, bei dem vom Server für Bolt-Anfragen an Cluster mit aktivierter IAM-Authentifizierung eine fehlerhafte Ausnahme wegen Nichtübereinstimmung der Signatur generiert wurde.
- Es wurde ein openCypher-Fehler behoben, bei dem eine Abfrage, die HTTP-Keep-Alive verwendete, fälschlicherweise geschlossen werden konnte, wenn sie nach einer fehlgeschlagenen Anfrage eingereicht wurde.
- Es wurde ein openCypher-Fehler behoben, der dazu führen konnte, dass ein interner Fehler ausgelöst wurde, wenn eine Abfrage, die einen konstanten Wert zurückgibt, gesendet wurde.
- Es wurde ein Fehler in den Erläuterungsdetails behoben, sodass die DFE-Unterabfrage die CPU-Zeiten von Operatoren innerhalb der DFE-Unterabfrage `Time(ms)` jetzt korrekt summiert. Betrachten Sie den folgenden Auszug aus der Erklärungsausgabe als Beispiel:

```
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      #
##########################################################################################
```

Die subQuery-Zeiten in der letzten Spalte der unteren Tabelle ergeben zusammen 0,36 ms (`.04 + .29 + .01 + .02 = .36`). Wenn Sie die Koordinationszeit für diese Unterabfrage (`.36 + .026 = .386`) hinzurechnen, erhalten Sie ein Ergebnis, das der Zeit für die subQuery, die in der letzten Spalte der oberen Tabelle aufgezeichnet wurde, sehr nahe kommt, nämlich `0.38` ms.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.2.0.0 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprache kompatibel ist:

- Die älteste unterstützte Version von Gremlin: `3.5.2`
- Die neueste unterstützte Version von Gremlin: `3.5.4`
- openCypher-Version: `Neptune-9.0.20190305-1.0`
- SPARQL-Version: `1.1`

Upgrade-Pfade zum Engine-Release 1.2.0.0

Da es sich um eine wichtige Engine-Version handelt, gibt es kein automatisches Upgrade darauf.

Sie können nur `1.2.0.0` manuell vom neuesten Patch-Release des Engine-Release 1.1.1.0 auf Release aktualisieren. Frühere Engine-Versionen müssen zuerst auf die neueste Version von `1.1.1.0` aktualisiert werden, bevor sie auf `1.2.0.0` aktualisiert werden können.

Bevor Sie versuchen, auf diese Version zu aktualisieren, sollten Sie daher überprüfen, ob Sie derzeit den neuesten Patch-Release von Release `1.1.1.0` verwenden. Falls nicht, führen Sie zunächst ein Upgrade auf den neuesten Patch-Release von `1.1.1.0` durch.

Vor dem Upgrade müssen Sie außerdem alle benutzerdefinierten DB-Cluster-Parametergruppen, die Sie mit Ihrer vorherigen Version verwendet haben, mithilfe der Parametergruppenfamilie `neptune1.2` neu erstellen. Weitere Informationen finden Sie unter Amazon-Neptune-Parametergruppen.

Wenn Sie zuerst auf Version `1.1.1.0` und dann sofort auf Version `1.2.0.0` aktualisieren, tritt möglicherweise ein Fehler wie der folgende auf:

```
  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.
```

Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann (siehe Warten eines Amazon-Neptune-DB-Clusters).

Upgrade auf diesen Release

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.0 \
    --allow-major-version-upgrade \
```

```
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Statt `--apply-immediately` können Sie `--no-apply-immediately` angeben. Um ein Upgrade einer Hauptversion durchzuführen, ist der allow-major-version-upgrade Parameter erforderlich. Stellen Sie außerdem sicher, dass Sie die Engine-Version angeben, da Ihre Engine sonst möglicherweise auf eine andere Version aktualisiert wird.

Wenn Ihr Cluster eine benutzerdefinierte Cluster-Parametergruppe verwendet, müssen Sie diesen Parameter einschließen, um ihn zu anzugeben:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Ebenso sollte für Instances im Cluster, die eine benutzerdefinierte DB-Parametergruppe verwenden, dieser Parameter eingeschlossen werden, um ihn zu spezifizieren:

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue

Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
> 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.
> ```
>
> Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine-Version 1.2.0.0.R4 (29.09.2023)

Ab dem 29.09.2023 wird die Engine-Version 1.2.0.0.R4 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

Note

Bei einem Upgrade von einer Engine-Version vor 1.2.0.0:

- Mit der Engine-Version 1.2.0.0 wurde ein neues Format für benutzerdefinierte Parametergruppen und benutzerdefinierte Cluster-Parametergruppen eingeführt. Wenn Sie also von einer Engine-Version vor 1.2.0.0 auf Engine-Version 1.2.0.0 oder höher aktualisieren, müssen Sie alle vorhandenen benutzerdefinierten Parametergruppen und benutzerdefinierten Cluster-Parametergruppen mithilfe der Parametergruppenfamilie `neptune1.2` neu erstellen. In früheren Versionen wurde die Parametergruppenfamilie `neptune1` verwendet und diese Parametergruppen funktionieren nicht mit Version 1.2.0.0 und höher. Weitere Informationen finden Sie unter Amazon-Neptune-Parametergruppen.
- Mit der Engine-Version 1.2.0.0 wurde auch ein neues Format für Undo-Protokolle eingeführt. Daher müssen alle Undo-Logs, die von einer früheren Engine-Version erstellt wurden, gelöscht werden und die `UndoLogsListSize` CloudWatch Metrik muss auf Null fallen, bevor ein Upgrade von einer Version vor 1.2.0.0 gestartet werden kann. Wenn beim Versuch, ein Update zu starten, zu viele Undo-Protokolldatensätze (200.000 oder mehr) vorhanden sind, kann es beim Upgrade-Versuch zu einer Zeitüberschreitung kommen, während auf den Abschluss des Löschvorgangs der Undo-Protokolle gewartet wird.

  Sie können die Löschgeschwindigkeit beschleunigen, indem Sie die Writer-Instance des Clusters aktualisieren, in der das Löschen stattfindet. Wenn Sie dies tun, bevor Sie versuchen, das Upgrade durchzuführen, kann die Anzahl der Undo-Logs vor dem Start reduziert werden. Wenn Sie die Größe des Writers auf einen 24XL-Instance-Typ erhöhen, kann sich Ihre Löschrate auf mehr als eine Million Datensätze pro Stunde erhöhen.

  Wenn die `UndoLogsListSize` CloudWatch Metrik extrem umfangreich ist, kann Ihnen die Eröffnung eines Support-Falls dabei helfen, zusätzliche Strategien zu finden, um sie zu reduzieren.
- Schließlich wurde mit Version 1.2.0.0 eine bahnbrechende Änderung eingeführt, die sich auf früheren Code auswirkt, der das Bolt-Protokoll mit IAM-Authentifizierung verwendete. Ab Version 1.2.0.0 benötigt Bolt einen Ressourcenpfad für die IAM-Signatur. In Java könnte das Festlegen des Ressourcenpfads so aussehen:

`request.setResourcePath("/openCypher"));`. In anderen Sprachen kann `/openCypher` an den Endpunkt-URI angehängt werden. Beispiele finden Sie unter Verwenden des Bolt-Protokolls.

Verbesserungen in dieser Engine-Version

- Allen Neptune ML wurde ein `enableInterContainerTrafficEncryption` Parameter hinzugefügt APIs, mit dem Sie die Verschlüsselung des Datenverkehrs zwischen Containern in Trainings- oder Hyperparameter-Tuning-Jobs aktivieren und deaktivieren können.
- Für serverlose DB-Cluster wurde die Einstellung für die Mindestkapazität auf 1,0 NCU und die niedrigste gültige Maximaleinstellung auf 2,5 geändert. NCUs Siehe Kapazitätsskalierung in einem Neptune-Serverless-DB-Cluster *(((before release, this change needs to be reflected in the serverless page too)))*.

In diesem Engine-Version behobene Fehler

- Es wurde ein OpenCypher-Fehler behoben, bei dem update-and-return Abfragen nicht oder nicht richtig behandelt `orderBy` wurden. `limit skip`
- Es wurde ein openCypher-Fehler behoben, durch den Parameter, die in einer Anfrage enthalten waren, durch Parameter in einer anderen gleichzeitigen Anfrage überschrieben werden konnten.
- Es wurde ein openCypher-Fehler bei der Transaktionsverarbeitung von Bolt behoben.
- Behebung von Gremlin-Korrektheitsproblemen bei DFE-Abfragen mit `limit` als untergeordneter Traversal von Nicht-Union-Schritten durch Rückgriff auf Tinkerpop. Zum Beispiel für Abfragen wie diese:

```
g.withSideEffect('Neptune#useDFE', true)
 .V()
 .as("a")
 .select("a")
 .by(out()
 .limit(1))
```

- Es wurde ein Gremlin-Fehler behoben, bei dem eine Abfrage fehlschlug, weil sie zu viele TinkerPop Schritte enthielt, und dann nicht bereinigt wurde.
- Es wurde ein Gremlin-Fehler behoben, bei dem Zeichenkettenausgaben von `order()` nicht richtig sortiert wurden, wenn einige von ihnen ein Leerzeichen enthielten.

- Es wurde ein Gremlin-Fehler behoben, durch den ein Transaktionsleck auftreten konnte, wenn eine Abfrage als Zeichenfolge gesendet wurde und `GroupCountStep` enthalten hat.
- Es wurde ein Gremlin-Fehler behoben, bei dem es zu einem Transaktionsleck kam, wenn der Endpunkt des Gremlin-Abfragestatus auf Abfragen mit Prädikaten in untergeordneten Durchläufen für Schritte überprüft wurde, die nicht nativ verarbeitet wurden.
- Es wurde ein Gremlin-Fehler behoben, durch den beim Hinzufügen einer Edge und ihrer Eigenschaften gefolgt von `inV()` oder `outV()` ein `InternalFailureException` verursacht wurde.
- Ein Problem mit der Parallelität in der Speicherebene wurde behoben.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.2.0.0.R4 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprache kompatibel ist:

- Die älteste unterstützte Version von Gremlin: `3.5.2`
- Die neueste unterstützte Version von Gremlin: `3.5.6`
- openCypher-Version: `Neptune-9.0.20190305-1.0`
- SPARQL-Version: `1.1`

Upgrade-Pfade zum Engine-Release 1.2.0.0.R4

Ihr Cluster wird während des nächsten Wartungsfensters automatisch auf diese Patch-Version aktualisiert, wenn Sie die Modulversion `1.2.0.0` ausführen.

Sie können nur `1.2.0.0` manuell vom neuesten Patch-Release des Engine-Release 1.1.1.0 auf Release aktualisieren. Frühere Engine-Versionen müssen zuerst auf die neueste Version von `1.1.1.0` aktualisiert werden, bevor sie auf `1.2.0.0` aktualisiert werden können.

Wenn Sie zuerst auf Version `1.1.1.0` und dann sofort auf Version `1.2.0.0` aktualisieren, tritt möglicherweise ein Fehler wie der folgende auf:

```
  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.
```

Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Upgrade auf diesen Release

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Statt `--apply-immediately` können Sie `--no-apply-immediately` angeben. Um ein Hauptversions-Upgrade durchzuführen, ist der allow-major-version-upgrade Parameter erforderlich. Stellen Sie außerdem sicher, dass Sie die Engine-Version angeben, da Ihre Engine sonst möglicherweise auf eine andere Version aktualisiert wird.

Wenn Ihr Cluster eine benutzerdefinierte Cluster-Parametergruppe verwendet, müssen Sie diesen Parameter einschließen, um ihn zu anzugeben:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Ebenso sollte für Instances im Cluster, die eine benutzerdefinierte DB-Parametergruppe verwenden, dieser Parameter eingeschlossen werden, um ihn zu spezifizieren:

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
> 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.
```

Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine-Version 1.2.0.0.R3 (15.12.2022)

Ab dem 15.12.2022 wird die Engine-Version 1.2.0.0.R3 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

Note

Bei einem Upgrade von einer Engine-Version vor 1.2.0.0:

- Mit der Engine-Version 1.2.0.0 wurde ein neues Format für benutzerdefinierte Parametergruppen und benutzerdefinierte Cluster-Parametergruppen eingeführt. Wenn Sie also von einer Engine-Version vor 1.2.0.0 auf Engine-Version 1.2.0.0 oder höher aktualisieren, müssen Sie alle vorhandenen benutzerdefinierten Parametergruppen und benutzerdefinierten Cluster-Parametergruppen mithilfe der Parametergruppenfamilie `neptune1.2` neu erstellen. In früheren Versionen wurde die Parametergruppenfamilie `neptune1` verwendet und diese Parametergruppen funktionieren nicht mit Version 1.2.0.0 und höher. Weitere Informationen finden Sie unter Amazon-Neptune-Parametergruppen.
- Mit der Engine-Version 1.2.0.0 wurde auch ein neues Format für Undo-Protokolle eingeführt. Daher müssen alle Undo-Logs, die von einer früheren Engine-Version erstellt wurden, gelöscht werden und die `UndoLogsListSize` CloudWatch Metrik muss auf Null fallen, bevor ein Upgrade von einer Version vor 1.2.0.0 gestartet werden kann. Wenn beim Versuch, ein Update zu starten, zu viele Undo-Protokolldatensätze (200.000 oder mehr) vorhanden sind, kann es beim Upgrade-Versuch zu einer Zeitüberschreitung kommen, während auf den Abschluss des Löschvorgangs der Undo-Protokolle gewartet wird.

Sie können die Löschgeschwindigkeit beschleunigen, indem Sie die Writer-Instance des Clusters aktualisieren, in der das Löschen stattfindet. Wenn Sie dies tun, bevor Sie versuchen, das Upgrade durchzuführen, kann die Anzahl der Undo-Logs vor dem Start reduziert werden. Wenn Sie die Größe des Writers auf einen 24XL-Instance-Typ erhöhen, kann sich Ihre Löschrate auf mehr als eine Million Datensätze pro Stunde erhöhen.

Wenn die `UndoLogsListSize` CloudWatch Metrik extrem umfangreich ist, kann Ihnen die Eröffnung eines Support-Falls dabei helfen, zusätzliche Strategien zu finden, um sie zu reduzieren.

- Schließlich wurde mit Version 1.2.0.0 eine bahnbrechende Änderung eingeführt, die sich auf früheren Code auswirkt, der das Bolt-Protokoll mit IAM-Authentifizierung verwendete. Ab Version 1.2.0.0 benötigt Bolt einen Ressourcenpfad für die IAM-Signatur. In Java könnte das Festlegen des Ressourcenpfads so aussehen: `request.setResourcePath("/openCypher"));`. In anderen Sprachen kann `/openCypher` an den Endpunkt-URI angehängt werden. Beispiele finden Sie unter Verwenden des Bolt-Protokolls.

Verbesserungen in dieser Engine-Version

- Verbesserte Leistung von openCypher-Abfragen mit `MERGE` und `OPTIONAL MATCH`.
- Die Leistung von openCypher-Abfragen, die `UNWIND` mit einer Liste von Zuordnungen mit Literalwerten beinhalten, wurde verbessert.
- Verbesserte Leistung von openCypher-Abfragen, die einen `IN`-Filter für `id` haben. Zum Beispiel:

```
MATCH (n) WHERE id(n) IN ['1', '2', '3'] RETURN n
```

- Leistungsverbesserungen und Korrekturkorrekturen für verschiedene Gremlin-Operatoren, darunter `repeat`, `coalesce`, `store` und `aggregate`.

In diesem Engine-Version behobene Fehler

- Es wurde ein openCypher-Fehler behoben, bei dem Abfragen die Zeichenfolge, `"null"`, anstelle eines Nullwerts in Bolt und SPARQL-JSON zurückgaben.
- Ein openCypher-Fehler wurde behoben, sodass der Parametertyp korrekt interpretiert werden konnte, wenn der Wert eine Liste oder eine Liste von Maps ist.

- Es wurde ein Fehler im Auditprotokoll behoben, der dazu führte, dass unnötige Informationen protokolliert wurden und bestimmte Felder in den Protokollen fehlten.
- Es wurde ein Auditprotokollfehler behoben, bei dem der IAM-ARN von HTTP-Anfragen an einen IAM-fähigen DB-Cluster nicht aufgezeichnet wurde.
- Es wurde ein Fehler im Lookup-Cache behoben, sodass der inkrementelle Speicher, der für Schreibvorgänge in den Cache verwendet wurde, begrenzt wurde.
- Es wurde ein Lookup-Cache-Fehler behoben, bei dem für den Lookup-Cache der Nur-Lese-Modus eingestellt wurde, wenn Schreibvorgänge fehlschlugen.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.2.0.0.R3 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprache kompatibel ist:

- Die älteste unterstützte Version von Gremlin: `3.5.2`
- Die neueste unterstützte Version von Gremlin: `3.5.4`
- openCypher-Version: `Neptune-9.0.20190305-1.0`
- SPARQL-Version: `1.1`

Upgrade-Pfade zum Engine-Release 1.2.0.0.R3

Ihr Cluster wird während des nächsten Wartungsfensters automatisch auf diese Patch-Version aktualisiert, wenn Sie die Modulversion `1.2.0.0` ausführen.

Sie können nur `1.2.0.0` manuell vom neuesten Patch-Release des Engine-Release 1.1.1.0 auf Release aktualisieren. Frühere Engine-Versionen müssen zuerst auf die neueste Version von `1.1.1.0` aktualisiert werden, bevor sie auf `1.2.0.0` aktualisiert werden können.

Wenn Sie zuerst auf Version `1.1.1.0` und dann sofort auf Version `1.2.0.0` aktualisieren, tritt möglicherweise ein Fehler wie der folgende auf:

```
  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.
```

Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Upgrade auf diesen Release

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Statt `--apply-immediately` können Sie `--no-apply-immediately` angeben. Um ein Upgrade auf eine Hauptversion durchzuführen, ist der allow-major-version-upgrade Parameter erforderlich. Stellen Sie außerdem sicher, dass Sie die Engine-Version angeben, da Ihre Engine sonst möglicherweise auf eine andere Version aktualisiert wird.

Wenn Ihr Cluster eine benutzerdefinierte Cluster-Parametergruppe verwendet, müssen Sie diesen Parameter einschließen, um ihn zu anzugeben:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Ebenso sollte für Instances im Cluster, die eine benutzerdefinierte DB-Parametergruppe verwenden, dieser Parameter eingeschlossen werden, um ihn zu spezifizieren:

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
> 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.
```

Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine-Version 1.2.0.0.R2 (14.10.2022)

Ab dem 14.10.2022 wird die Engine-Version 1.2.0.0.R2 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

Note

Bei einem Upgrade von einer Engine-Version vor 1.2.0.0:

- Mit der Engine-Version 1.2.0.0 wurde ein neues Format für benutzerdefinierte Parametergruppen und benutzerdefinierte Cluster-Parametergruppen eingeführt. Wenn Sie also von einer Engine-Version vor 1.2.0.0 auf Engine-Version 1.2.0.0 oder höher aktualisieren, müssen Sie alle vorhandenen benutzerdefinierten Parametergruppen und benutzerdefinierten Cluster-Parametergruppen mithilfe der Parametergruppenfamilie `neptune1.2` neu erstellen. In früheren Versionen wurde die Parametergruppenfamilie `neptune1` verwendet und diese Parametergruppen funktionieren nicht mit Version 1.2.0.0 und höher. Weitere Informationen finden Sie unter Amazon-Neptune-Parametergruppen.
- Mit der Engine-Version 1.2.0.0 wurde auch ein neues Format für Undo-Protokolle eingeführt. Daher müssen alle Undo-Logs, die von einer früheren Engine-Version erstellt wurden, gelöscht werden und die `UndoLogsListSize` CloudWatch Metrik muss auf Null fallen, bevor ein Upgrade von einer Version vor 1.2.0.0 gestartet werden kann. Wenn beim Versuch, ein Update zu starten, zu viele Undo-Protokolldatensätze (200.000 oder mehr) vorhanden sind, kann es beim Upgrade-Versuch zu einer Zeitüberschreitung kommen, während auf den Abschluss des Löschvorgangs der Undo-Protokolle gewartet wird.

Sie können die Löschgeschwindigkeit beschleunigen, indem Sie die Writer-Instance des Clusters aktualisieren, in der das Löschen stattfindet. Wenn Sie dies tun, bevor Sie versuchen, das Upgrade durchzuführen, kann die Anzahl der Undo-Logs vor dem Start reduziert werden. Wenn Sie die Größe des Writers auf einen 24XL-Instance-Typ erhöhen, kann sich Ihre Löschrate auf mehr als eine Million Datensätze pro Stunde erhöhen.

Wenn die `UndoLogsListSize` CloudWatch Metrik extrem umfangreich ist, kann Ihnen die Eröffnung eines Support-Falls dabei helfen, zusätzliche Strategien zu finden, um sie zu reduzieren.

- Schließlich wurde mit Version 1.2.0.0 eine bahnbrechende Änderung eingeführt, die sich auf früheren Code auswirkt, der das Bolt-Protokoll mit IAM-Authentifizierung verwendete. Ab Version 1.2.0.0 benötigt Bolt einen Ressourcenpfad für die IAM-Signatur. In Java könnte das Festlegen des Ressourcenpfads so aussehen: `request.setResourcePath("/openCypher"));`. In anderen Sprachen kann `/openCypher` an den Endpunkt-URI angehängt werden. Beispiele finden Sie unter Verwenden des Bolt-Protokolls.

Verbesserungen in dieser Engine-Version

- Verbesserte Leistung von Gremlin-`order-by`-Abfragen. Gremlin-Abfragen mit einem `order-by` am Ende eines `NeptuneGraphQueryStep` verwenden jetzt eine größere Chunk-Größe für bessere Leistung. Dies gilt nicht für `order-by` auf einem internen (Nicht-Wurzel-)Knoten des Abfrageplans.
- Verbesserte Leistung von Gremlin-Update-Abfragen. Scheitelpunkte und Kanten müssen jetzt beim Hinzufügen von Edges oder Eigenschaften gegen Löschen gesperrt werden. Durch diese Änderung werden doppelte Sperren innerhalb einer Transaktion beseitigt, wodurch die Leistung verbessert wird.
- Die Leistung von Gremlin-Abfragen, die `dedup()` innerhalb einer `repeat()` Unterabfrage verwendet werden, wurde verbessert, indem sie auf die native `dedup` Ausführungsebene verschoben wurden.
- Der Gremlin-`Neptune#cardinalityEstimates`-Abfragehinweis wurde hinzugefügt. Wenn auf `false` gesetzt, werden Kardinalitätsschätzungen deaktiviert.
- Es wurden benutzerfreundliche Fehlermeldungen für IAM-Authentifizierungsfehler hinzugefügt. Diese Nachrichten zeigen jetzt den ARN Ihres IAM-Benutzers oder Ihrer Rolle, den Ressourcen-

ARN und eine Liste der nicht autorisierten Aktionen für die Anfrage. Anhand der Liste der nicht autorisierten Aktionen können Sie erkennen, was in der von Ihnen verwendeten IAM-Richtlinie möglicherweise fehlt oder ausdrücklich verweigert wurde.

In diesem Engine-Version behobene Fehler

- Es wurde ein Gremlin-Korrekturfehler im Zusammenhang mit `WherePredicateStep`-Übersetzungen behoben, bei dem die Abfrage-Engine von Neptune falsche Ergebnisse für Abfragen mit `where(P.neq('x'))` und Varianten davon lieferte.
- Es wurde ein Gremlin-Fehler behoben, bei dem die Verwendung `PartitionStrategy` nach dem Upgrade auf TinkerPop 3.5 fälschlicherweise zu einem Fehler mit der Meldung „PartitionStrategy Funktioniert nicht mit anonymen Durchläufen“ führte, wodurch die Ausführung des Traversals verhindert wurde.
- Es wurden verschiedene Gremlin-Fehler im Zusammenhang mit einem `joinTime` eines finalen Join und mit Statistiken innerhalb von `Project.ASK`-Untergruppen behoben.
- Ein openCypher-Fehler in der `MERGE`-Klausel wurde behoben, der in einigen Fällen zur doppelten Erstellung von Knoten und Edges führte.
- Es wurde ein Transaktionsfehler behoben, durch den eine Sitzung Grafikdaten einfügen und festschreiben konnte, selbst wenn die entsprechenden gleichzeitigen Wörterbucheinfügungen rückgängig gemacht wurden.
- Es wurde ein Massenladerfehler behoben, der bei hohen Einfügebelatungen zu Leistungsregressionen führte.
- Es wurde ein SPARQL-Fehler bei der Behandlung von Abfragen behoben, die `(NOT) EXISTS` innerhalb einer `OPTIONAL`-Klausel enthalten waren, wodurch in einigen Fällen Abfrageergebnisse fehlten.
- Es wurde ein Fehler behoben, durch den Treiber scheinbar hängen blieben, wenn Anfragen aufgrund einer Zeitüberschreitung vor dem Start der Evaluierung storniert wurden. Es war möglich, in diesen Zustand zu gelangen, wenn alle Threads zur Abfrageverarbeitung auf dem Server verbraucht waren, während es bei Elementen in der Anforderungswarteschlange zu Zeitüberschreitungen kam. Da die Zeitüberschreitungen in der Anforderungswarteschlange nicht sofort auf das Senden von Nachrichten zurückzuführen waren, hatte der Client den Eindruck, dass die Antworten noch ausstehen würden.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.2.0.0.R2 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprache kompatibel ist:

- Die älteste unterstützte Version von Gremlin: `3.5.2`
- Die neueste unterstützte Version von Gremlin: `3.5.4`
- openCypher-Version: `Neptune-9.0.20190305-1.0`
- SPARQL-Version: `1.1`

Upgrade-Pfade zum Engine-Release 1.2.0.0.R2

Ihr Cluster wird während des nächsten Wartungsfensters automatisch auf diese Patch-Version aktualisiert, wenn Sie die Modulversion `1.2.0.0` ausführen.

Sie können nur `1.2.0.0` manuell vom neuesten Patch-Release des Engine-Release 1.1.1.0 auf Release aktualisieren. Frühere Engine-Versionen müssen zuerst auf die neueste Version von `1.1.1.0` aktualisiert werden, bevor sie auf `1.2.0.0` aktualisiert werden können.

Wenn Sie zuerst auf Version `1.1.1.0` und dann sofort auf Version `1.2.0.0` aktualisieren, tritt möglicherweise ein Fehler wie der folgende auf:

```
  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.
```

Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Upgrade auf diesen Release

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.2.0.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.2.0.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Statt `--apply-immediately` können Sie `--no-apply-immediately` angeben. Um ein Hauptversions-Upgrade durchzuführen, ist der Parameter erforderlich. allow-major-version-upgrade Stellen Sie außerdem sicher, dass Sie die Engine-Version angeben, da Ihre Engine sonst möglicherweise auf eine andere Version aktualisiert wird.

Wenn Ihr Cluster eine benutzerdefinierte Cluster-Parametergruppe verwendet, müssen Sie diesen Parameter einschließen, um ihn zu anzugeben:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Ebenso sollte für Instances im Cluster, die eine benutzerdefinierte DB-Parametergruppe verwenden, dieser Parameter eingeschlossen werden, um ihn zu spezifizieren:

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
> 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.
> ```
>
> Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine-Version 1.1.1.0 (19.04.2022)

Ab dem 19.04.2022 wird die Engine-Version 1.1.1.0 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

Important

Ein Upgrade von einer früheren Version auf diese Engine-Version löst **`1.1.0.0`** auch ein Betriebssystem-Upgrade auf allen Instances in Ihrem DB-Cluster aus. Da aktive Schreibanforderungen, die während des Betriebssystem-Upgrades auftreten, nicht verarbeitet werden, müssen Sie vor dem Start des Upgrades alle Schreib-Workloads für den Cluster, der aktualisiert wird, pausieren, einschließlich Massendatenladungen.
Um das Upgrade erfolgreich abzuschließen, muss für jedes Subnetz in jeder Availability Zone (AZ) mindestens eine IP-Adresse pro Neptune-Instance verfügbar sein. Wenn es beispielsweise eine Writer-Instance und zwei Reader-Instances in Subnetz 1 und zwei Reader-Instances in Subnetz 2 gibt, muss Subnetz 1 mindestens 3 freie IP-Adressen und Subnetz 2 mindestens 2 freie IP-Adressen haben, bevor das Upgrade gestartet wird.
Zu Beginn des Upgrades generiert Neptune einen Snapshot mit einem Namen, der sich aus `preupgrade` gefolgt von einer automatisch generierten Kennung zusammensetzt, die auf Ihren DB-Clusterinformationen basiert. Dieser Snapshot wird Ihnen nicht berechnet und Sie können ihn zur Wiederherstellung Ihres DB-Clusters verwenden, falls während des Upgrade-Vorgangs etwas schief geht.
Wenn das Engine-Upgrade selbst abgeschlossen ist, ist die neue Engine-Version kurzzeitig auf dem alten Betriebssystem verfügbar, aber in weniger als 5 Minuten beginnen alle Instances in Ihrem Cluster gleichzeitig mit einem Betriebssystem-Upgrade. Ihr DB-Cluster wird zu diesem Zeitpunkt für einige Minuten nicht verfügbar sein. Sie können die Schreib-Workloads fortsetzen, nachdem das Upgrade abgeschlossen ist.
Dieser Prozess generiert die folgenden Ereignisse:

- Ereignismeldungen pro Cluster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`

- Ereignismeldungen pro Instance:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Nachfolgende Patch-Veröffentlichungen für dieses Version

- Wartungs-Release: 1.1.1.0.R2 (16.05.2022)
- Release: 1.1.1.0.R3 (07.06.2022)
- Release: 1.1.1.0.R4 (23.06.2022)
- Release: 1.1.1.0.R5 (21.07.2022)
- Release: 1.1.1.0.R6 (23.09.2022)
- Release: 1.1.1.0.R7 (23.01.2023)

Neue Features in dieser Engine-Version

- Die openCypher-Abfragesprache ist jetzt allgemein für den produktiven Einsatz verfügbar.

> **Warning**
>
> In dieser Version gibt es eine signifikante Änderung für Code, der openCypher mit IAM-Authentifizierung verwendet. In der Neptune-Vorschau für openCypher enthielt der Host-String in der IAM-Signatur das Protokoll, zum Beispiel `bolt://`, wie folgt:
>
> ```
> "Host":"bolt://(host URL):(port)"
> ```
>
> Ab dieser Engine-Version muss das Protokoll weggelassen werden:
>
> ```
> "Host":"(host URL):(port)"
> ```
>
> Beispiele finden Sie unter Verwenden des Bolt-Protokolls.

- Unterstützung für hinzugefügt TinkerPop `3.5.2`. Zu den Änderungen in dieser Version gehören die Unterstützung für Remote-Transaktionen und Bytecode-Unterstützung für Sitzungen (Unter Verwendung von `g.tx`) sowie die Hinzufügung des `datetime()`-Features zur Gremlin-Sprache.

  > **Warning**
  >
  > In TinkerPop 3.5.0, 3.5.1 und 3.5.2 wurden mehrere wichtige Änderungen eingeführt, die sich auf Ihren Gremlin-Code auswirken können. Zum Beispiel funktioniert es nicht mehr, Traversals, die von a erzeugt wurden, GraphTraversalSource als Kinder wie folgt zu verwenden:. `g.V().union(identity(), g.V())`
  > Verwenden Sie jetzt stattdessen eine anonyme Traversal wie folgt: `g.V().union(identity(), __.V())`.

- Unterstützung für AWS globale Bedingungsschlüssel hinzugefügt, die Sie in IAM-Datenzugriffsrichtlinien verwenden können, die den Zugriff auf Daten steuern, die in Neptune, einem Neptune-DB-Cluster, gespeichert sind.
- Die Neptune DFE-Abfrage-Engine ist jetzt allgemein für den Produktionseinsatz mit der openCypher-Abfragesprache verfügbar, aber noch nicht für Gremlin- und SPARQL-Abfragen. Sie aktivieren sie jetzt mit ihrem eigenen neptune_dfe_query_engine-Instance-Parameter und nicht mit dem Lab-Modus-Parameter.

Verbesserungen in dieser Engine-Version

- openCypher wurde um neue Features erweitert, wie z. B. Unterstützung parametrisierter Abfragen, AST-Zwischenspeicherung (Abstract Syntax Tree) für parametrisierte Abfragen, Verbesserungen bei Pfaden mit variabler Länge (VLP) sowie neue Operatoren und Klauseln. Siehe Einhaltung der OpenCypher-Spezifikationen in Amazon Neptune für Informationen zum aktuellen Stand der Sprachunterstützung.
- openCypher wurde für einfache Lese- und Schreib-Workloads erheblich verbessert, was zu einem höheren Durchsatz im Vergleich zu Version 1.1.0.0 führte.
- Die bidirektionalen openCypher-Einschränkungen und die Tiefenbeschränkungen für Pfade variabler Länge wurden entfernt.
- Die Unterstützung für Gremlin `within`- und `without`-Prädikate in der DFE-Engine wurde abgeschlossen, einschließlich Fällen, in denen sie mit anderen Prädikatoperatoren kombiniert werden. Zum Beispiel:

```
g.V().has('age', within(12, 15, 18).or(gt(30)))
```

- Erweiterte Unterstützung in der DFE-Engine für den `order` Gremlin-Schritt, wenn der Gültigkeitsbereich global ist (d. h. nicht `order(local)`) und wenn keine `by()` Modulatoren verwendet werden. Zum Beispiel hätte diese Abfrage jetzt DFE-Unterstützung:

```
 g.V().values("age").order()
```

- Dem Antwortformat des Neptune-Streams-Änderungsprotokolls wurde ein `isLastOp`-Feld hinzugefügt, um anzuzeigen, dass ein Datensatz die letzte Operation in seiner Transaktion ist.
- Die Leistung der Audit-Protokollierung wurde deutlich verbessert und die Latenz reduziert, wenn die Audit-Protokollierung aktiviert ist.
- WebSocket Gremlin-Bytecode und HTTP-Abfragen wurden in Audit-Logs in ein vom Benutzer lesbares Format konvertiert. Abfragen können jetzt direkt aus den Audit-Protokollen kopiert werden, um sie in Neptune-Notebooks und anderswo auszuführen. Beachten Sie, dass diese Änderung des aktuellen Audit-Protokollformats eine bahnbrechende Änderung darstellt.

In diesem Engine-Version behobene Fehler

- Es wurde ein seltener Gremlin-Fehler behoben, bei dem keine Ergebnisse zurückgegeben wurden, wenn verschachtelte `filter()`- und `count()`-Schritte kombiniert wurden, wie in der folgenden Abfrage:

```
g.V("1").filter(out("knows")
        .filter(in("knows")
        .hasId("notExists"))
        .count())
```

- Es wurde ein Gremlin-Fehler behoben, bei dem ein Fehler zurückgegeben wurde, wenn ein Scheitelpunkt verwendet wurde, der in einem Aggregatschritt gespeichert wurde, `to()` oder `from()` bei Durchläufen in Verbindung mit einem `addE`-Schritt. Ein Beispiel für eine solche Abfrage:

```
g.V("id").aggregate("v").out().addE().to(select("v").unfold()))
```

- Es wurde ein Gremlin-Fehler behoben, bei dem der `not`-Schritt in Ausnahmefällen fehlschlug, wenn die DFE-Engine verwendet wurde. Zum Beispiel:

```
g.V().not(V())
```

- Es wurde ein Gremlin-Fehler behoben, bei dem `sideEffect`-Werte innerhalb von `to()`- und `from()`-Traversalen nicht verfügbar waren.
- Es wurde ein Fehler behoben, der gelegentlich dazu führte, dass ein schneller Reset einen Instance-Failover auslöste.
- Es wurde ein Massenlader-Fehler behoben, bei dem eine fehlgeschlagene Transaktion nicht geschlossen wurde, bevor der nächste Ladejob gestartet wurde.
- Es wurde ein Massenlader-Fehler behoben, bei dem ein unzureichender Speicherplatz zu einem Systemabsturz führen konnte.
- Es wurde ein erneuter Versuch hinzugefügt, um einen Massenladerfehler zu beheben, bei dem der Lader nicht lange genug wartete, bis die IAM-Anmeldeinformationen nach einem Failover verfügbar wurden.
- Es wurde ein Fehler behoben, bei dem der interne Cache für Anmeldeinformationen für Endpunkte, die keine Abfragen waren, wie z. B. den `status`-Endpunkt, nicht ordnungsgemäß gelöscht wurde.
- Es wurde ein Streams-Fehler behoben, um sicherzustellen, dass die Sequenznummern der Stream-Commits korrekt angeordnet sind.
- Es wurde ein Fehler behoben, bei dem lang andauernde Verbindungen auf IAM-fähigen Clustern nach weniger als zehn Tagen beendet wurden.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.1.1.0 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprache kompatibel ist:

- Die älteste unterstützte Version von Gremlin: `3.5.2`
- Die neueste unterstützte Version von Gremlin: `3.5.4`
- openCypher-Version: `Neptune-9.0.20190305-1.0`
- SPARQL-Version: `1.1`

Upgrade-Pfade zum Engine-Release 1.1.1.0

Sie können jeden vorherigen Neptune Engine-Release manuell auf diese Version aktualisieren. Beachten Sie, dass das Upgrade von Versionen vor der Hauptversions-Engine (1.1.0.0) auf diese Version länger dauern wird.

Es wird kein automatisches Upgrade auf diese Version durchgeführt.

Upgrade auf diesen Release

Important

Ein Upgrade von einer früheren Version auf **`1.1.0.0`** löst auch ein Betriebssystem-Upgrade auf allen Instances in Ihrem DB-Cluster aus. Da aktive Schreibanforderungen, die während des Betriebssystem-Upgrades auftreten, nicht verarbeitet werden, müssen Sie vor dem Start des Upgrades alle Schreib-Workloads für den Cluster, der aktualisiert wird, pausieren, einschließlich Massendatenladungen.
Zu Beginn des Upgrades generiert Neptune einen Snapshot mit einem Namen, der sich aus `preupgrade` gefolgt von einer automatisch generierten Kennung zusammensetzt, die auf Ihren DB-Clusterinformationen basiert. Dieser Snapshot wird Ihnen nicht berechnet und Sie können ihn zur Wiederherstellung Ihres DB-Clusters verwenden, falls während des Upgrade-Vorgangs etwas schief geht.
Wenn das Engine-Upgrade selbst abgeschlossen ist, ist die neue Engine-Version kurzzeitig auf dem alten Betriebssystem verfügbar, aber in weniger als 5 Minuten beginnen alle Instances in Ihrem Cluster gleichzeitig mit einem Betriebssystem-Upgrade. Ihr DB-Cluster wird zu diesem Zeitpunkt für etwa 6 Minuten nicht verfügbar sein. Sie können die Schreib-Workloads fortsetzen, nachdem das Upgrade abgeschlossen ist.
Dieser Prozess generiert die folgenden Ereignisse:

- Ereignismeldungen pro Cluster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- Ereignismeldungen pro Instance:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`

```
    • DB instance restarted
```

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder 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
```

Für 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
```

Statt `--apply-immediately` können Sie `--no-apply-immediately` angeben. Um ein Hauptversions-Upgrade durchzuführen, ist der Parameter erforderlich. allow-major-version-upgrade Stellen Sie außerdem sicher, dass Sie die Engine-Version angeben, da Ihre Engine sonst möglicherweise auf eine andere Version aktualisiert wird.

Wenn Ihr Cluster eine benutzerdefinierte Cluster-Parametergruppe verwendet, müssen Sie diesen Parameter einschließen, um ihn zu anzugeben:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Ebenso sollte für Instances im Cluster, die eine benutzerdefinierte DB-Parametergruppe verwenden, dieser Parameter eingeschlossen werden, um ihn zu spezifizieren:

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
> 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.
```

Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit ein vorheriges Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine-Version 1.1.1.0.R7 (23.01.2023)

Ab dem 23.01.2023 wird die Engine-Version 1.1.1.0.R7 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

Important

Ein Upgrade von einer früheren Version auf diese Engine-Version löst **`1.1.0.0`** auch ein Betriebssystem-Upgrade auf allen Instances in Ihrem DB-Cluster aus. Da aktive Schreibanforderungen, die während des Betriebssystem-Upgrades auftreten, nicht verarbeitet werden, müssen Sie vor dem Start des Upgrades alle Schreib-Workloads für den Cluster, der aktualisiert wird, pausieren, einschließlich Massendatenladungen.
Um das Upgrade erfolgreich abzuschließen, muss für jedes Subnetz in jeder Availability Zone (AZ) mindestens eine IP-Adresse pro Neptune-Instance verfügbar sein. Wenn es beispielsweise eine Writer-Instance und zwei Reader-Instances in Subnetz 1 und zwei Reader-Instances in Subnetz 2 gibt, muss Subnetz 1 mindestens 3 freie IP-Adressen und Subnetz 2 mindestens 2 freie IP-Adressen haben, bevor das Upgrade gestartet wird.
Zu Beginn des Upgrades generiert Neptune einen Snapshot mit einem Namen, der sich aus `preupgrade` gefolgt von einer automatisch generierten Kennung zusammensetzt, die auf Ihren DB-Clusterinformationen basiert. Dieser Snapshot wird Ihnen nicht berechnet und Sie können ihn zur Wiederherstellung Ihres DB-Clusters verwenden, falls während des Upgrade-Vorgangs etwas schief geht.
Wenn das Engine-Upgrade selbst abgeschlossen ist, ist die neue Engine-Version kurzzeitig auf dem alten Betriebssystem verfügbar, aber in weniger als 5 Minuten beginnen alle

Instances in Ihrem Cluster gleichzeitig mit einem Betriebssystem-Upgrade. Ihr DB-Cluster wird zu diesem Zeitpunkt für einige Minuten nicht verfügbar sein. Sie können die Schreib-Workloads fortsetzen, nachdem das Upgrade abgeschlossen ist.
Dieser Prozess generiert die folgenden Ereignisse:

- Ereignismeldungen pro Cluster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-(autogenerated snapshot ID)]`
  - `Database cluster major version has been upgraded`
- Ereignismeldungen pro Instance:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Verbesserungen in dieser Engine-Version

- Verbesserte Leistung von openCypher-Abfragen mit `MERGE` und `OPTIONAL MATCH`.
- Verbesserte Leistung von openCypher-Abfragen, die eine Liste `UNWIND` von Zuordnungen mit Literalwerten beinhalten.
- Verbesserte Leistung von openCypher-Abfragen, die einen `IN`-Filter für `id` haben. Zum Beispiel:

```
MATCH (n) WHERE id(n) IN ['1', '2', '3'] RETURN n
```

- Leistungsverbesserungen und Korrekturkorrekturen für verschiedene Gremlin-Operatoren, darunter `repeat`, `coalesce`, `store` und `aggregate`.

In diesem Engine-Version behobene Fehler

- Es wurde ein openCypher-Fehler behoben, bei dem eine Anfrage, die HTTP-Keep-Alive verwendete, fälschlicherweise geschlossen werden konnte, wenn sie nach einer fehlgeschlagenen Anfrage eingereicht wurde.
- Es wurde ein openCypher-Fehler behoben, bei dem der Parametertyp für eine Liste oder eine Liste von Maps nicht immer korrekt interpretiert wurde.

- Es wurde ein openCypher-Fehler behoben, bei dem Abfragen die Zeichenfolge, `"null"`, anstelle eines Nullwerts in Bolt und SPARQL-JSON zurückgaben.
- OpenCypher-Fehlercodes und Fehlermeldungen für Abfrage-Timeout-Ausfälle und Fehler wurden behoben. out-of-memory
- Es wurde ein Gremlin-Fehler behoben, der dazu führte, dass `valueMap()` bei einem `by()`-Traversal in der DFE-Engine nicht optimiert wurde.
- Es wurde ein Problem mit der Deadlock-Detektorlogik behoben, das gelegentlich dazu führte, dass die Engine nicht reagierte.
- Es wurde ein Fehler im Auditprotokoll behoben, der dazu führte, dass unnötige Informationen protokolliert wurden und bestimmte Felder in den Protokollen fehlten.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.1.1.0.R7 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprache kompatibel ist:

- Die älteste unterstützte Version von Gremlin: `3.5.2`
- Die neueste unterstützte Version von Gremlin: `3.5.3`
- openCypher-Version: `Neptune-9.0.20190305-1.0`
- SPARQL-Version: `1.1`

Upgrade-Pfade zum Engine-Release 1.1.1.0.R7

Ihr Cluster wird während des nächsten Wartungsfensters automatisch auf diese Patch-Version aktualisiert, wenn Sie die Modulversion `1.1.1.0` ausführen.

Upgrade auf diesen Release

> **Important**
>
> Ein Upgrade von einer früheren Version auf **`1.1.0.0`** löst auch ein Betriebssystem-Upgrade auf allen Instances in Ihrem DB-Cluster aus. Da aktive Schreibanforderungen, die während des Betriebssystem-Upgrades auftreten, nicht verarbeitet werden, müssen Sie vor dem Start des Upgrades alle Schreib-Workloads für den Cluster, der aktualisiert wird, pausieren, einschließlich Massendatenladungen.

Zu Beginn des Upgrades generiert Neptune einen Snapshot mit einem Namen, der sich aus `preupgrade` gefolgt von einer automatisch generierten Kennung zusammensetzt, die auf Ihren DB-Clusterinformationen basiert. Dieser Snapshot wird Ihnen nicht berechnet und Sie können ihn zur Wiederherstellung Ihres DB-Clusters verwenden, falls während des Upgrade-Vorgangs etwas schief geht.
Wenn das Engine-Upgrade selbst abgeschlossen ist, ist die neue Engine-Version kurzzeitig auf dem alten Betriebssystem verfügbar, aber in weniger als 5 Minuten beginnen alle Instances in Ihrem Cluster gleichzeitig mit einem Betriebssystem-Upgrade. Ihr DB-Cluster wird zu diesem Zeitpunkt für etwa 6 Minuten nicht verfügbar sein. Sie können die Schreib-Workloads fortsetzen, nachdem das Upgrade abgeschlossen ist.
Dieser Prozess generiert die folgenden Ereignisse:

- Ereignismeldungen pro Cluster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- Ereignismeldungen pro Instance:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.1.1.0 \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.1.1.0 ^
    --apply-immediately
```

Updates werden auf alle Instances in einem DB-Cluster gleichzeitig angewendet. Ein Update erfordert einen Datenbankneustart auf diesen Instances. Daher kommt es zu einer Ausfallzeit von 20–30 Sekunden bis zu mehreren Minuten. Anschließend können Sie die Nutzung Ihres DB-Clusters fortsetzen.

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune

einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
> 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.
> ```
>
> Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine-Version 1.1.1.0.R6 (23.09.2022)

Ab dem 23.09.2022 wird die Engine-Version 1.1.1.0.R6 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

> **Important**
>
> Ein Upgrade von einer früheren Version auf diese Engine-Version löst **`1.1.0.0`** auch ein Betriebssystem-Upgrade auf allen Instances in Ihrem DB-Cluster aus. Da aktive Schreibanforderungen, die während des Betriebssystem-Upgrades auftreten, nicht verarbeitet werden, müssen Sie vor dem Start des Upgrades alle Schreib-Workloads für den Cluster, der aktualisiert wird, pausieren, einschließlich Massendatenladungen.
> Um das Upgrade erfolgreich abzuschließen, muss für jedes Subnetz in jeder Availability Zone (AZ) mindestens eine IP-Adresse pro Neptune-Instance verfügbar sein. Wenn es beispielsweise eine Writer-Instance und zwei Reader-Instances in Subnetz 1 und zwei

Reader-Instances in Subnetz 2 gibt, muss Subnetz 1 mindestens 3 freie IP-Adressen und Subnetz 2 mindestens 2 freie IP-Adressen haben, bevor das Upgrade gestartet wird.
Zu Beginn des Upgrades generiert Neptune einen Snapshot mit einem Namen, der sich aus `preupgrade` gefolgt von einer automatisch generierten Kennung zusammensetzt, die auf Ihren DB-Clusterinformationen basiert. Dieser Snapshot wird Ihnen nicht berechnet und Sie können ihn zur Wiederherstellung Ihres DB-Clusters verwenden, falls während des Upgrade-Vorgangs etwas schief geht.
Wenn das Engine-Upgrade selbst abgeschlossen ist, ist die neue Engine-Version kurzzeitig auf dem alten Betriebssystem verfügbar, aber in weniger als 5 Minuten beginnen alle Instances in Ihrem Cluster gleichzeitig mit einem Betriebssystem-Upgrade. Ihr DB-Cluster wird zu diesem Zeitpunkt für einige Minuten nicht verfügbar sein. Sie können die Schreib-Workloads fortsetzen, nachdem das Upgrade abgeschlossen ist.
Dieser Prozess generiert die folgenden Ereignisse:

- Ereignismeldungen pro Cluster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- Ereignismeldungen pro Instance:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Note

In dieser Version gibt es eine signifikante Änderung für Code, der openCypher mit IAM-Authentifizierung verwendet. Bisher enthielt die Hostzeichenfolge in der IAM-Signatur das Protokoll, zum Beispiel `bolt://`, in etwa wie folgt:

```
"Host":"bolt://(host URL):(port)"
```

Ab Engine-Release `1.1.1.0` muss das Protokoll weggelassen werden:

```
"Host":"(host URL):(port)"
```

Beispiele finden Sie unter Verwenden des Bolt-Protokolls.

Verbesserungen in dieser Engine-Version

- Verbesserte Leistung von Gremlin-`order-by`-Abfragen. Gremlin-Abfragen mit einem `order-by` am Ende eines `NeptuneGraphQueryStep` verwenden jetzt eine größere Chunk-Größe für bessere Leistung. Dies gilt nicht für `order-by` auf einem internen (Nicht-Wurzel-)Knoten des Abfrageplans.
- Verbesserte Leistung von Gremlin-Update-Abfragen. Scheitelpunkte und Kanten müssen jetzt beim Hinzufügen von Edges oder Eigenschaften gegen Löschen gesperrt werden. Durch diese Änderung werden doppelte Sperren innerhalb einer Transaktion beseitigt, wodurch die Leistung verbessert wird.

In diesem Engine-Version behobene Fehler

- Ein openCypher-Fehler in der `MERGE`-Klausel wurde behoben, der in einigen Fällen zur doppelten Erstellung von Knoten und Edges führte.
- Es wurde ein Fehler bei der Behandlung von SPARQL-Abfragen behoben, die `(NOT) EXISTS` innerhalb einer `OPTIONAL`-Klausel enthalten waren, wodurch in einigen Fällen Abfrageergebnisse fehlten.
- Es wurde ein Fehler behoben, der den Serverneustart verzögerte, wenn ein Massenladevorgang im Gange war.
- Es wurde ein Fehler behoben, bei dem eine bidirektionale openCypher-Musterdurchquerung mit variabler Länge und einem Filter für die Beziehungseigenschaft zu einem Fehler führen würde. Ein Beispiel für ein solches Muster mit variabler Länge ist `(n)-[r*1..2]->(m)`.
- Ein Fehler im Zusammenhang mit der Art und Weise, wie zwischengespeicherte Daten an den Client zurückgesendet wurden, wurde behoben, der in einigen Fällen zu einer unerwartet langen Latenz führte.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.1.1.0.R6 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprache kompatibel ist:

- Die älteste unterstützte Version von Gremlin: `3.5.2`
- Die neueste unterstützte Version von Gremlin: `3.5.4`
- openCypher-Version: `Neptune-9.0.20190305-1.0`
- SPARQL-Version: `1.1`

Upgrade-Pfade zum Engine-Release 1.1.1.0.R6

Ihr Cluster wird während des nächsten Wartungsfensters automatisch auf diese Patch-Version aktualisiert, wenn Sie die Modulversion `1.1.1.0` ausführen.

Upgrade auf diesen Release

Important

Ein Upgrade von einer früheren Version auf **`1.1.0.0`** löst auch ein Betriebssystem-Upgrade auf allen Instances in Ihrem DB-Cluster aus. Da aktive Schreibanforderungen, die während des Betriebssystem-Upgrades auftreten, nicht verarbeitet werden, müssen Sie vor dem Start des Upgrades alle Schreib-Workloads für den Cluster, der aktualisiert wird, pausieren, einschließlich Massendatenladungen.

Zu Beginn des Upgrades generiert Neptune einen Snapshot mit einem Namen, der sich aus `preupgrade` gefolgt von einer automatisch generierten Kennung zusammensetzt, die auf Ihren DB-Clusterinformationen basiert. Dieser Snapshot wird Ihnen nicht berechnet und Sie können ihn zur Wiederherstellung Ihres DB-Clusters verwenden, falls während des Upgrade-Vorgangs etwas schief geht.

Wenn das Engine-Upgrade selbst abgeschlossen ist, ist die neue Engine-Version kurzzeitig auf dem alten Betriebssystem verfügbar, aber in weniger als 5 Minuten beginnen alle Instances in Ihrem Cluster gleichzeitig mit einem Betriebssystem-Upgrade. Ihr DB-Cluster wird zu diesem Zeitpunkt für etwa 6 Minuten nicht verfügbar sein. Sie können die Schreib-Workloads fortsetzen, nachdem das Upgrade abgeschlossen ist.

Dieser Prozess generiert die folgenden Ereignisse:

- Ereignismeldungen pro Cluster:

    - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-(autogenerated snapshot ID)]`
    - `Database cluster major version has been upgraded`
  - Ereignismeldungen pro Instance:
    - `Applying off-line patches to DB instance`
    - `DB instance shutdown`
    - `Finished applying off-line patches to DB instance`
    - `DB instance restarted`

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.1.1.0 \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.1.1.0 ^
    --apply-immediately
```

Updates werden auf alle Instances in einem DB-Cluster gleichzeitig angewendet. Ein Update erfordert einen Datenbankneustart auf diesen Instances. Daher kommt es zu einer Ausfallzeit von 20–30 Sekunden bis zu mehreren Minuten. Anschließend können Sie die Nutzung Ihres DB-Clusters fortsetzen.

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein

Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
> 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.
> ```

Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine-Version 1.1.1.0.R5 (21.07.2022)

Ab dem 21.07.2022 wird die Engine-Version 1.1.1.0.R5 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

Important

Ein Upgrade von einer früheren Version auf diese Engine-Version löst **`1.1.0.0`** auch ein Betriebssystem-Upgrade auf allen Instances in Ihrem DB-Cluster aus. Da aktive Schreibanforderungen, die während des Betriebssystem-Upgrades auftreten, nicht verarbeitet werden, müssen Sie vor dem Start des Upgrades alle Schreib-Workloads für den Cluster, der aktualisiert wird, pausieren, einschließlich Massendatenladungen.
Um das Upgrade erfolgreich abzuschließen, muss für jedes Subnetz in jeder Availability Zone (AZ) mindestens eine IP-Adresse pro Neptune-Instance verfügbar sein. Wenn es beispielsweise eine Writer-Instance und zwei Reader-Instances in Subnetz 1 und zwei Reader-Instances in Subnetz 2 gibt, muss Subnetz 1 mindestens 3 freie IP-Adressen und Subnetz 2 mindestens 2 freie IP-Adressen haben, bevor das Upgrade gestartet wird.
Zu Beginn des Upgrades generiert Neptune einen Snapshot mit einem Namen, der sich aus `preupgrade` gefolgt von einer automatisch generierten Kennung zusammensetzt, die auf Ihren DB-Clusterinformationen basiert. Dieser Snapshot wird Ihnen nicht berechnet und Sie können ihn zur Wiederherstellung Ihres DB-Clusters verwenden, falls während des Upgrade-Vorgangs etwas schief geht.
Wenn das Engine-Upgrade selbst abgeschlossen ist, ist die neue Engine-Version kurzzeitig auf dem alten Betriebssystem verfügbar, aber in weniger als 5 Minuten beginnen alle Instances in Ihrem Cluster gleichzeitig mit einem Betriebssystem-Upgrade. Ihr DB-Cluster wird zu diesem Zeitpunkt für einige Minuten nicht verfügbar sein. Sie können die Schreib-Workloads fortsetzen, nachdem das Upgrade abgeschlossen ist.
Dieser Prozess generiert die folgenden Ereignisse:

- Ereignismeldungen pro Cluster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-(autogenerated snapshot ID)]`
  - `Database cluster major version has been upgraded`
- Ereignismeldungen pro Instance:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Note

In dieser Version gibt es eine signifikante Änderung für Code, der openCypher mit IAM-Authentifizierung verwendet. Bisher enthielt die Hostzeichenfolge in der IAM-Signatur das Protokoll, zum Beispiel `bolt://`, in etwa wie folgt:

```
"Host":"bolt://(host URL):(port)"
```

Ab Engine-Release `1.1.1.0` muss das Protokoll weggelassen werden:

```
"Host":"(host URL):(port)"
```

Beispiele finden Sie unter Verwenden des Bolt-Protokolls.

Verbesserungen in dieser Engine-Version

- Es wurden Verbesserungen zur Unterstützung der Deadlock-Erkennung vorgenommen.

In diesem Engine-Version behobene Fehler

- Es wurde ein Fehler behoben, der ein fehlerfreies Herunterfahren von DB-Clustern unter bestimmten Bedingungen verhinderte.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.1.1.0.R5 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprache kompatibel ist:

- Die älteste unterstützte Version von Gremlin: `3.5.2`
- Die neueste unterstützte Version von Gremlin: `3.5.4`
- openCypher-Version: `Neptune-9.0.20190305-1.0`
- SPARQL-Version: `1.1`

Upgrade-Pfade zum Engine-Release 1.1.1.0.R5

Ihr Cluster wird während des nächsten Wartungsfensters automatisch auf diese Patch-Version aktualisiert, wenn Sie die Modulversion `1.1.1.0` ausführen.

Upgrade auf diesen Release

Important

Ein Upgrade von einer früheren Version auf **`1.1.0.0`** löst auch ein Betriebssystem-Upgrade auf allen Instances in Ihrem DB-Cluster aus. Da aktive Schreibanforderungen, die während des Betriebssystem-Upgrades auftreten, nicht verarbeitet werden, müssen Sie vor dem Start des Upgrades alle Schreib-Workloads für den Cluster, der aktualisiert wird, pausieren, einschließlich Massendatenladungen.

Zu Beginn des Upgrades generiert Neptune einen Snapshot mit einem Namen, der sich aus `preupgrade` gefolgt von einer automatisch generierten Kennung zusammensetzt, die auf Ihren DB-Clusterinformationen basiert. Dieser Snapshot wird Ihnen nicht berechnet und Sie können ihn zur Wiederherstellung Ihres DB-Clusters verwenden, falls während des Upgrade-Vorgangs etwas schief geht.

Wenn das Engine-Upgrade selbst abgeschlossen ist, ist die neue Engine-Version kurzzeitig auf dem alten Betriebssystem verfügbar, aber in weniger als 5 Minuten beginnen alle Instances in Ihrem Cluster gleichzeitig mit einem Betriebssystem-Upgrade. Ihr DB-Cluster wird zu diesem Zeitpunkt für etwa 6 Minuten nicht verfügbar sein. Sie können die Schreib-Workloads fortsetzen, nachdem das Upgrade abgeschlossen ist.

Dieser Prozess generiert die folgenden Ereignisse:

- Ereignismeldungen pro Cluster:

    - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-(autogenerated snapshot ID)]`
    - `Database cluster major version has been upgraded`
  - Ereignismeldungen pro Instance:
    - `Applying off-line patches to DB instance`
    - `DB instance shutdown`
    - `Finished applying off-line patches to DB instance`
    - `DB instance restarted`

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.1.1.0 \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.1.1.0 ^
    --apply-immediately
```

Updates werden auf alle Instances in einem DB-Cluster gleichzeitig angewendet. Ein Update erfordert einen Datenbankneustart auf diesen Instances. Daher kommt es zu einer Ausfallzeit von 20–30 Sekunden bis zu mehreren Minuten. Anschließend können Sie die Nutzung Ihres DB-Clusters fortsetzen.

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein

Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
> 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.
> ```

Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine-Version 1.1.1.0.R4 (23.06.2022)

Ab dem 23.06.2022 wird die Engine-Version 1.1.1.0.R4 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

Important

Ein Upgrade von einer früheren Version auf diese Engine-Version löst **`1.1.0.0`** auch ein Betriebssystem-Upgrade auf allen Instances in Ihrem DB-Cluster aus. Da aktive Schreibanforderungen, die während des Betriebssystem-Upgrades auftreten, nicht verarbeitet werden, müssen Sie vor dem Start des Upgrades alle Schreib-Workloads für den Cluster, der aktualisiert wird, pausieren, einschließlich Massendatenladungen.
Um das Upgrade erfolgreich abzuschließen, muss für jedes Subnetz in jeder Availability Zone (AZ) mindestens eine IP-Adresse pro Neptune-Instance verfügbar sein. Wenn es beispielsweise eine Writer-Instance und zwei Reader-Instances in Subnetz 1 und zwei Reader-Instances in Subnetz 2 gibt, muss Subnetz 1 mindestens 3 freie IP-Adressen und Subnetz 2 mindestens 2 freie IP-Adressen haben, bevor das Upgrade gestartet wird.
Zu Beginn des Upgrades generiert Neptune einen Snapshot mit einem Namen, der sich aus `preupgrade` gefolgt von einer automatisch generierten Kennung zusammensetzt, die auf Ihren DB-Clusterinformationen basiert. Dieser Snapshot wird Ihnen nicht berechnet und Sie können ihn zur Wiederherstellung Ihres DB-Clusters verwenden, falls während des Upgrade-Vorgangs etwas schief geht.
Wenn das Engine-Upgrade selbst abgeschlossen ist, ist die neue Engine-Version kurzzeitig auf dem alten Betriebssystem verfügbar, aber in weniger als 5 Minuten beginnen alle Instances in Ihrem Cluster gleichzeitig mit einem Betriebssystem-Upgrade. Ihr DB-Cluster wird zu diesem Zeitpunkt für einige Minuten nicht verfügbar sein. Sie können die Schreib-Workloads fortsetzen, nachdem das Upgrade abgeschlossen ist.
Dieser Prozess generiert die folgenden Ereignisse:

- Ereignismeldungen pro Cluster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-(autogenerated snapshot ID)]`
  - `Database cluster major version has been upgraded`
- Ereignismeldungen pro Instance:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Note

In dieser Version gibt es eine signifikante Änderung für Code, der openCypher mit IAM-Authentifizierung verwendet. Bisher enthielt die Hostzeichenfolge in der IAM-Signatur das Protokoll, zum Beispiel `bolt://`, in etwa wie folgt:

```
"Host":"bolt://(host URL):(port)"
```

Ab Engine-Release `1.1.1.0` muss das Protokoll weggelassen werden:

```
"Host":"(host URL):(port)"
```

Beispiele finden Sie unter Verwenden des Bolt-Protokolls.

Verbesserungen in dieser Engine-Version

- Die Instanzkonfiguration für `x2g` Instance-Typen wurde aktualisiert.
- Die Leistung von Scheitelpunkt-Drops wurde verbessert.

In diesem Engine-Version behobene Fehler

- Es wurde ein Gremlin-Fehler behoben, bei dem Lösungen für bestimmte Arten von ASK-Joins keine stabile Reihenfolge für eine Abfrage beibehielten, die mehrfach oder über mehrere Leser aufgerufen wurde.
- Außerdem wurde der Umfang einer Änderung in der vorherigen Version eingeschränkt, die zu Leistungseinbußen bei bestimmten Arten von ASK-Joins in Gremlin führte.
- Es wurde ein Gremlin-Fehler im `union()`-Schritt behoben, der auftrat, wenn eine Edge-Eingabe und eine Traversake zu einem Scheitelpunkt innerhalb untergeordneter Traversalen stattfanden.
- Es wurde ein Fehler im Gremlin-Profil behoben, bei dem einige Schritte als nicht optimiert gemeldet wurden, obwohl sie es tatsächlich waren.
- Es wurde ein SPARQL-Fehler behoben, bei dem Variablen, die in `FILTER` Ausdrücken verwendet wurden, die in `UNION` Klauseln verschachtelt waren, ungültige Bereichsinformationen zugewiesen wurden.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.1.1.0.R4 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprache kompatibel ist:

- Die älteste unterstützte Version von Gremlin: `3.5.2`
- Die neueste unterstützte Version von Gremlin: `3.5.4`
- openCypher-Version: `Neptune-9.0.20190305-1.0`
- SPARQL-Version: `1.1`

Upgrade-Pfade zum Engine-Release 1.1.1.0.R4

Ihr Cluster wird während des nächsten Wartungsfensters automatisch auf diese Patch-Version aktualisiert, wenn Sie die Modulversion `1.1.1.0` ausführen.

Upgrade auf diesen Release

> **Important**
>
> Ein Upgrade von einer früheren Version auf **`1.1.0.0`** löst auch ein Betriebssystem-Upgrade auf allen Instances in Ihrem DB-Cluster aus. Da aktive Schreibanforderungen, die während

des Betriebssystem-Upgrades auftreten, nicht verarbeitet werden, müssen Sie vor dem Start des Upgrades alle Schreib-Workloads für den Cluster, der aktualisiert wird, pausieren, einschließlich Massendatenladungen.
Zu Beginn des Upgrades generiert Neptune einen Snapshot mit einem Namen, der sich aus `preupgrade` gefolgt von einer automatisch generierten Kennung zusammensetzt, die auf Ihren DB-Clusterinformationen basiert. Dieser Snapshot wird Ihnen nicht berechnet und Sie können ihn zur Wiederherstellung Ihres DB-Clusters verwenden, falls während des Upgrade-Vorgangs etwas schief geht.
Wenn das Engine-Upgrade selbst abgeschlossen ist, ist die neue Engine-Version kurzzeitig auf dem alten Betriebssystem verfügbar, aber in weniger als 5 Minuten beginnen alle Instances in Ihrem Cluster gleichzeitig mit einem Betriebssystem-Upgrade. Ihr DB-Cluster wird zu diesem Zeitpunkt für etwa 6 Minuten nicht verfügbar sein. Sie können die Schreib-Workloads fortsetzen, nachdem das Upgrade abgeschlossen ist.
Dieser Prozess generiert die folgenden Ereignisse:

- Ereignismeldungen pro Cluster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- Ereignismeldungen pro Instance:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.1.1.0 \
```

```
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.1.1.0 ^
    --apply-immediately
```

Updates werden auf alle Instances in einem DB-Cluster gleichzeitig angewendet. Ein Update erfordert einen Datenbankneustart auf diesen Instances. Daher kommt es zu einer Ausfallzeit von 20–30 Sekunden bis zu mehreren Minuten. Anschließend können Sie die Nutzung Ihres DB-Clusters fortsetzen.

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
> 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.
> ```
>
> Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine-Version 1.1.1.0.R3 (07.06.2022)

Ab dem 07.06.2022 wird die Engine-Version 1.1.1.0.R3 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

> **Important**
>
> Ein Upgrade von einer früheren Version auf diese Engine-Version löst **`1.1.0.0`** auch ein Betriebssystem-Upgrade auf allen Instances in Ihrem DB-Cluster aus. Da aktive Schreibanforderungen, die während des Betriebssystem-Upgrades auftreten, nicht verarbeitet werden, müssen Sie vor dem Start des Upgrades alle Schreib-Workloads für den Cluster, der aktualisiert wird, pausieren, einschließlich Massendatenladungen.

Zu Beginn des Upgrades generiert Neptune einen Snapshot mit einem Namen, der sich aus `preupgrade` gefolgt von einer automatisch generierten Kennung zusammensetzt, die auf Ihren DB-Clusterinformationen basiert. Dieser Snapshot wird Ihnen nicht berechnet und Sie können ihn zur Wiederherstellung Ihres DB-Clusters verwenden, falls während des Upgrade-Vorgangs etwas schief geht.
Wenn das Engine-Upgrade selbst abgeschlossen ist, ist die neue Engine-Version kurzzeitig auf dem alten Betriebssystem verfügbar, aber in weniger als 5 Minuten beginnen alle Instances in Ihrem Cluster gleichzeitig mit einem Betriebssystem-Upgrade. Ihr DB-Cluster wird zu diesem Zeitpunkt für einige Minuten nicht verfügbar sein. Sie können die Schreib-Workloads fortsetzen, nachdem das Upgrade abgeschlossen ist.
Dieser Prozess generiert die folgenden Ereignisse:

- Ereignismeldungen pro Cluster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- Ereignismeldungen pro Instance:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Note

In dieser Version gibt es eine signifikante Änderung für Code, der openCypher mit IAM-Authentifizierung verwendet. Bisher enthielt die Hostzeichenfolge in der IAM-Signatur das Protokoll, zum Beispiel `bolt://`, in etwa wie folgt:

```
"Host":"bolt://(host URL):(port)"
```

Ab Engine-Release `1.1.1.0` muss das Protokoll weggelassen werden:

```
"Host":"(host URL):(port)"
```

Beispiele finden Sie unter Verwenden des Bolt-Protokolls.

Verbesserungen in dieser Engine-Version

- Unterstützung für Graviton2-basierte `x2g` Instance-Typen hinzugefügt, die für speicherintensive Workloads optimiert sind. Diese sind zunächst nur in vier Varianten erhältlich AWS-Regionen:
  - USA Ost (Nord-Virginia) (`us-east-1`)
  - USA Ost (Ohio) (`us-east-2`)
  - USA West (Oregon) (`us-west-2`)
  - Europa (Irland) (`eu-west-1`)

  Weitere Informationen finden Sie in der Neptune-Preisseite.
- Verbesserte Leistung von Gremlin-Schritten, bei denen mehrere Edge- oder Scheitelpunkttraversalen, Eigenschaftssuche oder Label-Suche erforderlich sind.

In diesem Engine-Version behobene Fehler

- Es wurde ein Gremlin-Fehler bei der Verarbeitung des `otherV()`-Schrittes innerhalb einer untergeordneten Traversalen behoben.
- Es wurde ein Gremlin-Fehler in Abfragen mit `union` behoben, bei denen nur Filterschritte als untergeordnete Elemente verwendet wurden. Zum Beispiel:

```
g.V().union(has("name"), out("knows")).out()
```

- Es wurde ein SPARQL-Fehler behoben, bei dem Variablen, die in `FILTER` Ausdrücken verwendet wurden, die in `UNION` Klauseln verschachtelt waren, ungültige Bereichsinformationen zugewiesen wurden.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.1.1.0.R3 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprache kompatibel ist:

- Die älteste unterstützte Version von Gremlin: `3.5.2`
- Die neueste unterstützte Version von Gremlin: `3.5.4`

- openCypher-Version: `Neptune-9.0.20190305-1.0`
- SPARQL-Version: `1.1`

Upgrade-Pfade zum Engine-Release 1.1.1.0.R3

Ihr Cluster wird während des nächsten Wartungsfensters automatisch auf diese Patch-Version aktualisiert, wenn Sie die Modulversion `1.1.1.0` ausführen.

Upgrade auf diesen Release

Important

Ein Upgrade von einer früheren Version auf **`1.1.0.0`** löst auch ein Betriebssystem-Upgrade auf allen Instances in Ihrem DB-Cluster aus. Da aktive Schreibanforderungen, die während des Betriebssystem-Upgrades auftreten, nicht verarbeitet werden, müssen Sie vor dem Start des Upgrades alle Schreib-Workloads für den Cluster, der aktualisiert wird, pausieren, einschließlich Massendatenladungen.

Zu Beginn des Upgrades generiert Neptune einen Snapshot mit einem Namen, der sich aus `preupgrade` gefolgt von einer automatisch generierten Kennung zusammensetzt, die auf Ihren DB-Clusterinformationen basiert. Dieser Snapshot wird Ihnen nicht berechnet und Sie können ihn zur Wiederherstellung Ihres DB-Clusters verwenden, falls während des Upgrade-Vorgangs etwas schief geht.

Wenn das Engine-Upgrade selbst abgeschlossen ist, ist die neue Engine-Version kurzzeitig auf dem alten Betriebssystem verfügbar, aber in weniger als 5 Minuten beginnen alle Instances in Ihrem Cluster gleichzeitig mit einem Betriebssystem-Upgrade. Ihr DB-Cluster wird zu diesem Zeitpunkt für etwa 6 Minuten nicht verfügbar sein. Sie können die Schreib-Workloads fortsetzen, nachdem das Upgrade abgeschlossen ist.

Dieser Prozess generiert die folgenden Ereignisse:

- Ereignismeldungen pro Cluster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- Ereignismeldungen pro Instance:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`

```
    • Finished applying off-line patches to DB instance
    • DB instance restarted
```

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.1.1.0 \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.1.1.0 ^
    --apply-immediately
```

Updates werden auf alle Instances in einem DB-Cluster gleichzeitig angewendet. Ein Update erfordert einen Datenbankneustart auf diesen Instances. Daher kommt es zu einer Ausfallzeit von 20–30 Sekunden bis zu mehreren Minuten. Anschließend können Sie die Nutzung Ihres DB-Clusters fortsetzen.

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
> 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.
> ```
>
> Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Neptune-Wartungsrelease, Version 1.1.1.0.R2 (16.05.2022)

Ab dem 16.05.2022 wird die Engine-Version 1.1.1.0.R2 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

Important

Ein Upgrade von einer früheren Version auf diese Engine-Version löst **`1.1.0.0`** auch ein Betriebssystem-Upgrade auf allen Instances in Ihrem DB-Cluster aus. Da aktive Schreibanforderungen, die während des Betriebssystem-Upgrades auftreten, nicht verarbeitet werden, müssen Sie vor dem Start des Upgrades alle Schreib-Workloads für den Cluster, der aktualisiert wird, pausieren, einschließlich Massendatenladungen.
Um das Upgrade erfolgreich abzuschließen, muss für jedes Subnetz in jeder Availability Zone (AZ) mindestens eine IP-Adresse pro Neptune-Instance verfügbar sein. Wenn es beispielsweise eine Writer-Instance und zwei Reader-Instances in Subnetz 1 und zwei Reader-Instances in Subnetz 2 gibt, muss Subnetz 1 mindestens 3 freie IP-Adressen und Subnetz 2 mindestens 2 freie IP-Adressen haben, bevor das Upgrade gestartet wird.
Zu Beginn des Upgrades generiert Neptune einen Snapshot mit einem Namen, der sich aus `preupgrade` gefolgt von einer automatisch generierten Kennung zusammensetzt, die auf Ihren DB-Clusterinformationen basiert. Dieser Snapshot wird Ihnen nicht berechnet und Sie können ihn zur Wiederherstellung Ihres DB-Clusters verwenden, falls während des Upgrade-Vorgangs etwas schief geht.
Wenn das Engine-Upgrade selbst abgeschlossen ist, ist die neue Engine-Version kurzzeitig auf dem alten Betriebssystem verfügbar, aber in weniger als 5 Minuten beginnen alle Instances in Ihrem Cluster gleichzeitig mit einem Betriebssystem-Upgrade. Ihr DB-Cluster wird zu diesem Zeitpunkt für einige Minuten nicht verfügbar sein. Sie können die Schreib-Workloads fortsetzen, nachdem das Upgrade abgeschlossen ist.
Dieser Prozess generiert die folgenden Ereignisse:

- Ereignismeldungen pro Cluster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`

    - `Database cluster major version has been upgraded`
  - Ereignismeldungen pro Instance:
    - `Applying off-line patches to DB instance`
    - `DB instance shutdown`
    - `Finished applying off-line patches to DB instance`
    - `DB instance restarted`

> **Note**
>
> In dieser Version gibt es eine signifikante Änderung für Code, der openCypher mit IAM-Authentifizierung verwendet. Bisher enthielt die Hostzeichenfolge in der IAM-Signatur das Protokoll, zum Beispiel `bolt://`, in etwa wie folgt:
>
> ```
> "Host":"bolt://(host URL):(port)"
> ```
>
> Ab Engine-Release `1.1.1.0` muss das Protokoll weggelassen werden:
>
> ```
> "Host":"(host URL):(port)"
> ```
>
> Beispiele finden Sie unter Verwenden des Bolt-Protokolls.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.1.1.0.R2 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprache kompatibel ist:

- Die älteste unterstützte Version von Gremlin: `3.5.2`
- Die neueste unterstützte Version von Gremlin: `3.5.4`
- openCypher-Version: `Neptune-9.0.20190305-1.0`
- SPARQL-Version: `1.1`

Upgrade-Pfade zum Engine-Release 1.1.1.0.R2

Ihr Cluster wird während des nächsten Wartungsfensters automatisch auf diese Wartungs-Patch-Version aktualisiert, wenn Sie die Modulversion `1.1.1.0` ausführen.

Sie können jeden vorherigen Neptune Engine-Release manuell auf diese Version aktualisieren.

Upgrade auf diesen Release

Important

Ein Upgrade von einer früheren Version auf **`1.1.0.0`** löst auch ein Betriebssystem-Upgrade auf allen Instances in Ihrem DB-Cluster aus. Da aktive Schreibanforderungen, die während des Betriebssystem-Upgrades auftreten, nicht verarbeitet werden, müssen Sie vor dem Start des Upgrades alle Schreib-Workloads für den Cluster, der aktualisiert wird, pausieren, einschließlich Massendatenladungen.
Zu Beginn des Upgrades generiert Neptune einen Snapshot mit einem Namen, der sich aus `preupgrade` gefolgt von einer automatisch generierten Kennung zusammensetzt, die auf Ihren DB-Clusterinformationen basiert. Dieser Snapshot wird Ihnen nicht berechnet und Sie können ihn zur Wiederherstellung Ihres DB-Clusters verwenden, falls während des Upgrade-Vorgangs etwas schief geht.
Wenn das Engine-Upgrade selbst abgeschlossen ist, ist die neue Engine-Version kurzzeitig auf dem alten Betriebssystem verfügbar, aber in weniger als 5 Minuten beginnen alle Instances in Ihrem Cluster gleichzeitig mit einem Betriebssystem-Upgrade. Ihr DB-Cluster wird zu diesem Zeitpunkt für etwa 6 Minuten nicht verfügbar sein. Sie können die Schreib-Workloads fortsetzen, nachdem das Upgrade abgeschlossen ist.
Dieser Prozess generiert die folgenden Ereignisse:

- Ereignismeldungen pro Cluster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- Ereignismeldungen pro Instance:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`

```
    • DB instance restarted
```

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.1.1.0 \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.1.1.0 ^
    --apply-immediately
```

Updates werden auf alle Instances in einem DB-Cluster gleichzeitig angewendet. Ein Update erfordert einen Datenbankneustart auf diesen Instances. Daher kommt es zu einer Ausfallzeit von 20–30 Sekunden bis zu mehreren Minuten. Anschließend können Sie die Nutzung Ihres DB-Clusters fortsetzen.

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue

Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
> 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.
> ```
>
> Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine-Version 1.1.0.0 (19.11.2021)

Ab dem 19.11.2021 wird die Engine-Version 1.1.0.0 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

Important

Ein Upgrade von einer früheren Version auf diese Engine-Version löst **`1.1.0.0`** auch ein Betriebssystem-Upgrade auf allen Instances in Ihrem DB-Cluster aus. Da aktive Schreibanforderungen, die während des Betriebssystem-Upgrades auftreten, nicht verarbeitet werden, müssen Sie vor dem Start des Upgrades alle Schreib-Workloads für den Cluster, der aktualisiert wird, pausieren, einschließlich Massendatenladungen.
Um das Upgrade erfolgreich abzuschließen, muss für jedes Subnetz in jeder Availability Zone (AZ) mindestens eine IP-Adresse pro Neptune-Instance verfügbar sein. Wenn es beispielsweise eine Writer-Instance und zwei Reader-Instances in Subnetz 1 und zwei Reader-Instances in Subnetz 2 gibt, muss Subnetz 1 mindestens 3 freie IP-Adressen und Subnetz 2 mindestens 2 freie IP-Adressen haben, bevor das Upgrade gestartet wird.
Zu Beginn des Upgrades generiert Neptune einen Snapshot mit einem Namen, der sich aus `preupgrade` gefolgt von einer automatisch generierten Kennung zusammensetzt, die auf Ihren DB-Clusterinformationen basiert. Dieser Snapshot wird Ihnen nicht berechnet und Sie können ihn zur Wiederherstellung Ihres DB-Clusters verwenden, falls während des Upgrade-Vorgangs etwas schief geht.
Wenn das Engine-Upgrade selbst abgeschlossen ist, ist die neue Engine-Version kurzzeitig auf dem alten Betriebssystem verfügbar, aber in weniger als 5 Minuten beginnen alle Instances in Ihrem Cluster gleichzeitig mit einem Betriebssystem-Upgrade. Ihr DB-Cluster wird zu diesem Zeitpunkt für einige Minuten nicht verfügbar sein. Sie können die Schreib-Workloads fortsetzen, nachdem das Upgrade abgeschlossen ist.
Dieser Prozess generiert die folgenden Ereignisse:

- Ereignismeldungen pro Cluster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- Ereignismeldungen pro Instance:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`

- `Finished applying off-line patches to DB instance`
- `DB instance restarted`

> **Note**
>
> Ab diesem Engine-Release unterstützt Neptune keine R4-Instance-Typen mehr. Wenn Sie eine R4-Instance in Ihrem DB-Cluster verwenden, müssen Sie sie vor dem Upgrade auf diese Version manuell durch einen anderen Instance-Typ ersetzen. Wenn es sich bei Ihrer Writer-Instance um eine R4 handelt, folgen Sie diesen Anweisungen, um sie zu verschieben.

Nachfolgende Patch-Releases für diesen Release

- Wartungs-Release: 1.1.0.0.R2 (16.05.2022)
- Wartungs-Release: 1.1.0.0.R3 (23.12.2022)

Neue Features in dieser Engine-Version

- Einführung von universellen T4g und speicheroptimierten R6g Datenbank-Instances, die auf dem AWS Graviton2-Prozessor basieren. Graviton2-basierte Instances bieten eine deutlich bessere Leistung price/performance als vergleichbare x86-basierte Instances der aktuellen Generation für eine Vielzahl von Workloads. Anwendungen funktionieren auf diesen neuen Instance-Typen wie gewohnt, und es ist nicht erforderlich, Anwendungscode zu portieren, wenn Sie auf sie aktualisieren.

  Weitere Informationen zu Preisen und regionaler Verfügbarkeit finden Sie auf der Amazon Neptune-Preisseite.
- Einführung von benutzerdefinierten Modellen in Neptune ML.
- Unterstützung von SPARQL-Inferenzabfragen in Neptune ML.
- Es wurde ein neuer Streams-Endpunkt für Eigenschaftsgraphdaten hinzugefügt, und zwar:

```
https://Neptune-DNS:8182/propertygraph/stream
```

Das Ausgabeformat dieses Endpunkts, `PG_JSON`, entspricht exakt dem `GREMLIN_JSON`-Ausgabeformat des alten `gremlin/stream`.

Der neue `propertygraph/stream`-Endpunkt erweitert die Neptune-Stream-Unterstützung auf openCypher und ersetzt den `gremlin/stream`-Endpunkt durch das zugehörige `GREMLIN_JSON`-Ausgabeformat.

Verbesserungen in dieser Engine-Version

- Es wurden Verbesserungen an Neptun-Streams vorgenommen:
  - Dem `records`-Objekt wurde ein `commitTimestamp`-Feld hinzugefügt, das Neptune Streams-Änderungsprotokoll-Antwortformat verwendet, um einen Zeitstempel für jeden Datensatz in einem Änderungsprotokollstream bereitzustellen.
  - Dem `iteratorType`-Parameter wurde ein `LATEST`-Wert hinzugefügt, mit dem Sie die letzte gültige eventId der Streams abrufen können. Siehe Aufrufen der Streams-API.
- Unterstützung für das Abrufen des Inferenz-Konfidenzwerts in Gremlin-Knotenklassifizierungs- und Regressionsabfragen hinzugefügt.
- Unterstützung für die `OPTIONAL MATCH`-Klausel in openCypher hinzugefügt.
- Unterstützung für die `MERGE`-Klausel in openCypher hinzugefügt.
- Unterstützung für die Verwendung von `ORDER BY`- und `WITH`-Klauseln in openCypher hinzugefügt.
- Unterstützung für das Verständnis von Mustern in openCypher wurde hinzugefügt und die Unterstützung für Musterausdrücke wurde erweitert, die über die Existenzprüfung hinausgeht.
- Erweiterte Unterstützung für die `DELETE`- und `DELETE DETACH`-Klauseln in openCypher, sodass sie jetzt mit anderen Update-Klauseln verwendet werden können.
- Erweiterte Unterstützung für `CREATE`- und `UPDATE`-Klauseln, die mit `RETURN` in openCypher verwendet werden.
- In der DFE-Engine wurde die Unterstützung für die Schritte Gremlin `limit`, `range` und `skip` hinzugefügt.
- Die Abfrageausführung in der DFE-Engine wurde verbessert, wenn weder `explain` noch `profile` angefordert wird.
- Die Abfrageausführung in der DFE-Engine für den `value`-Ausdruck wurde verbessert.

- Eine Reihe verketteter bedingter Gremlin-Einfügemuster wurde verbessert, um Ausnahmen bei gleichzeitiger Änderung zu vermeiden und die Verkettung von Abfragemustern wie den folgenden zu ermöglichen:
  - Bedingtes Einfügen von Scheitelpunkten nach ID, z. B.:

```
g.V(ID).fold().coalesce(unfold(), g.addV("L1").property(id,ID))
```

  - Bedingtes Einfügen von Scheitelpunkten mit mehreren Beschriftungen, z. B.:

```
g.V(ID).fold().coalesce(unfold(), g.addV("L1::L2").property(id,ID))
```

  - Bedingtes Einfügen von Edges nach ID, z. B.:

```
g.E(ID).fold().coalesce(unfold(), V(from).addE(label).to(V(to)).property(id, ID))
```

  - Bedingtes Einfügen von Edges mit mehreren Beschriftungen, z. B.:

```
g.E(ID).fold().coalesce(unfold(),
 g.addE(label).from(V(from)).to(V(to)).property(id, ID))
```

  - Bedingtes Einfügen, gefolgt von einer Abfrage, wie zum Beispiel:

```
g.V(ID).fold().coalesce(unfold(),
 g.addV("L1").property(id,ID)).project("myvalues").by(valueMap())
```

  - Bedingtes Einfügen mit zusätzlichen Eigenschaften wie:

```
g.V(ID).fold().coalesce(unfold(),
 g.addV("L1").property(id,ID).property("name","pumba"))
```

In diesem Engine-Version behobene Fehler

- Das Statistikfeature für `T3.medium`-Instance-Typen, die sie nicht unterstützen konnten, wurde deaktiviert.
- Ein SPARQL-Fehler `explain` mit einem `IN`-Feature, die nicht konstante Werte annahm, wurde behoben.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.1.0.0 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprache kompatibel ist:

- Gremlin-Version: `3.4.11`
- SPARQL-Version: `1.1`

Upgrade-Pfade zum Engine-Release 1.1.0.0

Sie können jeden vorherigen Neptune Engine-Release manuell auf diese Version aktualisieren.

Es wird kein automatisches Upgrade auf diese Version durchgeführt.

Upgrade auf diesen Release

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.1.0.0 \
    --allow-major-version-upgrade \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.1.0.0 ^
    --allow-major-version-upgrade ^
    --apply-immediately
```

Statt `--apply-immediately` können Sie `--no-apply-immediately` angeben. Um ein Upgrade auf eine Hauptversion durchzuführen, ist der Parameter erforderlich. allow-major-version-

upgrade Stellen Sie außerdem sicher, dass Sie die Engine-Version angeben, da Ihre Engine sonst möglicherweise auf eine andere Version aktualisiert wird.

Wenn Ihr Cluster eine benutzerdefinierte Cluster-Parametergruppe verwendet, müssen Sie diesen Parameter einschließen, um ihn zu anzugeben:

```
    --db-cluster-parameter-group-name (name of the custom DB cluster parameter group)
```

Ebenso sollte für Instances im Cluster, die eine benutzerdefinierte DB-Parametergruppe verwenden, dieser Parameter eingeschlossen werden, um ihn zu spezifizieren:

```
    --db-instance-parameter-group-name (name of the custom instance parameter group)
```

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
> 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.
> ```
>
> Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Neptune-Wartungsrelease, Version 1.1.0.0.R3 (23.12.2022)

Ab dem 23.12.2022 wird die Engine-Version 1.1.0.0.R3 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

> **Important**
>
> Ein Upgrade von einer früheren Version auf diese Engine-Version löst **`1.1.0.0`** auch ein Betriebssystem-Upgrade auf allen Instances in Ihrem DB-Cluster aus. Da aktive Schreibanforderungen, die während des Betriebssystem-Upgrades auftreten, nicht verarbeitet

werden, müssen Sie vor dem Start des Upgrades alle Schreib-Workloads für den Cluster, der aktualisiert wird, pausieren, einschließlich Massendatenladungen.
Um das Upgrade erfolgreich abzuschließen, muss für jedes Subnetz in jeder Availability Zone (AZ) mindestens eine IP-Adresse pro Neptune-Instance verfügbar sein. Wenn es beispielsweise eine Writer-Instance und zwei Reader-Instances in Subnetz 1 und zwei Reader-Instances in Subnetz 2 gibt, muss Subnetz 1 mindestens 3 freie IP-Adressen und Subnetz 2 mindestens 2 freie IP-Adressen haben, bevor das Upgrade gestartet wird.
Zu Beginn des Upgrades generiert Neptune einen Snapshot mit einem Namen, der sich aus `preupgrade` gefolgt von einer automatisch generierten Kennung zusammensetzt, die auf Ihren DB-Clusterinformationen basiert. Dieser Snapshot wird Ihnen nicht berechnet und Sie können ihn zur Wiederherstellung Ihres DB-Clusters verwenden, falls während des Upgrade-Vorgangs etwas schief geht.
Wenn das Engine-Upgrade selbst abgeschlossen ist, ist die neue Engine-Version kurzzeitig auf dem alten Betriebssystem verfügbar, aber in weniger als 5 Minuten beginnen alle Instances in Ihrem Cluster gleichzeitig mit einem Betriebssystem-Upgrade. Ihr DB-Cluster wird zu diesem Zeitpunkt für einige Minuten nicht verfügbar sein. Sie können die Schreib-Workloads fortsetzen, nachdem das Upgrade abgeschlossen ist.
Dieser Prozess generiert die folgenden Ereignisse:

- Ereignismeldungen pro Cluster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- Ereignismeldungen pro Instance:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Verbesserungen in dieser Engine-Version

- Leistungsverbesserungen und Korrekturkorrekturen für verschiedene Gremlin-Operatoren, darunter `repeat`, `coalesce`, `store` und `aggregate`.

In diesem Engine-Version behobene Fehler

- Ein CPU-Spitzenproblem wurde behoben.
- Es wurde ein openCypher-Fehler behoben, bei dem Abfragen die Zeichenfolge, `"null"`, anstelle eines Nullwerts in Bolt und SPARQL-JSON zurückgaben.
- Es wurde ein Fehler im Auditprotokoll behoben, der dazu führte, dass unnötige Informationen protokolliert wurden und bestimmte Felder in den Protokollen fehlten.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.1.0.0.R3 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprache kompatibel ist:

- Gremlin-Version: `3.4.11`
- openCypher-Version: `Neptune-9.0.20190305-1.0`
- SPARQL-Version: `1.1`

Upgrade-Pfade zum Engine-Release 1.1.0.0.R3

Ihr Cluster wird während des nächsten Wartungsfensters automatisch auf diese Wartungs-Patch-Version aktualisiert, wenn Sie die Modulversion `1.1.0.0` ausführen.

> **Important**
>
> Ein Upgrade von einer früheren Version auf **`1.1.0.0`** löst auch ein Betriebssystem-Upgrade auf allen Instances in Ihrem DB-Cluster aus. Da aktive Schreibanforderungen, die während des Betriebssystem-Upgrades auftreten, nicht verarbeitet werden, müssen Sie vor dem Start des Upgrades alle Schreib-Workloads für den Cluster, der aktualisiert wird, pausieren, einschließlich Massendatenladungen.
>
> Zu Beginn des Upgrades generiert Neptune einen Snapshot mit einem Namen, der sich aus `preupgrade` gefolgt von einer automatisch generierten Kennung zusammensetzt, die auf Ihren DB-Clusterinformationen basiert. Dieser Snapshot wird Ihnen nicht berechnet und Sie können ihn zur Wiederherstellung Ihres DB-Clusters verwenden, falls während des Upgrade-Vorgangs etwas schief geht.
>
> Wenn das Engine-Upgrade selbst abgeschlossen ist, ist die neue Engine-Version kurzzeitig auf dem alten Betriebssystem verfügbar, aber in weniger als 5 Minuten beginnen alle

Instances in Ihrem Cluster gleichzeitig mit einem Betriebssystem-Upgrade. Ihr DB-Cluster wird zu diesem Zeitpunkt für etwa 6 Minuten nicht verfügbar sein. Sie können die Schreib-Workloads fortsetzen, nachdem das Upgrade abgeschlossen ist.
Dieser Prozess generiert die folgenden Ereignisse:

- Ereignismeldungen pro Cluster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- Ereignismeldungen pro Instance:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

Note

Ab diesem Engine-Release unterstützt Neptune keine R4-Instance-Typen mehr. Wenn Sie eine R4-Instance in Ihrem DB-Cluster verwenden, müssen Sie sie vor dem Upgrade auf diese Version manuell durch einen anderen Instance-Typ ersetzen. Wenn es sich bei Ihrer Writer-Instance um eine R4 handelt, folgen Sie diesen Anweisungen, um sie zu verschieben.

Upgrade auf diesen Release

Amazon Neptune 1.1.0.0.R3 ist jetzt allgemein verfügbar.

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
```

```
    --engine-version 1.1.0.0 \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.1.0.0 ^
    --apply-immediately
```

Updates werden auf alle Instances in einem DB-Cluster gleichzeitig angewendet. Ein Update erfordert einen Datenbankneustart auf diesen Instances. Daher kommt es zu einer Ausfallzeit von 20–30 Sekunden bis zu mehreren Minuten. Anschließend können Sie die Nutzung Ihres DB-Clusters fortsetzen.

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
> 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.
> ```
>
> Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Neptune-Wartungsrelease, Version 1.1.0.0.R2 (16.05.2022)

Ab dem 16.05.2022 wird die Engine-Version 1.1.0.0.R2 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

> **Important**
>
> Ein Upgrade von einer früheren Version auf diese Engine-Version löst **`1.1.0.0`** auch ein Betriebssystem-Upgrade auf allen Instances in Ihrem DB-Cluster aus. Da aktive Schreibanforderungen, die während des Betriebssystem-Upgrades auftreten, nicht verarbeitet

werden, müssen Sie vor dem Start des Upgrades alle Schreib-Workloads für den Cluster, der aktualisiert wird, pausieren, einschließlich Massendatenladungen.
Zu Beginn des Upgrades generiert Neptune einen Snapshot mit einem Namen, der sich aus `preupgrade` gefolgt von einer automatisch generierten Kennung zusammensetzt, die auf Ihren DB-Clusterinformationen basiert. Dieser Snapshot wird Ihnen nicht berechnet und Sie können ihn zur Wiederherstellung Ihres DB-Clusters verwenden, falls während des Upgrade-Vorgangs etwas schief geht.
Wenn das Engine-Upgrade selbst abgeschlossen ist, ist die neue Engine-Version kurzzeitig auf dem alten Betriebssystem verfügbar, aber in weniger als 5 Minuten beginnen alle Instances in Ihrem Cluster gleichzeitig mit einem Betriebssystem-Upgrade. Ihr DB-Cluster wird zu diesem Zeitpunkt für einige Minuten nicht verfügbar sein. Sie können die Schreib-Workloads fortsetzen, nachdem das Upgrade abgeschlossen ist.
Dieser Prozess generiert die folgenden Ereignisse:

- Ereignismeldungen pro Cluster:
  - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
  - `Database cluster major version has been upgraded`
- Ereignismeldungen pro Instance:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

In diesem Engine-Version behobene Fehler

- Es wurde ein Fehler behoben, bei dem der interne Cache für Anmeldeinformationen für Endpunkte, die keine Abfragen waren, wie z. B. den Status-Endpunkt, nicht ordnungsgemäß gelöscht wurde.
- Es wurde ein Fehler behoben, der dazu führte, dass die Replikationsverzögerung nach einem Engine-Upgrade zunahm.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.1.0.0.R2 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprache kompatibel ist:

- Gremlin-Version: `3.4.11`
- openCypher-Version: `Neptune-9.0.20190305-1.0`
- SPARQL-Version: `1.1`

Upgrade-Pfade zum Engine-Release 1.1.0.0.R2

Ihr Cluster wird während des nächsten Wartungsfensters automatisch auf diese Wartungs-Patch-Version aktualisiert, wenn Sie die Modulversion `1.1.0.0` ausführen.

> **Important**
>
> Ein Upgrade von einer früheren Version auf **`1.1.0.0`** löst auch ein Betriebssystem-Upgrade auf allen Instances in Ihrem DB-Cluster aus. Da aktive Schreibanforderungen, die während des Betriebssystem-Upgrades auftreten, nicht verarbeitet werden, müssen Sie vor dem Start des Upgrades alle Schreib-Workloads für den Cluster, der aktualisiert wird, pausieren, einschließlich Massendatenladungen.
> Zu Beginn des Upgrades generiert Neptune einen Snapshot mit einem Namen, der sich aus `preupgrade` gefolgt von einer automatisch generierten Kennung zusammensetzt, die auf Ihren DB-Clusterinformationen basiert. Dieser Snapshot wird Ihnen nicht berechnet und Sie können ihn zur Wiederherstellung Ihres DB-Clusters verwenden, falls während des Upgrade-Vorgangs etwas schief geht.
> Wenn das Engine-Upgrade selbst abgeschlossen ist, ist die neue Engine-Version kurzzeitig auf dem alten Betriebssystem verfügbar, aber in weniger als 5 Minuten beginnen alle Instances in Ihrem Cluster gleichzeitig mit einem Betriebssystem-Upgrade. Ihr DB-Cluster wird zu diesem Zeitpunkt für etwa 6 Minuten nicht verfügbar sein. Sie können die Schreib-Workloads fortsetzen, nachdem das Upgrade abgeschlossen ist.
> Dieser Prozess generiert die folgenden Ereignisse:
>
> - Ereignismeldungen pro Cluster:
>   - `Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-`*`(autogenerated snapshot ID)`*`]`
>   - `Database cluster major version has been upgraded`

- Ereignismeldungen pro Instance:
  - `Applying off-line patches to DB instance`
  - `DB instance shutdown`
  - `Finished applying off-line patches to DB instance`
  - `DB instance restarted`

> **Note**
>
> Ab diesem Engine-Release unterstützt Neptune keine R4-Instance-Typen mehr. Wenn Sie eine R4-Instance in Ihrem DB-Cluster verwenden, müssen Sie sie vor dem Upgrade auf diese Version manuell durch einen anderen Instance-Typ ersetzen. Wenn es sich bei Ihrer Writer-Instance um eine R4 handelt, folgen Sie diesen Anweisungen, um sie zu verschieben.

Upgrade auf diesen Release

Amazon Neptune 1.1.0.0.R2 ist jetzt allgemein verfügbar.

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.1.0.0 \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.1.0.0 ^
    --apply-immediately
```

Updates werden auf alle Instances in einem DB-Cluster gleichzeitig angewendet. Ein Update erfordert einen Datenbankneustart auf diesen Instances. Daher kommt es zu einer Ausfallzeit von 20–30 Sekunden bis zu mehreren Minuten. Anschließend können Sie die Nutzung Ihres DB-Clusters fortsetzen.

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:

```
  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.
```

Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine-Version 1.0.5.1 (01.10.2021)

Ab dem 01.10.2021 wird die Engine-Version 1.0.5.1 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

Nachfolgende Patch-Veröffentlichungen für dieses Version

- Release: 1.0.5.1.R2 (26.10.2021)
- Release: 1.0.5.1.R3 (13.01.2022)
- Wartungs-Release: 1.0.5.1.R4 (16.05.2022)

Neue Features in dieser Engine-Version

- Es wurde ein Ergebnis-Cache hinzugefügt, um die Ergebnisse bestimmter Abfragen zwischenzuspeichern.
- Date/time Unterstützung in Neptune OpenCypher hinzugefügt.
- Unterstützung für `List`- und `Map`-Zugriff auf Elemente in Neptune openCypher hinzugefügt.

Verbesserungen in dieser Engine-Version

- Bei Neptune openCypher-Endpunktnamen wird nicht zwischen Groß- und Kleinschreibung unterschieden.
- Verbesserte Erklärung von openCypher.
- Verbesserte Gremlin-Single-Upsert-Abfragemuster, die mit Schritten `iterate()` und `profile()` enden.
- Verbesserte Leistung `keys()`- und `property()`-Features in Gremlin.
- Der Gremlin-`dedup()`-Schritt wird im DFE ausgeführt, wenn er mit globalem Gültigkeitsbereich verwendet wird.
- Die folgenden Gremlin-`HAS`-Prädikate werden in der DFE-Engine ausgeführt, wenn die DFE-Engine aktiviert ist:
  - `EQ`
  - `NEQ`
  - `LT`
  - `LTE`
  - `GT`
  - `GTE`
  - `BETWEEN`
  - `INSIDE`
  - `OUTSIDE`
  - `WITHIN`
  - `AND (connectives)`
  - `OR (connectives)`
- Verbesserung der LIMIT-Abfrageleistung.
- Verbesserte Leistung der allgemeinen Aggregationsabfragen von openCypher.

In diesem Engine-Version behobene Fehler

- Es wurde ein Gremlin-Fehler behoben, durch den eine Edge mit einer anderen Edge verbunden werden konnte.

- Es wurde ein Gremlin-Fehler behoben, der dazu führte, dass eine suboptimale Verbindungsstrategie gewählt wurde.
- Es wurde ein Gremlin-Fehler behoben, der dazu führte, dass die Serialisierung von Knoten und Beziehungen zum Stillstand kam, wenn mehr als 100 Eigenschaften vorhanden waren.
- Es wurde ein Fehler behoben, der die Planung der Abfrageausführung bei Abfragen mit großen Graphmustern verlangsamte.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.0.5.1 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprache kompatibel ist:

- Gremlin-Version: `3.4.11`
- SPARQL-Version: `1.1`

Upgrade-Pfade zum Engine-Release 1.0.5.1

Sie können jeden vorherigen Neptune Engine-Release manuell auf diese Version aktualisieren.

Es wird kein automatisches Upgrade auf diese Version durchgeführt.

Upgrade auf diesen Release

Amazon Neptune 1.0.5.1 ist jetzt allgemein verfügbar.

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.5.1 \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.5.1 ^
    --apply-immediately
```

Updates werden auf alle Instances in einem DB-Cluster gleichzeitig angewendet. Ein Update erfordert einen Datenbankneustart auf diesen Instances. Daher kommt es zu einer Ausfallzeit von 20–30 Sekunden bis zu mehreren Minuten. Anschließend können Sie die Nutzung Ihres DB-Clusters fortsetzen.

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune

einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
>   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.
> ```
>
> Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Neptune-Wartungsrelease, Version 1.0.5.1.R4 (16.05.2022)

Ab dem 16.05.2022 wird die Engine-Version 1.0.5.1.R4 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.0.5.1.R4 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprache kompatibel ist:

- Gremlin-Version: `3.4.11`
- SPARQL-Version: `1.1`

Upgrade-Pfade zum Engine-Release 1.0.5.1.R4

Ihr Cluster wird während des nächsten Wartungsfensters automatisch auf diese Wartungs-Patch-Version aktualisiert, wenn Sie die Modulversion `1.0.5.1` ausführen.

Sie können jeden vorherigen Neptune Engine-Release manuell auf diese Version aktualisieren.

Upgrade auf diesen Release

Amazon Neptune 1.0.5.1.R4 ist jetzt allgemein verfügbar.

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.5.1 \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.5.1 ^
    --apply-immediately
```

Updates werden auf alle Instances in einem DB-Cluster gleichzeitig angewendet. Ein Update erfordert einen Datenbankneustart auf diesen Instances. Daher kommt es zu einer Ausfallzeit von 20–30 Sekunden bis zu mehreren Minuten. Anschließend können Sie die Nutzung Ihres DB-Clusters fortsetzen.

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
> 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.
> ```
>
> Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine-Version 1.0.5.1.R3 (13.01.2022)

Ab dem 13.01.2022 wird die Engine-Version 1.0.5.1.R3 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

In diesem Engine-Version behobene Fehler

- Es wurde ein Fehler behoben, der zu einem Ressourcenleck führen kann, wenn eine Abfrage nicht alle benötigten Ressourcen abruft.
- Es wurde ein kleines Speicherleck während der Abfrageausführung behoben, das durch eine nicht beanspruchte Speicherzuweisung verursacht wurde.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.0.5.1.R3 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprache kompatibel ist:

- Gremlin-Version: `3.4.11`
- SPARQL-Version: `1.1`

Upgrade-Pfade zum Engine-Release 1.0.5.1.R3

Ihr Cluster wird während des nächsten Wartungsfensters automatisch auf diese Patch-Version aktualisiert, wenn Sie die Modulversion `1.0.5.1` ausführen.

Sie können jeden vorherigen Neptune Engine-Release manuell auf diese Version aktualisieren.

Upgrade auf diesen Release

Amazon Neptune 1.0.5.1.R3 ist jetzt allgemein verfügbar.

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.5.1 \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.5.1 ^
    --apply-immediately
```

Updates werden auf alle Instances in einem DB-Cluster gleichzeitig angewendet. Ein Update erfordert einen Datenbankneustart auf diesen Instances. Daher kommt es zu einer Ausfallzeit von 20–30 Sekunden bis zu mehreren Minuten. Anschließend können Sie die Nutzung Ihres DB-Clusters fortsetzen.

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
> 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.
> ```
>
> Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine-Version 1.0.5.1.R2 (26.10.2021)

Ab dem 26.10.2021 wird die Engine-Version 1.0.5.1.R2 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

In diesem Engine-Version behobene Fehler

- Es wurde ein Fehler behoben, der einen Neustart des Servers verursachte, wenn ein vorübergehender Fehler bei der Erstellung einer älteren Version eines Graphelements unter

wiederholter Leseisolierung auftrat. Neptune gibt jetzt stattdessen einen Fehler aus, sodass der Client es erneut versuchen kann.

- Es wurde ein Fehler behoben, der zu einem Serverneustart führte, wenn während eines einzelnen Kardinalitätsupdates ein vorübergehender Fehler auftrat. Neptune gibt jetzt stattdessen einen Fehler aus, sodass der Client es erneut versuchen kann.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.0.5.1.R2 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprache kompatibel ist:

- Gremlin-Version: `3.4.11`
- SPARQL-Version: `1.1`

Upgrade-Pfade zum Engine-Release 1.0.5.1.R2

Ihr Cluster wird während des nächsten Wartungsfensters automatisch auf diese Patch-Version aktualisiert, wenn Sie die Modulversion `1.0.5.1` ausführen.

Sie können jeden vorherigen Neptune Engine-Release manuell auf diese Version aktualisieren.

Upgrade auf diesen Release

Amazon Neptune 1.0.5.1.R2 ist jetzt allgemein verfügbar.

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.5.1 \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
```

```
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.5.1 ^
    --apply-immediately
```

Updates werden auf alle Instances in einem DB-Cluster gleichzeitig angewendet. Ein Update erfordert einen Datenbankneustart auf diesen Instances. Daher kommt es zu einer Ausfallzeit von 20–30 Sekunden bis zu mehreren Minuten. Anschließend können Sie die Nutzung Ihres DB-Clusters fortsetzen.

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
>   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.
> ```
>
> Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine-Version 1.0.5.0 (27.07.2021)

Ab dem 27.07.2021 wird die Engine-Version 1.0.5.0 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

Nachfolgende Patch-Veröffentlichungen für dieses Version

- Release: 1.0.5.0.R2 (16.08.2021)
- Release: 1.0.5.0.R3 (15.09.2021)
- Wartungs-Release: 1.0.5.0.R5 (16.05.2022)

Neue Features in dieser Engine-Version

- Neptune ML wurde mit vielen neuen Features für den Produktionseinsatz veröffentlicht und befindet sich nicht mehr im Labormodus.

- Anfängliche Unterstützung für die openCypher Abfragesprache wurde im Labormodus hinzugefügt. openCypher ist der Open-Source-Standard für die Abfragesprache Cypher. Seine Syntax ist in der Cypher Query Language Reference (Version 9) spezifiziert und wird vom openCypher-Projekt verwaltet.

  Siehe Zugriff auf das Neptun-Diagramm mit openCypher für Informationen zur Neptun-Implementierung der Sprache finden Sie unter.

  Support für das Bolt-Protokoll, das Neptune-Clients für openCypher-Abfragen verwenden, wird ebenfalls unterstützt. Siehe Verwenden des Bolt-Protokolls für openCypher-Abfragen an Neptune.

  Die Support für openCypher ist jetzt automatisch aktiviert, hängt aber von der Neptune-DFE-Engine ab, die derzeit nur im Lab-Modus verfügbar ist. Die Standardeinstellung `DFEQueryEngine` im `neptune_lab_mode` DB-Cluster-Parameter ist jetzt `DFEQueryEngine=viaQueryHint`, was bedeutet, dass die Engine aktiviert ist, aber nur für Abfragen verwendet wird, bei denen der Abfragehinweis `useDFE` vorhanden und auf `true` gesetzt ist. Wenn Sie die DFE-Engine per Einstellung von `DFEQueryEngine=disabled` deaktivieren, können Sie openCypher nicht verwenden.
- Unterstützung für das SPARQL 1.1 Graph Store HTTP-Protokoll hinzugefügt. Siehe Verwenden des SPARQL 1.1-Graph-Store-Protokolls (GSP) über HTTP in Amazon Neptune.
- Die Standardeinstellung für den Labormodus wurde von Neptune-DFE-Engine auf `viaQueryHint` geändert. Das bedeutet, dass die DFE-Engine jetzt standardmäßig aktiviert ist, aber nur für Abfragen verwendet wird, bei denen der Abfragehinweis `useDFE` vorhanden und auf `true` gesetzt ist.
- Es wurde eine neue CloudWatch Amazon-Metrik hinzugefügt`StatsNumStatementsScanned`, um die Berechnung von Statistiken für die Neptune DFE-Engine zu überwachen. Siehe Verwendung der `StatsNumStatementsScanned` CloudWatch Metrik zur Überwachung der Statistikberechnung.

Verbesserungen in dieser Engine-Version

- Unterstützung für Apache 3.4.11 hinzugefügt. TinkerPop

> **Important**
>
> In TinkerPop Version 3.4.11 wurde eine Änderung vorgenommen, die die Richtigkeit der Verarbeitung von Abfragen verbessert, die Abfrageleistung jedoch vorerst ernsthaft beeinträchtigen kann.
> Zum Beispiel könnte eine Abfrage dieser Art deutlich langsamer ausgeführt werden:
>
> ```
> g.V().hasLabel('airport').
>   order().
>     by(out().count(),desc).
>   limit(10).
>   out()
> ```
>
> Die Scheitelpunkte nach dem Limit-Schritt werden jetzt aufgrund der Änderung in 3.4.11 nicht optimal abgerufen. TinkerPop Um dies zu vermeiden, können Sie die Abfrage ändern, indem Sie den Schritt barrier() an einer beliebigen Stelle nach `order().by()` hinzufügen. Zum Beispiel:
>
> ```
> g.V().hasLabel('airport').
>   order().
>     by(out().count(),desc).
>   limit(10).
>   barrier().
>   out()
> ```

- Der SPARQL `joinOrder`-Abfragehinweis wird jetzt von der alternativen Abfrage-Engine Neptune DFE unterstützt.
- Die Ausgabe der Neptune-Status-API wurde erweitert und neu organisiert, um mehr Klarheit über die Einstellungen und Features Ihres DB-Clusters zu schaffen.

  Die neue Ausgabe enthält ein `features`-Objekt der obersten Ebene, das Statusinformationen zu den Features Ihres DB-Clusters enthält, und ein `settings`-Objekt der obersten Ebene, das Einstellungsinformationen enthält. Eine Übersicht über das neue Format finden Sie unter Beispielausgabe des Befehls Instanzstatus.
- Die Verarbeitung von Streaming-Änderungsprotokollen wurde verbessert, wenn `AFTER_SEQUENCE_NUMBER` Streams mit der letzten Ereignis-ID auf dem Server angefordert werden und diese Ereignis-ID bereits abgelaufen ist. Der Server gibt nicht mehr den Fehler

„Abgelaufene Ereignis-ID“ aus, wenn es sich bei der angeforderten Ereignis-ID um die zuletzt gelöschte Ereignis-ID auf dem Server handelt.

In diesem Engine-Version behobene Fehler

- Es wurde ein Gremlin-Fehler im Zusammenhang mit der Reihenfolge numerischer Werte behoben.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.0.5.0 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprache kompatibel ist:

- Gremlin-Version: `3.4.11`
- SPARQL-Version: `1.1`

Upgrade-Pfade zum Engine-Release 1.0.5.0

Sie können jeden vorherigen Neptune Engine-Release manuell auf diese Version aktualisieren.

Es wird kein automatisches Upgrade auf diese Version durchgeführt.

Upgrade auf diesen Release

Amazon Neptune 1.0.5.0 ist jetzt allgemein verfügbar.

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.5.0 \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.5.0 ^
    --apply-immediately
```

Updates werden auf alle Instances in einem DB-Cluster gleichzeitig angewendet. Ein Update erfordert einen Datenbankneustart auf diesen Instances. Daher kommt es zu einer Ausfallzeit von 20–30 Sekunden bis zu mehreren Minuten. Anschließend können Sie die Nutzung Ihres DB-Clusters fortsetzen.

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune

einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
> 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.
> ```
>
> Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Neptune-Wartungsrelease, Version 1.0.5.0.R5 (16.05.2022)

Ab dem 16.05.2022 wird die Engine-Version 1.0.5.0.R5 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.0.5.0.R3 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprache kompatibel ist:

- Gremlin-Version: `3.4.11`
- SPARQL-Version: `1.1`

Upgrade-Pfade zum Engine-Release 1.0.5.0.R5

Ihr Cluster wird während des nächsten Wartungsfensters automatisch auf diese Wartungs-Patch-Version aktualisiert, wenn Sie die Modulversion 1.0.5.0 ausführen.

Sie können jeden vorherigen Neptune Engine-Release manuell auf diese Version aktualisieren.

Upgrade auf diesen Release

Amazon Neptune 1.0.5.0.R5 ist jetzt allgemein verfügbar.

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.5.0 \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.5.0 ^
    --apply-immediately
```

Updates werden auf alle Instances in einem DB-Cluster gleichzeitig angewendet. Ein Update erfordert einen Datenbankneustart auf diesen Instances. Daher kommt es zu einer Ausfallzeit von 20–30 Sekunden bis zu mehreren Minuten. Anschließend können Sie die Nutzung Ihres DB-Clusters fortsetzen.

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
> 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.
> ```
>
> Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine-Version 1.0.5.0.R3 (15.09.2021)

Ab dem 15.09.2021 wird die Engine-Version 1.0.5.0.R3 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

In diesem Engine-Version behobene Fehler

- Es wurde ein Fehler behoben, der dazu führte, dass die Engine in einer der folgenden Situationen nicht mehr reagierte:
  - Ein Massenladevorgang erfolgt gleichzeitig mit der automatischen Statistikberechnung.
  - Eine Statistikberechnung wurde zu dem Zeitpunkt, zu dem bereits eine durchgeführt wurde, manuell angefordert.
- Es wurde ein Fehler bei der Erkennung von Deadlocks und bei der Erfassung von Sperren behoben, der zum Absturz der Engine führen konnte.
- Es wurde ein Gremlin-Fehler behoben, bei dem die Engine einen Fehler ausgab, wenn sie in einer Gremlin-Inferenzabfrage auf unbekannte Daten von einem entfernten ML-Endpunkt stieß.
- Es wurden mehrere Fehler in der ML-Modellverwaltung APIs im Zusammenhang mit Modelltransformationsjobs und Instanzempfehlungen behoben.
- Es wurde ein Fehler behoben, der dazu führen konnte, dass die Engine beim Generieren von Knoten und Kanten abstürzte IDs.
- Es wurde ein Fehler behoben, der die Generierung von Abfrageplänen für Abfragen mit großen Grafikmustern verlangsamte.
- Es wurde ein openCypher-Fehler behoben, der dazu führen konnte, dass eine Abfrage beim Abrufen eines Knotens mit mehr als 100 Eigenschaften zum Stillstand kam.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.0.5.0.R3 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprache kompatibel ist:

- Gremlin-Version: `3.4.11`
- SPARQL-Version: `1.1`

Upgrade-Pfade zum Engine-Release 1.0.5.0.R4

Ihr Cluster wird während des nächsten Wartungsfensters automatisch auf diese Patch-Version aktualisiert, wenn Sie die Modulversion 1.0.5.0 ausführen.

Sie können jeden vorherigen Neptune Engine-Release manuell auf diese Version aktualisieren.

Upgrade auf diesen Release

Amazon Neptune 1.0.5.0.R3 ist jetzt allgemein verfügbar.

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.5.0 \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.5.0 ^
    --apply-immediately
```

Updates werden auf alle Instances in einem DB-Cluster gleichzeitig angewendet. Ein Update erfordert einen Datenbankneustart auf diesen Instances. Daher kommt es zu einer Ausfallzeit von 20–30 Sekunden bis zu mehreren Minuten. Anschließend können Sie die Nutzung Ihres DB-Clusters fortsetzen.

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
> 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.
> ```
>
> Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine Version 1.0.5.0.R2 (16.08.2021)

Ab dem 16.08.2023 wird die Engine-Version 1.0.5.0.R2 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

In diesem Engine-Version behobene Fehler

- Eine in der Engine-Version `1.0.5.0` vorgenommene Optimierung wurde deaktiviert, durch die der Neptune-Lookup-Cache Engine-Neustarts auf Replikas überstand. Bei Neustarts von Replikas wird jetzt der Lookup-Cache gelöscht.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.0.5.0.R2 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprache kompatibel ist:

- Gremlin-Version: `3.4.11`
- SPARQL-Version: `1.1`

Upgrade-Pfade zum Engine-Release 1.0.5.0.R2

Ihr Cluster wird während des nächsten Wartungsfensters automatisch auf diese Patch-Version aktualisiert, wenn Sie die Modulversion `1.0.5.0` ausführen.

Sie können jeden vorherigen Neptune Engine-Release manuell auf diese Version aktualisieren.

Upgrade auf diesen Release

Amazon Neptune 1.0.5.0.R2 ist jetzt allgemein verfügbar.

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.5.0 \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.5.0 ^
    --apply-immediately
```

Updates werden auf alle Instances in einem DB-Cluster gleichzeitig angewendet. Ein Update erfordert einen Datenbankneustart auf diesen Instances. Daher kommt es zu einer Ausfallzeit von 20–30 Sekunden bis zu mehreren Minuten. Anschließend können Sie die Nutzung Ihres DB-Clusters fortsetzen.

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
>   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.
> ```
>
> Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine-Version 1.0.4.2 (01.06.2021)

> **Note**
>
> Die Engine-Release-Version 1.0.4.2.R2 war die erste Version von 1.0.4.2, die tatsächlich veröffentlicht wurde.

Themen

- Amazon Neptune Engine-Version 1.0.4.2.R5 (16.08.2021)
- Amazon Neptune Engine-Version 1.0.4.2.R4 (23.07.2021)
- Amazon Neptune Engine-Version 1.0.4.2.R3 (28.06.2021)
- Amazon Neptune Engine-Version 1.0.4.2.R1 (01.06.2021)
- Amazon Neptune Engine-Version 1.0.4.2.R1 (27.05.2021)

Amazon Neptune Engine-Version 1.0.4.2.R5 (16.08.2021)

Ab dem 16.08.2021 wird die Engine-Version 1.0.4.2.R5 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

In diesem Engine-Version behobene Fehler

- Eine in der Engine-Version `1.0.4.2.R4` vorgenommene Optimierung wurde deaktiviert, durch die der Neptune-Lookup-Cache Engine-Neustarts auf Replikas überstand. Bei Neustarts von Replikas wird jetzt der Lookup-Cache gelöscht.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.0.4.2.R5 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprache kompatibel ist:

- Gremlin-Version: `3.4.10`
- SPARQL-Version: `1.1`

Upgrade-Pfade zum Engine-Release 1.0.4.2.R5

Ihr Cluster wird während des nächsten Wartungsfensters automatisch auf diese Patch-Version aktualisiert, wenn Sie die Modulversion `1.0.4.2` ausführen.

Sie können jeden vorherigen Neptune Engine-Release manuell auf diese Version aktualisieren.

Amazon Neptune Engine-Version 1.0.4.2.R4 (23.07.2021)

Ab dem 23.07.2021 wird die Engine-Version 1.0.4.2.R4 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

Verbesserungen in dieser Engine-Version

- Das Verhalten des Lookup-Caches wurde verbessert, um ein redundantes Löschen des Caches nach einem schnellen Reset auf einem Replika zu vermeiden.
- Die Verarbeitung von Streaming-Änderungsprotokollen wurde verbessert, wenn `AFTER_SEQUENCE_NUMBER` Streams mit der letzten Ereignis-ID auf dem Server angefordert werden und diese Ereignis-ID bereits abgelaufen ist. Der Server gibt nicht mehr den Fehler „Abgelaufene Ereignis-ID“ aus, wenn es sich bei der angeforderten Ereignis-ID um die zuletzt gelöschte Ereignis-ID auf dem Server handelt.

In diesem Engine-Version behobene Fehler

- Es wurde ein in 1.0.4.0.R1 eingeführter Fehler behoben, bei dem Abfragen nicht die Gesamtheit von Zeichenkettenwerten mit mehr als 760 Zeichen zurückgaben. Die von diesem Fehler betroffenen Begriffe waren RDF-Literale und/oder Gremlin URIs IDs, Schlüssel und Zeichenkettenwerte.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.0.4.2.R4 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprache kompatibel ist:

- Gremlin-Version: `3.4.10`
- SPARQL-Version: `1.1`

Upgrade-Pfade zum Engine-Release 1.0.4.2.R4

Ihr Cluster wird während des nächsten Wartungsfensters automatisch auf diese Patch-Version aktualisiert, wenn Sie die Modulversion `1.0.4.2` ausführen.

Sie können jeden vorherigen Neptune Engine-Release manuell auf diese Version aktualisieren.

Amazon Neptune Engine-Version 1.0.4.2.R3 (28.06.2021)

Ab dem 28.06.2021 wird die Engine-Version 1.0.4.2.R3 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

Bekannte Probleme in diesem Engine-Release

Problem:

Ein SPARQL-Fehler, der den Medientyp in einem `Accept` Header nicht berücksichtigt, wenn Leerzeichen vorhanden sind.

Zum Beispiel eine Abfrage, die statt einer CSV-Ausgabe eine JSON-Ausgabe `-H "Accept: text/csv; q=1.0, */*; q=0.1"` zurückgibt.

Workaround:

Wenn Sie die Leerzeichen in der `Accept`-Klausel in der Kopfzeile entfernen, gibt die Engine die Ausgabe im richtigen angeforderten Format zurück. Mit anderen Worten, verwenden Sie anstelle von `-H "Accept: text/csv; q=1.0, */*; q=0.1"` :

```
-H "Accept: text/csv;q=1.0,*/*;q=0.1"
```

In diesem Engine-Version behobene Fehler

- Ein Fehler beim Löschen des Lookup-Cache auf Replikas nach einem schnellen Reset wurde behoben.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.0.4.2.R3 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprache kompatibel ist:

- Gremlin-Version: `3.4.10`
- SPARQL-Version: `1.1`

Upgrade-Pfade zum Engine-Release 1.0.4.2.R3

Diese Patch-Version ist optional, es sei denn, Ihr DB-Cluster verwendet eine oder mehrere `R5d` Instances. Wenn Ihr Cluster `R5d` Instances hat, wird er im nächsten Wartungszeitfenster automatisch aktualisiert. Andernfalls wird es nicht automatisch auf diese Patch-Version aktualisiert.

Sie können die `1.0.4.2.R3` Version `1.0.4.2.R2` manuell mit dem AWS CLI apply-pending-maintenance-actionBefehl (der ApplyPendingMaintenanceAction API) auf diese Version aktualisieren.

Amazon Neptune Engine-Version 1.0.4.2.R1 (01.06.2021)

Ab dem 01.06.2021 wird die Engine-Version 1.0.4.2.R2 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

Nachfolgende Patch-Veröffentlichungen für dieses Version

- Release: 1.0.4.2.R3 (28.06.2021)

Bekannte Probleme in diesem Engine-Release

Problem:

Ein SPARQL-Fehler, der den Medientyp in einem `Accept` Header nicht berücksichtigt, wenn Leerzeichen vorhanden sind.

Zum Beispiel eine Abfrage, die statt einer CSV-Ausgabe eine JSON-Ausgabe `-H "Accept: text/csv; q=1.0, */*; q=0.1"` zurückgibt.

Workaround:

Wenn Sie die Leerzeichen in der `Accept`-Klausel in der Kopfzeile entfernen, gibt die Engine die Ausgabe im richtigen angeforderten Format zurück. Mit anderen Worten, verwenden Sie anstelle von `-H "Accept: text/csv; q=1.0, */*; q=0.1"` :

```
-H "Accept: text/csv;q=1.0,*/*;q=0.1"
```

Neue Features in dieser Engine-Version

- Der neue R5d-Instance-Typ wurde hinzugefügt, der einen Lookup-Cache zur Beschleunigung von Lesevorgängen in Anwendungsfällen mit einem hohen Volumen an Eigenschaftswerten oder RDF-Literal-Suchen beinhaltet. Siehe Neptune-Nachschlage-Cache kann Leseabfragen beschleunigen.
- Es wurde ein neuer Labormodus-Parameter hinzugefügt, mit dem die experimentelle DFE-Engine nur pro Abfrage mit dem Abfragehinweis `useDFE` aufgerufen werden kann.

Verbesserungen in dieser Engine-Version

- Unterstützung für TinkerPop 3.4.10 hinzugefügt.

- Unterstützung für die Verwendung des Konfigurationsschritts `withStrategies( )` beim Senden von Gremlin-Skriptanfragen hinzugefügt. Insbesondere werden die `SubgraphStrategy`, `PartitionStrategy`, `ReadOnlyStrategy`, `EdgeLabelVerificationStrategy` und alle `ReservedKeysVerificationStrategy` unterstützt.
- Es wurde eine Optimierung für `V()` Traversale während einer Abfrage hinzugefügt. Bisher wurden solche Traversalen in Neptune nicht optimiert.
- Unterstützung für RFC 2141 URNs zur Verwendung als `namedGraphUri` UND-Parameter für einen `baseUri` Bulk-Load wurde hinzugefügt.

In diesem Engine-Version behobene Fehler

- Es wurde ein Gremlin-Fehler im Parser behoben, durch den falsche Abfragen als gültig behandelt wurden.
- Es wurde ein Gremlin-Fehler behoben, bei dem das Entfalten eines `aggregate()` Nebeneffekts mit `cap().unfold()` zu einer `valueMap()` zu einer Ausnahme führte.
- Es wurde ein Gremlin-Fehler behoben, bei dem einige `property()`-Schritte nach einem `addV()`-Schritt mit dem Fehler „Kann nicht in String umgewandelt werden“ fehlschlugen.
- Es wurde ein Gremlin-Fehler behoben, der verhinderte, dass einige bedingte Einfügemuster Ausnahmen bei gleichzeitiger Änderung auslösten.
- Es wurde ein Gremlin-Fehler behoben, sodass das Timeout für Abfrageanfragen das Sitzungs-Timeout nicht überschreiten kann.
- Es wurde ein SPARQL-Fehler behoben, bei dem Aktualisierungen mit LOAD oder UNLOAD mit einem HTTP-Code 500 statt HTTP-Code 400 fehlschlagen konnten, wenn der Remoteserver nicht verfügbar war.
- Es wurde ein Fehler behoben, bei dem Stream-API-Aufrufe fehlschlugen, wenn `commitNum`- oder opNum-Werte verwendet wurden, die über dem 32-Bit-Grenzwert für vorzeichenbehaftete Ganzzahlen (2.147.483.647) lagen.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.0.4.2.R2 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprache kompatibel ist:

- Gremlin-Version: `3.4.10`
- SPARQL-Version: `1.1`

Upgrade-Pfade zum Engine-Release 1.0.4.2.R2

Sie können jeden vorherigen Neptune Engine-Release manuell auf diese Version aktualisieren.

Es wird kein automatisches Upgrade auf diese Version durchgeführt.

Upgrade auf diesen Release

Amazon Neptune 1.0.4.2.R2 ist jetzt allgemein verfügbar.

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.4.2 \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.4.2 ^
    --apply-immediately
```

Updates werden auf alle Instances in einem DB-Cluster gleichzeitig angewendet. Ein Update erfordert einen Datenbankneustart auf diesen Instances. Daher kommt es zu einer Ausfallzeit von 20–30 Sekunden bis zu mehreren Minuten. Anschließend können Sie die Nutzung Ihres DB-Clusters fortsetzen.

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
> 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.
> ```
>
> Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine-Version 1.0.4.2.R1 (27.05.2021)

Die Engine-Version 1.0.4.2.R1 wurde nie bereitgestellt.

Amazon Neptune Engine-Version 1.0.4.1 (08.12.2020)

Ab dem 08.12.2020 wird die Engine-Version 1.0.4.1 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

Nachfolgende Patch-Veröffentlichungen für dieses Version

- Release: 1.0.4.1.R1.1 (22.03.2021)
- Release: 1.0.4.1.R2 (24.02.2021)

> **Important**
>
> Release: 1.0.4.0 (12.10.2020) hat TLS 1.2 und HTTPS für alle Verbindungen zu Amazon Neptune zur Pflicht gemacht. Ein Fehler in dieser Version hat es jedoch ermöglicht, dass HTTP-Verbindungen, and/or veraltete TLS-Verbindungen, weiterhin für Kunden funktionieren, die zuvor einen DB-Cluster-Parameter festgelegt haben, um die Durchsetzung von HTTPS-Verbindungen zu verhindern.
> Dieser Fehler wurde in den Patch-Versionen 1.0.4.0.R2 und 1.0.4.1.R2 behoben, aber die Behebung hat zu unerwarteten Verbindungsfehlern geführt, wenn die Patches automatisch installiert wurden. Aus diesem Grund wurden beide Patches rückgängig gemacht und können nur manuell installiert werden, um Ihnen die Möglichkeit zu geben, Ihr Setup für TLS 1.2 zu aktualisieren.
> Die Verwendung SSL/TLS für alle Verbindungen zu Neptune wirkt sich auf Ihre Verbindungen mit der Gremlin-Konsole, dem Gremlin-Treiber, Gremlin Python, .NET, NodeJS, REST und auch Load-Balancer-Verbindungen aus. APIs Wenn Sie bisher HTTP oder eine ältere TLS-Version für einen oder alle diese Bereiche verwendet haben, müssen Sie den entsprechenden Client und die Treiber aktualisieren und Ihren Code so ändern, dass er ausschließlich HTTPS verwendet, bevor Sie Ihr System auf die neuesten Patches aktualisieren.

Neue Features in dieser Engine-Version

- Einführung der Neptune ML-Funktion, die Amazon Neptune leistungsstarke Funktionen für Machine Learning bietet. Siehe Amazon Neptune ML für Machine Learning anhand von Diagrammen.
- Es wurde eine benutzerdefinierte `UNLOAD` SPARQL-Operation zum Entfernen von Daten hinzugefügt, die aus einer Remote-Quelle abgerufen wurden. Siehe SPARQL UPDATE UNLOAD.

Verbesserungen in dieser Engine-Version

- Einige bedingte Gremlin-Einfügemuster wurden optimiert, um Ausnahmen bei gleichzeitigen Änderungen zu vermeiden.

In diesem Engine-Version behobene Fehler

- Es wurde ein Gremlin-Fehler behoben, der dazu führen konnte, dass Ergebnisse für ein bestimmtes Muster von Abfragen, die den Schritt `as()` verwendet haben, fehlten.
- Es wurde ein Gremlin-Fehler behoben, der zu Fehlern führen konnte, wenn der Schritt `project()` in einem anderen Schritt verschachtelt war, wie `union()`.
- Ein Gremlin-Fehler im `project()`-Schritt wurde behoben.
- Es wurde ein Gremlin-Fehler bei Traversalen von Zeichenketten behoben, bei dem der Schritt `none()` nicht funktionierte.
- Es wurde ein Gremlin-Fehler bei Traversalen von Zeichenketten behoben, bei dem eine leere Map nicht als Argument für den Schritt `inject()` unterstützt wurde.
- Es wurde ein Gremlin-Fehler behoben, durch den eine Terminalmethode wie `toList()` in der DFE-Engine bei der Ausführung auf Zeichenketten basierender Traversalausführung nicht richtig funktionierte.
- Ein Gremlin-Fehler wurde behoben, durch den Transaktionen, die den Schritt `iterate()` in String-Abfragen verwendeten, nicht geschlossen werden konnten.
- Ein Gremlin-Fehler wurde behoben, der dazu führen konnte, dass Abfragen, die das `is(P.gte(0))`-Muster verwendeten, in manchen Situationen eine Exception auslösten.
- Ein Gremlin-Fehler wurde behoben, der dazu führen konnte, dass Abfragen, die das `order().by(T.id)`-Muster verwendeten, in manchen Situationen eine Exception auslösten.

- Ein Gremlin-Fehler wurde behoben, der dazu führen konnte, dass Abfragen, die das `addV().aggregate()`-Muster verwendeten, in manchen Situationen falsche Ergebnisse lieferten.
- Ein Gremlin-Fehler wurde behoben, der dazu führen konnte, dass Abfragen, die das Schritt-`path()`-Muster gefolgt von Schritt `project()` verwendeten, in manchen Situationen eine Exception auslösten.
- Es wurde ein SPARQL-Fehler behoben, bei dem die Funktion `SUBSTR` einen Fehler signalisiert, anstatt eine leere Zeichenfolge zurückzugeben.
- Es wurde ein Fehler in der DFE-Engine behoben, der dazu führen konnte, dass Join-Operationen in nicht blockierenden Abfrageplänen bei Vorhandensein ungebundener Variablen falsche Ergebnisse generierten.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.0.4.1 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprache kompatibel ist:

- Gremlin-Version: `3.4.8`
- SPARQL-Version: `1.1`

Upgrade-Pfade zum Engine-Release 1.0.4.1

Ihr Cluster wird während des nächsten Wartungsfensters automatisch auf diese Patch-Version aktualisiert, wenn Sie die Modulversion `1.0.4.1` ausführen.

Sie können jeden vorherigen Neptune Engine-Release manuell auf diese Version aktualisieren.

Upgrade auf diesen Release

Amazon Neptune 1.0.4.1 ist jetzt allgemein verfügbar.

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.4.1 \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.4.1 ^
    --apply-immediately
```

Updates werden auf alle Instances in einem DB-Cluster gleichzeitig angewendet. Ein Update erfordert einen Datenbankneustart auf diesen Instances. Daher kommt es zu einer Ausfallzeit von 20–30 Sekunden bis zu mehreren Minuten. Anschließend können Sie die Nutzung Ihres DB-Clusters fortsetzen.

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
> 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.
> ```
>
> Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine-Version 1.0.4.1.R1.1 (22.03.2021)

Ab dem 22.03.2021 wird die Engine-Version 1.0.4.1.R1.1 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

In diesem Engine-Version behobene Fehler

- Eine Optimierung für bedingte Gremlin-Einfügemuster, mit denen bestehende Labels und Eigenschaften hinzugefügt oder an diese angehängt werden können, wurde deaktiviert.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.0.4.1.R1.1 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprache kompatibel ist:

- Gremlin-Version: `3.4.8`
- SPARQL-Version: `1.1`

Upgrade-Pfade zum Engine-Release 1.0.4.1.R1.1

Ihr Cluster wird während des nächsten Wartungsfensters automatisch auf diese Patch-Version aktualisiert, wenn Sie die Modulversion `1.0.4.1` ausführen.

Upgrade auf diesen Release

Amazon Neptune 1.0.4.1.R1.1 ist jetzt allgemein verfügbar.

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.4.1 \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.4.1 ^
    --apply-immediately
```

Updates werden auf alle Instances in einem DB-Cluster gleichzeitig angewendet. Ein Update erfordert einen Datenbankneustart auf diesen Instances. Daher kommt es zu einer Ausfallzeit von 20–30 Sekunden bis zu mehreren Minuten. Anschließend können Sie die Nutzung Ihres DB-Clusters fortsetzen.

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
> 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.
```

Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine-Version 1.0.4.1.R2 (24.02.2021)

Ab dem 24.02.2021 wird die Engine-Version 1.0.4.1.R2 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

Nachfolgende Patch-Veröffentlichungen für dieses Version

- Release: 1.0.4.1.R2.1 (11.03.2021)

Neue Features in dieser Engine-Version

- Neptune unterstützt jetzt die Komprimierung einzelner Dateien im `bzip2`-Format zum Massenladen. Siehe Ladedatenformate.

In diesem Engine-Version behobene Fehler

- Es wurde ein Fehler in Release: 1.0.4.0 (12.10.2020) behoben, der Verbindungen zu Neptune über `HTTP` oder frühere Versionen von TLS ermöglichte, anstatt über `HTTPS` und TLS 1.2.

> **Important**
>
> Es kann eine bahnbrechende Änderung sein, SSL/TLS für alle Verbindungen zu Neptune verwenden zu müssen. Es wirkt sich auf Ihre Verbindungen mit der Gremlin-Konsole, dem Gremlin-Treiber, Gremlin Python, .NET, NodeJS, REST und auch Load-Balancer-Verbindungen aus. APIs Wenn Sie bisher HTTP oder eine ältere TLS-Version für einige

oder alle davon verwendet haben, müssen Sie den entsprechenden Client und die Treiber aktualisieren, bevor Sie diesen Patch installieren, und Ihren Code so ändern, dass er ausschließlich HTTPS verwendet.

- Es wurde ein Gremlin-Fehler behoben, bei dem unter bestimmten Umständen `InternalFailureException` als Antwortcode festgelegt wurde, wenn ein `ConcurrentModificationException` auftrat.
- Es wurde ein Gremlin-Fehler behoben, bei dem das Aktualisieren von Edges oder Scheitelpunkten unter bestimmten Bedingungen zu einem transienten `InternalFailureException` führen konnte.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.0.4.1.R2 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprache kompatibel ist:

- Gremlin-Version: `3.4.8`
- SPARQL-Version: `1.1`

Upgrade-Pfade zum Engine-Release 1.0.4.1.R2

Ihr Cluster wird während des nächsten Wartungsfensters automatisch auf diese Patch-Version aktualisiert, wenn Sie die Modulversion `1.0.4.1` ausführen.

Upgrade auf diesen Release

Amazon Neptune 1.0.4.1.R2 ist jetzt allgemein verfügbar.

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.4.1 \
```

```
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.4.1 ^
    --apply-immediately
```

Updates werden auf alle Instances in einem DB-Cluster gleichzeitig angewendet. Ein Update erfordert einen Datenbankneustart auf diesen Instances. Daher kommt es zu einer Ausfallzeit von 20–30 Sekunden bis zu mehreren Minuten. Anschließend können Sie die Nutzung Ihres DB-Clusters fortsetzen.

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
> 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.
> ```
>
> Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine-Version 1.0.4.1.R2.1 (11.03.2021)

Ab dem 11.03.2021 wird die Engine-Version 1.0.4.1.R2.1 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

In diesem Engine-Version behobene Fehler

- Eine Optimierung für bedingte Gremlin-Einfügemuster, mit denen bestehende Labels und Eigenschaften hinzugefügt oder an diese angehängt werden können, wurde deaktiviert.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.0.4.1.R2.1 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprache kompatibel ist:

- Gremlin-Version: `3.4.8`
- SPARQL-Version: `1.1`

Upgrade-Pfade zum Engine-Release 1.0.4.1.R2.1

Ihr Cluster wird während des nächsten Wartungsfensters automatisch auf diese Patch-Version aktualisiert, wenn Sie die Modulversion `1.0.4.1.R2` ausführen.

Sie können jeden vorherigen Neptune Engine-Release manuell auf diese Version aktualisieren.

Upgrade auf diesen Release

Amazon Neptune 1.0.4.1.R2.1 ist jetzt allgemein verfügbar.

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.4.1.R2 \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.4.1.R2 ^
    --apply-immediately
```

Updates werden auf alle Instances in einem DB-Cluster gleichzeitig angewendet. Ein Update erfordert einen Datenbankneustart auf diesen Instances. Daher kommt es zu einer Ausfallzeit von

20–30 Sekunden bis zu mehreren Minuten. Anschließend können Sie die Nutzung Ihres DB-Clusters fortsetzen.

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:

```
  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.
```

Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine-Version 1.0.4.0 (12.10.2020)

Ab dem 12.10.2023 wird die Engine-Version 1.0.4.0 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

Nachfolgende Patch-Veröffentlichungen für dieses Version

- Release: 1.0.4.0.R2 (24.02.2021)

Neue Features in dieser Engine-Version

- Komprimierung auf Frame-Ebene für Gremlin hinzugefügt.

Verbesserungen in dieser Engine-Version

- Amazon Neptune erfordert jetzt die Verwendung von Secure Sockets Layer (SSL) mit dem TLSv1 2.2-Protokoll für alle Verbindungen zu Neptune in allen Regionen, wobei diese starken Verschlüsselungssammlungen verwendet werden:
  - `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`

Dies gilt sowohl für REST als auch für WebSocket Verbindungen zu Neptune und bedeutet, dass Sie HTTPS anstelle von HTTP verwenden müssen, wenn Sie in allen Regionen eine Verbindung zu Neptune herstellen.

Da Client-Verbindungen, die HTTP oder TLS 1.1 verwenden, nirgends mehr unterstützt werden, stellen Sie bitte sicher, dass Ihre Clients und Ihr Code für die Verwendung von TLS 1.2 und HTTPS aktualisiert wurden, bevor Sie auf diesen Engine-Release aktualisieren.

> **Important**
>
> Es kann eine bahnbrechende Änderung sein, SSL/TLS für alle Verbindungen zu Neptune verwenden zu müssen. Es wirkt sich auf Ihre Verbindungen mit der Gremlin-Konsole, dem Gremlin-Treiber, Gremlin Python, .NET, NodeJS, REST und auch Load-Balancer-Verbindungen aus. APIs Wenn Sie HTTP für einige oder alle dieser Features verwendet haben, müssen Sie jetzt den entsprechenden Client und die Treiber aktualisieren und Ihren Code so ändern, dass HTTPS verwendet wird, sonst schlagen Ihre Verbindungen fehl.
> Ein Fehler in dieser Version hat dazu geführt, dass HTTP-Verbindungen, and/or veraltete TLS-Verbindungen, weiterhin für Kunden funktionieren, die zuvor einen DB-Cluster-Parameter festgelegt haben, um die Durchsetzung von HTTPS-Verbindungen zu verhindern. Dieser Fehler wurde in den Patch-Versionen 1.0.4.0.R2 und 1.0.4.1.R2 behoben, aber die Behebung hat zu unerwarteten Verbindungsfehlern geführt, wenn die Patches automatisch installiert wurden.
> Aus diesem Grund wurden beide Patches rückgängig gemacht und können nur manuell installiert werden, um Ihnen die Möglichkeit zu geben, Ihr Setup für TLS 1.2 zu aktualisieren.

- Auf Version 3.4.8 aktualisiert TinkerPop . Dies ist ein abwärtskompatibles Upgrade. Was neu ist, finden Sie im TinkerPopÄnderungsprotokoll.
- Verbesserte Leistung für den Gremlin-Schritt `properties()`.
- Details zu `BindOp` und `MultiplexerOp` in Erklärungs- und Profilberichten hinzugefügt.

- Es wurde ein Daten-Prefetch hinzugefügt, um die Leistung bei Cache-Fehlern zu verbessern.
- Dem `parserConfiguration` Parameter des Bulk-Loaders wurde eine neue `allowEmptyStrings`-Einstellung hinzugefügt, die es ermöglicht, leere Zeichenketten beim Laden von CSV-Dateien als gültige Eigenschaftswerte zu behandeln (siehe Neptune-Loader-Anforderungsparameter).
- Der Loader erlaubt jetzt ein maskiertes Semikolon in mehrwertigen CSV-Spalten.

In diesem Engine-Version behobene Fehler

- Ein potenzielles Gremlin-Speicherleck im Zusammenhang mit dem `both()`-Schritt wurde behoben.
- Es wurde ein Fehler behoben, bei dem Anforderungsmetriken fehlten, weil ein Endpunkt, der auf ‚/' endete, nicht korrekt behandelt wurde.
- Es wurde ein Fehler behoben, der dazu führte, dass Replikas unter hoher Last ins Hintertreffen gerieten und neu gestartet wurden, wenn die DFE-Engine im Lab-Modus aktiviert war.
- Es wurde ein Fehler behoben, der verhinderte, dass die richtige Fehlermeldung gemeldet wurde, wenn ein Massenladevorgang aufgrund eines Fehlers out-of-memory fehlschlug.
- Es wurde ein SPARQL-Fehler behoben, bei dem die Zeichenkodierung in den Content-Encoding-Header von SPARQL-Abfrageantworten eingefügt wurde. `charset` wird jetzt im Content-Type-Header platziert, sodass HTTP-Clients den verwendeten Zeichensatz automatisch erkennen können.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.0.4.0 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprache kompatibel ist:

- Gremlin-Version: `3.4.8`
- SPARQL-Version: `1.1`

Upgrade-Pfade zum Engine-Release 1.0.4.0

Sie können jeden vorherigen Neptune Engine-Release manuell auf diese Version aktualisieren.

Es wird kein automatisches Upgrade auf diese Version durchgeführt.

Upgrade auf diesen Release

Amazon Neptune 1.0.4.0 ist jetzt allgemein verfügbar.

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.4.0 \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.4.0 ^
    --apply-immediately
```

Updates werden auf alle Instances in einem DB-Cluster gleichzeitig angewendet. Ein Update erfordert einen Datenbankneustart auf diesen Instances. Daher kommt es zu einer Ausfallzeit von 20–30 Sekunden bis zu mehreren Minuten. Anschließend können Sie die Nutzung Ihres DB-Clusters fortsetzen.

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue

Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
> 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.
> ```
>
> Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine-Version 1.0.4.0.R2 (24.02.2021)

Ab dem 24.02.2021 wird die Engine-Version 1.0.4.0.R2 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

In diesem Engine-Version behobene Fehler

- Es wurde ein Fehler in Release: 1.0.4.0 (12.10.2020) behoben, der Verbindungen zu Neptune über HTTP oder frühere Versionen von TLS ermöglichte, anstatt über HTTPS und TLS 1.2.

  > **Important**
  >
  > Es kann eine bahnbrechende Änderung sein, SSL/TLS für alle Verbindungen zu Neptune verwenden zu müssen. Es wirkt sich auf Ihre Verbindungen mit der Gremlin-Konsole, dem Gremlin-Treiber, Gremlin Python, .NET, NodeJS, REST und auch Load-Balancer-Verbindungen aus. APIs Wenn Sie bisher HTTP oder eine ältere TLS-Version für einige oder alle davon verwendet haben, müssen Sie den entsprechenden Client und die Treiber aktualisieren, bevor Sie diesen Patch installieren, und Ihren Code so ändern, dass er ausschließlich HTTPS verwendet.

- Es wurde ein Fehler beim Massenladen von CSVs behoben, der Labels betraf, die auf # enden.
- Es wurde ein Gremlin-Fehler behoben, bei dem unter bestimmten Umständen `InternalFailureException` als Antwortcode festgelegt wurde, wenn ein `ConcurrentModificationException` auftrat.
- Es wurde ein Gremlin-Fehler behoben, bei dem das Aktualisieren von Edges oder Scheitelpunkten unter bestimmten Bedingungen zu einem transienten `InternalFailureException` führen konnte.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.0.4.0.R2 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprache kompatibel ist:

- Gremlin-Version: `3.4.8`
- SPARQL-Version: `1.1`

Upgrade-Pfade zum Engine-Release 1.0.4.0.R2

Ihr Cluster wird während des nächsten Wartungsfensters automatisch auf diese Patch-Version aktualisiert, wenn Sie die Modulversion `1.0.4.0` ausführen.

Sie können jeden vorherigen Neptune Engine-Release manuell auf diese Version aktualisieren.

Upgrade auf diesen Release

Amazon Neptune 1.0.4.0.R2 ist jetzt allgemein verfügbar.

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.4.0 \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.4.0 ^
    --apply-immediately
```

Updates werden auf alle Instances in einem DB-Cluster gleichzeitig angewendet. Ein Update erfordert einen Datenbankneustart auf diesen Instances. Daher kommt es zu einer Ausfallzeit von 20–30 Sekunden bis zu mehreren Minuten. Anschließend können Sie die Nutzung Ihres DB-Clusters fortsetzen.

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein

Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
> 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.
> ```

Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine-Version 1.0.3.0 (08.03.2020)

Ab dem 08.03.2023 wird die Engine-Version 1.0.3.0 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

Nachfolgende Patch-Veröffentlichungen für dieses Version

- Release: 1.0.3.0.R2 (12.10.2020)
- Release: 1.0.3.0.R3 (19.02.2021)

Neue Features in dieser Engine-Version

- Neptune hat eine neue, alternative Abfrage-Engine (DFE) eingeführt, die die Abfrageausführung erheblich beschleunigen kann. Siehe Die alternative Abfrage-Engine (DFE) von Amazon Neptune.
- Das DFE stützt sich auf vorgenerierte Statistiken über Ihre Neptun-Grafikdaten, die über neue Statistikendpunkte verwaltet werden. Siehe DFE-Statistiken.
- Sie können jetzt Ladeaufträge in der Warteschlange aus der Liste der von der Get-Status Loader-API IDs zurückgegebenen Lasten ausschließen, indem Sie den neuen `includeQueuedLoads` Parameter auf FALSE setzen. Siehe Neptune Loader-Anforderungsparameter Get-Status.
- Neptune unterstützt jetzt abschließende Header für SPARQL-Abfrageantworten, die einen Fehlercode und eine Meldung enthalten können, wenn eine Anforderung fehlschlägt, nachdem sie begonnen hat, Antwortblöcke zurückzugeben. Siehe Optionale HTTP-Trailing-Header für mehrteilige SPARQL-Antworten.
- Mit Neptune können Sie jetzt auch die Chunked-Response-Kodierung für Gremlin-Abfragen aktivieren. Wie im Fall von SPARQL haben die Antwort-Chunks abschließende Header, die einen

Fehlercode und eine Meldung enthalten können, falls ein Fehler auftritt, nachdem die Abfrage begonnen hat, Antwortblöcke zurückzugeben. Siehe Verwenden optionaler nachgestellter HTTP-Header zum Aktivieren mehrteiliger Gremlin-Antworten.

Verbesserungen in dieser Engine-Version

- Sie können jetzt die Größe von Batch-Anfragen ElasticSearch für Volltextsuchen in Gremlin angeben.
- Die Speichernutzung für SPARQL GROUP BY-Abfragen wurde verbessert.
- Es wurde ein neuer Gremlin-Abfrageoptimierer hinzugefügt, um bestimmte ungebundene Filter zu löschen.
- Die maximale Dauer, für die eine mit IAM authentifizierte WebSocket Verbindung bestehen kann, wurde von 36 Stunden auf 10 Tage erhöht.

In diesem Engine-Version behobene Fehler

- Es wurde ein Fehler behoben, bei dem Neptune den HTTP-Statuscode 500 und einen `InternalServerErrorException` ausgab, wenn Sie in einer POST-Anfrage einen uncodierten URL-Parameter sendeten. Jetzt gibt Neptune den HTTP-Statuscode „400“ und einen `BadRequestException` aus, mit der Meldung: `Failure to process the POST request parameters`.
- Es wurde ein Gremlin-Fehler behoben, bei dem ein WebSocket Verbindungsfehler nicht korrekt gemeldet wurde.
- Es wurde ein Gremlin-Fehler behoben, der dazu führte, dass sideEffects verschwanden.
- Es wurde ein Gremlin-Fehler behoben, bei dem der Volltext-Suchparameter `batchsize` nicht richtig unterstützt wurde.
- Ein Gremlin-Fehler wurde behoben, um `toV` und `fromV` individuell für jede Richtung auf `bothE` zu behandeln.
- In diesem Schritt wurde ein Gremlin-Fehler behoben, der `Edge pathType` im Schritt `hasLabel` betraf.
- Es wurde ein SPARQL-Fehler behoben, bei dem die Neuanordnung von Verknüpfungen mit statischen Bindungen nicht korrekt funktionierte.
- Es wurde ein SPARQL UPDATE LOAD-Fehler behoben, bei dem ein nicht Amazon S3-Bucket nicht korrekt gemeldet wurde.

- Es wurde ein SPARQL-Fehler behoben, bei dem ein Problem mit einem SERVICE-Knoten in einer Unterabfrage nicht korrekt gemeldet wurde.
- Es wurde ein SPARQL-Fehler behoben, bei dem Abfragen, die verschachtelte FILTER EXISTS oder FILTER NOT EXISTS Bedingungen enthielten, nicht korrekt ausgewertet wurden.
- Es wurde ein SPARQL-Fehler behoben, durch den doppelt generierte Bindungen korrekt behandelt wurden, wenn SPARQL-Dienstendpunkte über generierte Abfragen aufgerufen wurden.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.0.3.0 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprache kompatibel ist:

- Gremlin-Version: `3.4.3`
- SPARQL-Version: `1.1`

Upgrade-Pfade zum Engine-Release 1.0.3.0

Sie können jeden vorherigen Neptune Engine-Release manuell auf diese Version aktualisieren.

Wenn der `AutoMinorVersionUpgrade`-Parameter des Clusters auf `True` eingestellt ist, wird der Cluster während eines Wartungsfensters zwei bis drei Wochen nach dem Datum dieses Releases automatisch auf diese Engine-Version aufgerüstet.

Upgrade auf diesen Release

Amazon Neptune 1.0.3.0 ist jetzt allgemein verfügbar.

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.3.0 \
```

```
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.3.0 ^
    --apply-immediately
```

Updates werden auf alle Instances in einem DB-Cluster gleichzeitig angewendet. Ein Update erfordert einen Datenbankneustart auf diesen Instances. Daher kommt es zu einer Ausfallzeit von 20–30 Sekunden bis zu mehreren Minuten. Anschließend können Sie die Nutzung Ihres DB-Clusters fortsetzen.

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
> 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.
> ```
>
> Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine-Version 1.0.3.0.R3 (19.02.2021)

Ab dem 19.02.2021 wird die Engine-Version 1.0.3.0.R3 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

In diesem Engine-Version behobene Fehler

- Es wurde ein Fehler beim Massenladen von CSVs behoben, der Labels betraf, die auf # enden.
- Es wurde ein Gremlin-Fehler behoben, der dazu führen konnte, dass Ergebnisse für ein bestimmtes Muster von Abfragen, die diesen den `as()` verwenden, fehlten.
- Es wurde ein Gremlin-Fehler behoben, der zu Fehlern führen konnte, wenn der Schritt `project()` in einem anderen Schritt verschachtelt war, wie `union()`.

- Es wurde ein Gremlin-Fehler bei der Ausführung von String-Traversal in der experimentellen DFE-Engine behoben, wenn eine Terminalmethode wie `toList()` verwendet wurde.
- Es wurde ein Gremlin-Fehler behoben, durch den eine Transaktion nicht geschlossen werden konnte, wenn der Schritt `iterate()` in einer Zeichenkettenabfrage verwendet wurde.
- Es wurde ein Gremlin-Fehler behoben, der dazu führen konnte, dass Abfragen, die das Muster `is(P.gte(0))` verwendeten, unter bestimmten Bedingungen eine Ausnahme auslösten.
- Es wurde ein Gremlin-Fehler behoben, bei dem unter bestimmten Umständen `InternalFailureException` als Antwortcode festgelegt wurde, wenn ein `ConcurrentModificationException` auftrat.
- Es wurde ein Gremlin-Fehler behoben, bei dem das Aktualisieren von Edges oder Scheitelpunkten unter bestimmten Bedingungen zu einem transienten `InternalFailureException` führen konnte.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.0.3.0.R3 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprache kompatibel ist:

- Gremlin-Version: `3.4.8`
- SPARQL-Version: `1.1`

Upgrade-Pfade zum Engine-Release 1.0.3.0.R3

Ihr Cluster wird während des nächsten Wartungsfensters automatisch auf diese Patch-Version aktualisiert, wenn Sie die Modulversion `1.0.3.0` ausführen.

Sie können jeden vorherigen Neptune Engine-Release manuell auf diese Version aktualisieren.

Upgrade auf diesen Release

Amazon Neptune 1.0.3.0.R3 ist jetzt allgemein verfügbar.

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.3.0 \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.3.0 ^
    --apply-immediately
```

Updates werden auf alle Instances in einem DB-Cluster gleichzeitig angewendet. Ein Update erfordert einen Datenbankneustart auf diesen Instances. Daher kommt es zu einer Ausfallzeit von 20–30 Sekunden bis zu mehreren Minuten. Anschließend können Sie die Nutzung Ihres DB-Clusters fortsetzen.

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
>   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.
> ```
>
> Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine-Version 1.0.3.0.R2 (12.10.2020)

Ab dem 12.10.2020 wird die Engine-Version 1.0.3.0.R2 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

Verbesserungen in dieser Engine-Version

- Verbesserte Leistung für den Gremlin-Schritt `properties()`.
- Details zu `BindOp` und `MultiplexerOp` in Erklärungs- und Profilberichten hinzugefügt.

- Für SPARQL-Abfrageantworten wurde dem Content-Type-Header `charset` hinzugefügt, sodass HTTP-Clients den verwendeten Zeichensatz automatisch erkennen können.

In diesem Engine-Version behobene Fehler

- Ein SPARQL-Fehler wurde behoben, bei dem `CancellationException` nicht behandelt wurde.
- Es wurde ein SPARQL-Fehler behoben, bei dem Abfragen, die verschachtelte optionale Elemente enthielten, nicht korrekt funktionierten.
- Es wurde ein SPARQL-Fehler in LOAD behoben, bei dem ein `ConcurrentModificationException` dazu führen konnte, dass eine Abfrage sich aufhängte.
- Es wurde ein SPARQL-Fehler behoben, der verhinderte, dass Abfrageantworten gzip-komprimiert wurden.
- Ein Gremlin-Fehler im `groupBy()`-Schritt wurde behoben.
- Ein Gremlin-Fehler im Zusammenhang mit der Verwendung eines `aggregate()`-Schrittes innerhalb eines `local()`-Schrittes wurde behoben.
- Ein Gremlin-Fehler im Zusammenhang mit der Verwendung von `bothE()` gefolgt von einem Prädikat, das Aggregatwerte verwendet, wurde behoben.
- Ein Gremlin-Fehler im Zusammenhang mit der Verwendung des `bothE()`-Schrittes mit dem `repeat()`-Schritt wurde behoben.
- Ein potenzielles Gremlin-Speicherleck im Zusammenhang mit dem `both()`-Schritt wurde behoben.
- Es wurde ein Fehler behoben, bei dem Anforderungsmetriken fehlten, weil ein Endpunkt, der auf ‚/' endete, nicht korrekt behandelt wurde.
- Es wurde ein Fehler behoben, der ein `ThrottlingException` auslösen konnte, auch wenn die Warteschlange nicht voll ist.
- Es wurde ein Fehler beim Abrufen des Ladestatus behoben, wenn ein Ladevorgang aus einem Grund wie `LOAD_DATA_FAILED_DUE_TO_FEED_MODIFIED_OR_DELETE` fehlschlägt.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.0.3.0.R2 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprache kompatibel ist:

- Gremlin-Version: `3.4.3`
- SPARQL-Version: `1.1`

Upgrade-Pfade zum Engine-Release 1.0.3.0.R2

Sie können jeden vorherigen Neptune Engine-Release manuell auf diese Version aktualisieren.

Wenn der `AutoMinorVersionUpgrade`-Parameter des Clusters auf `True` eingestellt ist, wird der Cluster während eines Wartungsfensters zwei bis drei Wochen nach dem Datum dieses Releases automatisch auf diese Engine-Version aufgerüstet.

Upgrade auf diesen Release

Amazon Neptune 1.0.3.0.R2 ist jetzt allgemein verfügbar.

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.3.0 \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.3.0 ^
    --apply-immediately
```

Updates werden auf alle Instances in einem DB-Cluster gleichzeitig angewendet. Ein Update erfordert einen Datenbankneustart auf diesen Instances. Daher kommt es zu einer Ausfallzeit von 20–30 Sekunden bis zu mehreren Minuten. Anschließend können Sie die Nutzung Ihres DB-Clusters fortsetzen.

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein

Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
> 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.
> ```

Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine-Version 1.0.2.2 09.03.2020

Ab dem 09.03.2023 wird die Engine-Version 1.0.2.2 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

Nachfolgende Patch-Veröffentlichungen für dieses Version

- Release: 1.0.2.2.R2 (02.04.2020)
- Release: 1.0.2.2.R3 (22.07.2020)
- Release: 1.0.2.2.R4 (23.07.2020)
- Release: 1.0.2.2.R5 (12.10.2020)
- Release: 1.0.2.2.R6 (19.02.2021)

Verbesserungen in dieser Engine-Version

- Der Status-API wurden Informationen zu Transaktionen hinzugefügt, die zurückgesetzt werden. Siehe Instance-Status.
- Die Version von Apache wurde TinkerPop auf 3.4.3 aktualisiert.

  Version 3.4.3 ist abwärtskompatibel mit der von Neptune unterstützten Vorgängerversion (3.4.1). Mit ihr wird eine geringfügige Änderung des Verhaltens eingeführt: Gremlin gibt keinen Fehler mehr zurück, wenn Sie versuchen, eine Sitzung zu schließen, die nicht vorhanden ist (siehe Fehler beim Schließen von nicht vorhandenen Sitzungen verhindern).
- Es wurden Leistungsengpässe bei der Ausführung von Schritten der Gremlin-Volltextsuche beseitigt.

In diesem Engine-Version behobene Fehler

- Es wurde ein SPARQL-Fehler bei der Behandlung von leeren Graphenmustern in Abfragen behoben.
- Es wurde ein SPARQL-Fehler bei der Behandlung von unkodierten Semikolons in URL-kodierten Abfragen behoben.
- Es wurde ein Gremlin-Fehler bei der Behandlung von wiederholten Vertices (Eckpunkten) im `Union`-Schritt behoben.
- Es wurde ein Gremlin-Fehler behoben, der dazu führte, dass einige Abfragen mit einem `.simplePath()` oder `.cyclicPath()` innerhalb einer `.repeat()` fehlerhaften Ergebnisses zurückgaben.
- Es wurde ein Gremlin-Fehler behoben, der dazu führte, dass `.project()` falsche Ergebnisse zurückgab, wenn sein untergeordnetes Traversal keine Lösungen zurückgab.
- Es wurde ein Gremlin-Fehler behoben, bei dem Fehler aus Lese- und Schreibkonflikten eine `InternalFailureException` statt eine `ConcurrentModificationException` auslösten.
- Es wurde ein Gremlin-Fehler behoben, der `.group().by(...).by(values("property"))`-Fehler verursachte.
- Gremlin-Fehler in der Profilausgabe für Schritte wurden behoben. full-text-search
- Ein Ressourcenleck in Gremlin-Sitzungen wurde behoben.
- Es wurde ein Fehler behoben, der in einigen Fällen verhinderte, dass die Status-API die richtige bestellbare Version meldete.
- Es wurde ein Fehler beim Massenladen behoben, der es ermöglichte, eine URL zu einem anderen Ort als als Quelle in einer Massenladeanforderung zu verwenden.
- Es wurde ein Massen-Loader-Fehler im ausführlichen Ladestatus behoben.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.0.2.2 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprache kompatibel ist:

- Gremlin-Version: `3.4.3`
- SPARQL-Version: `1.1`

Upgrade-Pfade zum Engine-Release 1.0.2.2

Sie können jeden vorherigen Neptune Engine-Release manuell auf diese Version aktualisieren.

Wenn der `AutoMinorVersionUpgrade`-Parameter des Clusters auf `True` eingestellt ist, wird der Cluster während eines Wartungsfensters zwei bis drei Wochen nach dem Datum dieses Releases automatisch auf diese Engine-Version aufgerüstet.

Upgrade auf diesen Release

Amazon Neptune 1.0.2.2 ist jetzt allgemein verfügbar.

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.2 \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.2 ^
    --apply-immediately
```

Updates werden auf alle Instances in einem DB-Cluster gleichzeitig angewendet. Ein Update erfordert einen Datenbankneustart auf diesen Instances. Daher kommt es zu einer Ausfallzeit von 20–30 Sekunden bis zu mehreren Minuten. Anschließend können Sie die Nutzung Ihres DB-Clusters fortsetzen.

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein

Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
> 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.
> ```

Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine-Version 1.0.2.2.R6 (19.02.2021)

Ab dem 19.02.2020 wird die Engine-Version 1.0.2.2.R6 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

In diesem Engine-Version behobene Fehler

- Es wurde ein Gremlin-Fehler behoben, bei dem unter bestimmten Umständen `InternalFailureException` als Antwortcode festgelegt wurde, wenn ein `ConcurrentModificationException` auftrat.
- Es wurde ein Gremlin-Fehler behoben, bei dem das Aktualisieren von Edges oder Scheitelpunkten unter bestimmten Bedingungen zu einem transienten `InternalFailureException` führen konnte.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.0.2.2.R6 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprache kompatibel ist:

- Gremlin-Version: `3.4.8`
- SPARQL-Version: `1.1`

Upgrade-Pfade zum Engine-Release 1.0.2.2.R6

Ihr Cluster wird während des nächsten Wartungsfensters automatisch auf diese Patch-Version aktualisiert, wenn Sie die Modulversion `1.0.2.2` ausführen.

Sie können jeden vorherigen Neptune Engine-Release manuell auf diese Version aktualisieren.

Upgrade auf diesen Release

Amazon Neptune 1.0.2.2.R6 ist jetzt allgemein verfügbar.

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.2 \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.2 ^
    --apply-immediately
```

Updates werden auf alle Instances in einem DB-Cluster gleichzeitig angewendet. Ein Update erfordert einen Datenbankneustart auf diesen Instances. Daher kommt es zu einer Ausfallzeit von 20–30 Sekunden bis zu mehreren Minuten. Anschließend können Sie die Nutzung Ihres DB-Clusters fortsetzen.

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue

Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
> 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.
> ```
>
> Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine-Version 1.0.2.2.R5 (12.10.2020)

Ab dem 12.10.2020 wird die Engine-Version 1.0.2.2.R5 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

Verbesserungen in dieser Engine-Version

- Verbesserte Leistung für den Gremlin-Schritt `properties()`.
- Details zu `BindOp` und `MultiplexerOp` in Erklärungs- und Profilberichten hinzugefügt.
- Für SPARQL-Abfrageantworten wurde dem Content-Type-Header `charset` hinzugefügt, sodass HTTP-Clients den verwendeten Zeichensatz automatisch erkennen können.

In diesem Engine-Version behobene Fehler

- Ein SPARQL-Fehler wurde behoben, bei dem `CancellationException` nicht behandelt wurde.
- Es wurde ein SPARQL-Fehler behoben, bei dem Abfragen, die verschachtelte optionale Elemente enthielten, nicht korrekt funktionierten.
- Es wurde ein SPARQL-Fehler in LOAD behoben, bei dem ein `ConcurrentModificationException` dazu führen konnte, dass eine Abfrage sich aufhängte.
- Es wurde ein SPARQL-Fehler behoben, der verhinderte, dass Abfrageantworten gzip-komprimiert wurden.
- Ein Gremlin-Fehler im `groupBy()`-Schritt wurde behoben.
- Ein Gremlin-Fehler im Zusammenhang mit der Verwendung eines `aggregate()`-Schrittes innerhalb eines `local()`-Schrittes wurde behoben.
- Ein Gremlin-Fehler im Zusammenhang mit der Verwendung von `bothE()` gefolgt von einem Prädikat, das Aggregatwerte verwendet, wurde behoben.
- Ein Gremlin-Fehler im Zusammenhang mit der Verwendung des `bothE()`-Schrittes mit dem `repeat()`-Schritt wurde behoben.
- Ein potenzielles Gremlin-Speicherleck im Zusammenhang mit dem `both()`-Schritt wurde behoben.
- Es wurde ein Fehler behoben, bei dem Anforderungsmetriken fehlten, weil ein Endpunkt, der auf ‚/' endete, nicht korrekt behandelt wurde.
- Es wurde ein Fehler behoben, der ein `ThrottlingException` auslösen konnte, auch wenn die Warteschlange nicht voll ist.

- Es wurde ein Fehler beim Abrufen des Ladestatus behoben, wenn ein Ladevorgang aus einem Grund wie `LOAD_DATA_FAILED_DUE_TO_FEED_MODIFIED_OR_DELETE` fehlschlägt.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.0.2.2.R5 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprache kompatibel ist:

- Gremlin-Version: `3.4.3`
- SPARQL-Version: `1.1`

Upgrade-Pfade zum Engine-Release 1.0.2.2.R5

Ihr Cluster wird während des nächsten Wartungsfensters automatisch auf diese Patch-Version aktualisiert, wenn Sie die Modulversion `1.0.2.2` ausführen.

Sie können jeden vorherigen Neptune Engine-Release manuell auf diese Version aktualisieren.

Upgrade auf diesen Release

Amazon Neptune 1.0.2.2.R5 ist jetzt allgemein verfügbar.

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.2 \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.2 ^
    --apply-immediately
```

Updates werden auf alle Instances in einem DB-Cluster gleichzeitig angewendet. Ein Update erfordert einen Datenbankneustart auf diesen Instances. Daher kommt es zu einer Ausfallzeit von 20–30 Sekunden bis zu mehreren Minuten. Anschließend können Sie die Nutzung Ihres DB-Clusters fortsetzen.

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:

```
  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.
```

Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine-Version 1.0.2.2.R4 (23.07.2020)

Ab dem 23.07.2020 wird die Engine-Version 1.0.2.2.R4 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

Verbesserungen in dieser Engine-Version

- Die Speichernutzung wurde verbessert, indem ungenutzter Speicher häufiger für das Betriebssystem freigegeben wurde.
- Außerdem wurde die Speichernutzung für SPARQL GROUP BY-Abfragen verbessert.
- Die maximale Dauer, die eine mit IAM authentifizierte WebSocket Verbindung bestehen kann, wurde von 36 Stunden auf 10 Tage erhöht.
- Die `BufferCacheHitRatio` CloudWatch Metrik wurde hinzugefügt, die bei der Diagnose der Abfragelatenz und der Optimierung von Instance-Typen nützlich sein kann. Siehe Neptune-Metriken.

In diesem Engine-Version behobene Fehler

- Ein Fehler beim Schließen inaktiver oder abgelaufener WebSocket IAM-Verbindungen wurde behoben. Neptune sendet jetzt einen geschlossenen Frame, bevor die Verbindung geschlossen wird.

- Ein SPARQL-Fehler bei der Auswertung von Abfragen, die verschachtelte FILTER EXISTS and/or FILTER NOT EXISTS-Bedingungen enthielten, wurde behoben.
- Es wurde ein Fehler beim Beenden von SPARQL-Abfragen behoben, der unter bestimmten extremen Bedingungen zu blockierten Threads auf dem Server führte.
- In diesem Schritt `hasLabel` wurde ein Gremlin-Fehler behoben, der Edge PathType betraf.
- Ein Gremlin-Fehler wurde behoben, um `toV` und `fromV` individuell für jede Richtung auf `bothE` zu behandeln.
- Es wurde ein Gremlin-Fehler behoben, der dazu führte, dass sideEffects verschwanden.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.0.2.2.R4 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprache kompatibel ist:

- Gremlin-Version: `3.4.3`
- SPARQL-Version: `1.1`

Upgrade-Pfade zum Engine-Release 1.0.2.2.R4

Ihr Cluster wird während des nächsten Wartungsfensters automatisch auf diese Patch-Version aktualisiert, wenn Sie die Modulversion `1.0.2.2` ausführen.

Sie können jeden vorherigen Neptune Engine-Release manuell auf diese Version aktualisieren.

Upgrade auf diesen Release

Amazon Neptune 1.0.2.2.R4 ist jetzt allgemein verfügbar.

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
```

```
    --engine-version 1.0.2.2 \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.2 ^
    --apply-immediately
```

Updates werden auf alle Instances in einem DB-Cluster gleichzeitig angewendet. Ein Update erfordert einen Datenbankneustart auf diesen Instances. Daher kommt es zu einer Ausfallzeit von 20–30 Sekunden bis zu mehreren Minuten. Anschließend können Sie die Nutzung Ihres DB-Clusters fortsetzen.

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
>   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.
> ```
>
> Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine-Version 1.0.2.2.R3 (22.07.2020)

Die Engine-Version 1.0.2.2.R3 wurde in den Engine-Release 1.0.2.2.R4 aufgenommen.

Amazon Neptune Engine-Version 1.0.2.2.R2 (02.04.2020)

Ab dem 02.04.2020 wird die Engine-Version 1.0.2.2.R2 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

Verbesserungen in dieser Engine-Version

- Sie können jetzt bis zu 64 Bulk-Ladeaufträge in die Warteschlange stellen, anstatt mit dem Starten eines Auftrags warten zu müssen, bis eine anderer fertig ist. Sie können mithilfe des Parameters

`dependencies` des Befehls `load` die Ausführung einer Ladeanforderung in der Warteschlange auch davon abhängig machen, ob einer oder mehrere der zuvor in die Warteschlange eingestellten Ladeaufträge erfolgreich abgeschlossen wurden. Siehe Neptune-Loader-Befehl.

- Full-text-search Die Ausgabe kann jetzt sortiert werden (siehe Parameter für die Volltextsuche).
- Es gibt jetzt einen DB-Cluster-Parameter zum Aufrufen von Neptune-Streams, und das Feature wurde aus dem Labormodus verschoben. Siehe Verwenden von Neptune-Streams.

In diesem Engine-Version behobene Fehler

- Ein stochastischer Fehler beim Serverstart, der die Instance-Erstellung verzögert hat, wurde behoben.
- Es wurde ein Optimierer-Problem behoben, bei dem `BIND`-Anweisungen in der Abfrage bewirkten, dass der Optimierer mit unselektiven Mustern in der Join-Order-Planung gestartet wurde.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.0.2.2.R2 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprache kompatibel ist:

- Gremlin-Version: `3.4.3`
- SPARQL-Version: `1.1`

Upgrade-Pfade zum Engine-Release 1.0.2.2.R2

Ihr Cluster wird während des nächsten Wartungsfensters automatisch auf diese Patch-Version aktualisiert, wenn Sie die Modulversion `1.0.2.2` ausführen.

Sie können jeden vorherigen Neptune Engine-Release manuell auf diese Version aktualisieren.

Upgrade auf diesen Release

Amazon Neptune 1.0.2.2.R2 ist jetzt allgemein verfügbar.

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.2 \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.2 ^
    --apply-immediately
```

Updates werden auf alle Instances in einem DB-Cluster gleichzeitig angewendet. Ein Update erfordert einen Datenbankneustart auf diesen Instances. Daher kommt es zu einer Ausfallzeit von 20–30 Sekunden bis zu mehreren Minuten. Anschließend können Sie die Nutzung Ihres DB-Clusters fortsetzen.

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
> 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.
> ```
>
> Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine-Version 1.0.2.1 (22.11.2019)

Nachfolgende Patch-Veröffentlichungen für dieses Version

- Release: 1.0.2.1.R6 (22.04.2020)
- Release: 1.0.2.1.R5 (22.04.2020) Diese Patch-Version wurde nicht bereitgestellt.
- Release: 1.0.2.1.R4 (20.12.2019)

- Release: 1.0.2.1.R3 (12.12.2019)
- Release: 1.0.2.1.R2 (25.11.2019)

Neue Features in dieser Engine-Version

- Durch die Integration mit dem Amazon OpenSearch Service wurden Funktionen für die Volltextsuche hinzugefügt. Siehe Neptun-Volltextsuche
- Es wurde eine Option im Labor-Modus zur Erstellung eines vierten Index (einen OSGP-Index) für eine große Anzahl von Prädikaten hinzugefügt. Siehe OSGP-Index.
- Es wurde ein Detailmodus zu SPARQL Explain hinzugefügt. Weitere Informationen hierzu finden Sie unter Verwenden von SPARQL `explain` und Ausgabe im Detailmodus.
- Zum Engine-Statusbericht wurden Informationen zum Labor-Modus hinzugefügt. Details dazu finden Sie unter Instance-Status.
- DB-Cluster-Snapshots können jetzt regionsübergreifend AWS kopiert werden. Siehe Kopieren eines Snapshots.

Verbesserungen in dieser Engine-Version

- Verbesserte Leistung bei der Handhabung einer großen Anzahl von Prädikaten.
- Verbesserte Abfrageoptimierung. Obwohl dies für Kunden völlig transparent sein sollte, empfehlen wir Ihnen, Ihre Anwendungen vor dem Upgrade zu testen, um sicherzustellen, dass sie sich wie erwartet verhalten.
- Kleinere Verbesserungen bei der Fehlerberichterstellung.
- Optimierungen für Gremlin `.project()` und `.identity()` Schritte wurden hinzugefügt.
- Optimierungen für nicht abschließende `.union()`-Gremlin-Fälle wurden hinzugefügt.
- Native Unterstützung für `.path().by()`-Gremlin-Traversalen wurde hinzugefügt.
- Native Unterstützung für Gremlin `.coalesce()` wurde hinzugefügt.
- Weitere Optimierung von Bulk-Write.
- Wir verlangen jetzt, dass HTTPS-Verbindungen mindestens TLS Version 1.2 oder höher verwenden, um zu verhindern, dass outdated/insecure Chiffren verwendet werden.

In diesem Engine-Version behobene Fehler

- Es wurde ein Fehler bei der Handhabung des `addE()`-inneren Traversalen bei Gremlin behoben.
- Es wurde Fehler bei Gremlin behoben, der durch AST-Annotationen von untergeordneten Traversalen an das übergeordnete Element verursacht wurde.
- Es wurde ein Fehler behoben, der in Gremlin auftrat, wenn `.otherV()` nach `select()` aufgerufen wurde.
- Ein Gremlin-Fehler wurde behoben, der zu Fehlern bei `.hasLabel()`-Schritten führte, wenn sie auf einen `bothE()`-Schritt folgten.
- Es wurden kleine Korrekturen für Gremlin .sum() und .project() vorgenommen.
- Es wurde ein Fehler bei der Verarbeitung von SPARQL-Abfragen behoben, denen die schließende Klammer fehlte.
- Es wurden einige kleinere Fehler bei SPARQL Explain behoben.
- Es wurde ein Fehler bei der Verarbeitung gleichzeitiger Abrufe von Ladestatus-Anforderungen behoben.
- Der für die Ausführung einiger Gremlin-Traversalen mit `.project()`-Schritten verwendete Speicher wurde reduziert.
- Es wurden numerische Vergleiche von Sonderwerten in SPARQL behoben. Siehe Einhaltung von Standards.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.0.2.1 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprache kompatibel ist:

- Gremlin-Version: `3.4.1`
- SPARQL-Version: `1.1`

Upgrade-Pfade zum Engine-Release 1.0.2.1

Sie können jeden vorherigen Neptune Engine-Release manuell auf diese Version aktualisieren.

Es wird kein automatisches Upgrade auf diese Version durchgeführt.

Upgrade auf diesen Release

Amazon Neptune 1.0.2.1 ist jetzt allgemein verfügbar.

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.1 \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.1 ^
    --apply-immediately
```

Updates werden auf alle Instances in einem DB-Cluster gleichzeitig angewendet. Ein Update erfordert einen Datenbankneustart auf diesen Instances. Daher kommt es zu einer Ausfallzeit von 20–30 Sekunden bis zu mehreren Minuten. Anschließend können Sie die Nutzung Ihres DB-Clusters fortsetzen.

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue

Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
> 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.
> ```
>
> Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine-Version 1.0.2.1.R6 (22.04.2020)

Ab dem 22.04.2020 wird die Engine-Version 1.0.2.1.R6 allgemein bereitgestellt. Bitte beachten Sie, dass es mehrere Tage dauert, bis eine neue Version in jeder Region verfügbar ist.

In diesem Engine-Version behobene Fehler

- Es wurde ein Fehler behoben, bei dem `ConcurrentModificationConflictException` und `TransactionException` nicht in einen `NeptuneGremlinException` umgewandelt wurden, was die Rückgabe von `InternalFailureException` an Kunden verursachte.
- Es wurde ein Fehler behoben, durch den der Status von Neptune als fehlerfrei gemeldet wurde, bevor der Server vollständig bereit war.
- Es wurde ein Fehler behoben, durch den Lexikon-und Benutzertransaktions-Commits nicht mehr in Ordnung waren, wenn zwei `value->id`-Zuweisungen gleichzeitig eingefügt wurden.
- Es wurde ein Fehler bei der Laststatus-Serialisierung behoben.
- Es wurde ein Fehler bei Gremlin-Sitzungen behoben.
- Es wurde ein Fehler behoben, bei dem Neptune keine Ausnahme auslöste, wenn der Server nicht gestartet wurde.
- Es wurde ein Fehler behoben, bei dem Neptune kein Web-Socket-Close-Frame sendete, bevor der Kanal geschlossen wurde.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.0.2.1.R6 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprache kompatibel ist:

- Gremlin-Version: `3.4.1`
- SPARQL-Version: `1.1`

Upgrade-Pfade zum Engine-Release 1.0.2.1.R6

Ihr Cluster wird während des nächsten Wartungsfensters automatisch auf diese Patch-Version aktualisiert, wenn Sie die Modulversion `1.0.2.1` ausführen.

Sie können jeden vorherigen Neptune Engine-Release manuell auf diese Version aktualisieren.

Upgrade auf diesen Release

Amazon Neptune 1.0.2.1.R6 ist jetzt allgemein verfügbar.

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.1 \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.1 ^
    --apply-immediately
```

Updates werden auf alle Instances in einem DB-Cluster gleichzeitig angewendet. Ein Update erfordert einen Datenbankneustart auf diesen Instances. Daher kommt es zu einer Ausfallzeit von 20–30 Sekunden bis zu mehreren Minuten. Anschließend können Sie die Nutzung Ihres DB-Clusters fortsetzen.

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue

Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
> 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.
> ```
>
> Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine-Version 1.0.2.1.R5 (22.04.2020)

Die Engine-Version 1.0.2.1.R5 wurde nie bereitgestellt.

Amazon Neptune Engine-Version 1.0.2.1.R4 (20.12.2019)

Verbesserungen in dieser Engine-Version

- Neptune versucht nun, jeden full-text-search Aufruf immer zuerst in der Ausführungspipeline zu platzieren. Dadurch wird die Anzahl der Aufrufe reduziert OpenSearch, was die Leistung erheblich verbessern kann. Siehe Full-text-search Ausführung von Abfragen.
- Neptune löst nun eine `IllegalArgumentException` aus, wenn Sie versuchen, auf eine nicht existierende Eigenschaft, einen Scheitelpunkt oder ein Edge zuzugreifen. Zuvor löste Neptune in dieser Situation eine `UnsupportedOperationException` aus.

  Wenn Sie beispielsweise versuchen, ein Edge hinzuzufügen, das auf einen nicht vorhandenen Scheitelpunkt verweist, wird nun eine `IllegalArgumentException` ausgelöst.

In diesem Engine-Version behobene Fehler

- Es wurde ein Gremlin-Fehler behoben, bei dem ein `union`-Traversal innerhalb eines `project-by` keine oder falsche Ergebnisse zurückgab.
- Es wurde ein Gremlin-Fehler behoben, der dazu führte, dass verschachtelte `.project().by()`-Schritte falsche Ergebnisse zurückgaben.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.0.2.1.R4 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprache kompatibel ist:

- Gremlin-Version: `3.4.1`
- SPARQL-Version: `1.1`

Upgrade-Pfade zum Engine-Release 1.0.2.1.R4

Sie können jeden vorherigen Neptune Engine-Release manuell auf diese Version aktualisieren.

Ein automatisches Update auf diese Version wird jedoch nicht unterstützt.

Upgrade auf diesen Release

Amazon Neptune 1.0.2.1.R4 ist jetzt allgemein verfügbar.

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.1 \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.1 ^
    --apply-immediately
```

Updates werden auf alle Instances in einem DB-Cluster gleichzeitig angewendet. Ein Update erfordert einen Datenbankneustart auf diesen Instances. Daher kommt es zu einer Ausfallzeit von 20–30 Sekunden bis zu mehreren Minuten. Anschließend können Sie die Nutzung Ihres DB-Clusters fortsetzen.

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue

Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
> 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.
> ```
>
> Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine-Version 1.0.2.1.R3 (12.12.2019)

In diesem Engine-Version behobene Fehler

- Ein Fehler wurde behoben, durch den der OSGP-Index deaktiviert wurde, obwohl das Feature in Labor-Modus mit dem `ObjectIndex`-Wert im `neptune_lab_mode`-Parameter korrekt aktiviert wurde.
- Es wurde ein Fehler behoben, der sich auf Gremlin-Abfragen mit einem `.fold()` innerhalb eines `.project().by()`-Schrittes ausgewirkt hat. Dies führte beispielsweise zur Rückgabe von unvollständigen Ergebnissen bei folgender Abfrage:

```
g.V().project("a").by(valueMap().fold())
```

- Ein Leistungsengpass bei Massenladungen von RDF-Daten wurde behoben.
- Es wurde ein Fehler behoben, der bei Aktivierung von Streams und dem Replika-Neustart durch den Primären einen Absturz auf Replikas verursacht hat.
- Es wurde ein Fehler behoben, durch den rotierte SSL-Zertifikate auf Instances nicht ohne Instance-Neustart abgerufen wurden.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.0.2.1.R3 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprache kompatibel ist:

- Gremlin-Version: `3.4.1`
- SPARQL-Version: `1.1`

Upgrade-Pfade zum Engine-Release 1.0.2.1.R3

Sie können jeden vorherigen Neptune Engine-Release manuell auf diese Version aktualisieren.

Ein automatisches Update auf diese Version wird jedoch nicht unterstützt.

Upgrade auf diesen Release

Amazon Neptune 1.0.2.1.R3 ist jetzt allgemein verfügbar.

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.1 \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.1 ^
    --apply-immediately
```

Updates werden auf alle Instances in einem DB-Cluster gleichzeitig angewendet. Ein Update erfordert einen Datenbankneustart auf diesen Instances. Daher kommt es zu einer Ausfallzeit von 20–30 Sekunden bis zu mehreren Minuten. Anschließend können Sie die Nutzung Ihres DB-Clusters fortsetzen.

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
> 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.
> ```
>
> Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine-Version 1.0.2.1.R2 (25.11.2019)

In diesem Engine-Version behobene Fehler

- Es wurde ein Fehler behoben, der alle `project().by()`-Abfragen mit Nicht-Roundrobin-By-Traversalen und Nicht-`path()`-By-Traversalen betrifft.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.0.2.1.R2 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprache kompatibel ist:

- Gremlin-Version: `3.4.1`
- SPARQL-Version: `1.1`

Upgrade-Pfade zum Engine-Release 1.0.2.1.R2

Sie können jeden vorherigen Neptune Engine-Release manuell auf diese Version aktualisieren.

Ein automatisches Update auf diese Version wird jedoch nicht unterstützt.

Upgrade auf diesen Release

Amazon Neptune 1.0.2.1.R2 ist jetzt allgemein verfügbar.

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.1 \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.1 ^
    --apply-immediately
```

Updates werden auf alle Instances in einem DB-Cluster gleichzeitig angewendet. Ein Update erfordert einen Datenbankneustart auf diesen Instances. Daher kommt es zu einer Ausfallzeit von 20–30 Sekunden bis zu mehreren Minuten. Anschließend können Sie die Nutzung Ihres DB-Clusters fortsetzen.

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune

einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
>   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.
> ```
>
> Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine-Version 1.0.2.0 (08.11.2019)

WICHTIG: DIESE ENGINE-VERSION IST INZWISCHEN VERALTET

Ab dem 19.05.2020 werden keine neuen Instances erstellt, die diese Engine-Version verwenden.

Diese Engine-Version wird jetzt durch Version 1.0.2.1 ersetzt, die alle Bugfixes in dieser Version sowie zusätzliche Funktionen wie die Integration der Volltextsuche, OSGP-Indexunterstützung und regionsübergreifendes Kopieren von Datenbank-Snapshot-Clustern enthält. AWS

Ab dem 1. Juni 2020 aktualisiert Neptune automatisch alle Cluster, die auf dieser Engine-Version ausgeführt werden, während des nächsten Wartungsfensters auf den neuesten Patch der Version 1.0.2.1. Sie können die Aktualisierung vorher manuell ausführen, wie hierbeschrieben.

Wenn Sie Probleme mit dem Upgrade haben, kontaktieren Sie uns bitte über den AWS Support oder die AWS Entwicklerforen.

Nachfolgende Patch-Veröffentlichungen für dieses Version

- Release: 1.0.2.0.R3 (05.05.2020)
- Release: 1.0.2.0.R2 (21.11.2019)

Neue Features in dieser Engine-Version

Zusätzlich zu Wartungs-Updates werden in dieser Version neue Features hinzugefügt, um mehr als eine Engine-Version gleichzeitig zu unterstützen (siehe Warten eines Amazon-Neptune-DB-Clusters).

Infolgedessen hat sich die Nummerierung der Engine-Releases geändert (siehe Versionsnummerierung vor Engine-Version 1.3.0.0).

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.0.2.0 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprache kompatibel ist:

- Gremlin-Version: `3.4.1`
- SPARQL-Version: `1.1`

Upgrade-Pfade zum Engine-Release 1.0.2.0

Sie können jeden vorherigen Neptune Engine-Release manuell auf diese Version aktualisieren.

Es wird kein automatisches Upgrade auf diese Version durchgeführt.

Upgrade auf diesen Release

Amazon Neptune 1.0.2.0 ist jetzt allgemein verfügbar.

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
```

```
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.0 \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.0 ^
    --apply-immediately
```

Updates werden auf alle Instances in einem DB-Cluster gleichzeitig angewendet. Ein Update erfordert einen Datenbankneustart auf diesen Instances. Daher kommt es zu einer Ausfallzeit von 20–30 Sekunden bis zu mehreren Minuten. Anschließend können Sie die Nutzung Ihres DB-Clusters fortsetzen.

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
>   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.
> ```
>
> Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine-Version 1.0.2.0.R3 (05.05.2020)

WICHTIG: DIESE ENGINE-VERSION IST INZWISCHEN VERALTET

Ab dem 19.05.2020 werden keine neuen Instances erstellt, die diese Engine-Version verwenden.

Diese Engine-Version wird jetzt durch Version 1.0.2.1 ersetzt, die alle Bugfixes in dieser Version sowie zusätzliche Funktionen wie die Integration der Volltextsuche, OSGP-Indexunterstützung und regionsübergreifendes Kopieren von Datenbank-Snapshot-Clustern enthält. AWS

Ab dem 1. Juni 2020 aktualisiert Neptune automatisch alle Cluster, die auf dieser Engine-Version ausgeführt werden, während des nächsten Wartungsfensters auf den neuesten Patch der Version 1.0.2.1. Sie können die Aktualisierung vorher manuell ausführen, wie hierbeschrieben.

Wenn Sie Probleme mit dem Upgrade haben, kontaktieren Sie uns bitte über den AWS Support oder die AWS Entwicklerforen.

In diesem Engine-Version behobene Fehler

- Es wurde ein Fehler behoben, bei dem `ConcurrentModificationConflictException` und `TransactionException` als generische `InternalFailureExceptions` gemeldet wurden.
- Es wurden Fehler bei Zustandsprüfungen behoben, die häufige Neustarts des Servers während des Startvorgangs verursachten.
- Es wurde ein Fehler behoben, bei dem Daten auf Replikaten nicht sichtbar waren, da Commits unter bestimmten Bedingungen nicht in Ordnung waren.
- Es wurde ein Fehler bei der Laststatus-Serialisierung behoben, bei dem das Laden aufgrund fehlender Amazon S3-Zugriffsberechtigungen fehlschlug.
- Ein Ressourcenleck in Gremlin-Sitzungen wurde behoben.
- Es wurde ein Fehler bei Zustandsprüfungen behoben, bei dem der Status „nicht in Ordnung“ beim Start von Komponenten ausgeblendet wurde, die die IAM-Authentifizierung verwalten.
- Es wurde ein Fehler behoben, bei dem Neptune vor dem WebSocket Schließen des Kanals keinen geschlossenen Frame senden konnte.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.0.2.0.R3 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprache kompatibel ist:

- Gremlin-Version: `3.4.1`
- SPARQL-Version: `1.1`

Upgrade-Pfade zum Engine-Release 1.0.2.0.R3

Ihr Cluster wird während des nächsten Wartungsfensters automatisch auf diese Patch-Version aktualisiert, wenn Sie die Modulversion `1.0.2.0` ausführen.

Sie können jeden vorherigen Neptune Engine-Release manuell auf diese Version aktualisieren.

Upgrade auf diesen Release

Amazon Neptune 1.0.2.0.R3 ist jetzt allgemein verfügbar.

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.0 \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.0 ^
    --apply-immediately
```

Updates werden auf alle Instances in einem DB-Cluster gleichzeitig angewendet. Ein Update erfordert einen Datenbankneustart auf diesen Instances. Daher kommt es zu einer Ausfallzeit von 20–30 Sekunden bis zu mehreren Minuten. Anschließend können Sie die Nutzung Ihres DB-Clusters fortsetzen.

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
> 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.
> ```
>
> Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine-Version 1.0.2.0.R2 (21.11.2019)

WICHTIG: DIESE ENGINE-VERSION IST INZWISCHEN VERALTET

Ab dem 19.05.2020 werden keine neuen Instances erstellt, die diese Engine-Version verwenden.

Diese Engine-Version wird jetzt durch Version 1.0.2.1 ersetzt, die alle Bugfixes in dieser Version sowie zusätzliche Funktionen wie die Integration der Volltextsuche, OSGP-Indexunterstützung und regionsübergreifendes Kopieren von Datenbank-Snapshot-Clustern enthält. AWS

Ab dem 1. Juni 2020 aktualisiert Neptune automatisch alle Cluster, die auf dieser Engine-Version ausgeführt werden, während des nächsten Wartungsfensters auf den neuesten Patch der Version 1.0.2.1. Sie können die Aktualisierung vorher manuell ausführen, wie hierbeschrieben.

Wenn Sie Probleme mit dem Upgrade haben, kontaktieren Sie uns bitte über den AWS Support oder die AWS Entwicklerforen.

In diesem Engine-Version behobene Fehler

- Die Caching-Strategie für geänderte Seiten auf dem Server wurde verbessert, sodass `FreeableMemory` schneller wiederhergestellt wird, wenn der Server in einen niedrigen Speicherstatus wechselt.
- Es wurde ein Fehler behoben, der zu einer Race-Condition und einem Absturz führen konnte, wenn viele and/or Ladestatus-Startladeanfragen gleichzeitig auf dem Server verarbeitet wurden.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.0.2.0.R2 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprache kompatibel ist:

- Gremlin-Version: `3.4.1`
- SPARQL-Version: `1.1`

Upgrade-Pfade zum Engine-Release 1.0.2.0.R2

Sie können jeden vorherigen Neptune Engine-Release manuell auf diese Version aktualisieren.

Ein automatisches Update auf diese Version wird jedoch nicht unterstützt.

Upgrade auf diesen Release

Amazon Neptune 1.0.2.0.R2 ist jetzt allgemein verfügbar.

Wenn auf einem DB-Cluster eine Engine-Version ausgeführt wird, für die es einen Upgrade-Pfad zu dieser Version gibt, kann sie jetzt aktualisiert werden. Sie können jeden geeigneten Cluster mithilfe der DB-Cluster-Operationen auf der Konsole oder mithilfe des SDK aktualisieren. Mit dem folgenden CLI-Befehl wird ein geeignetes Cluster sofort aktualisiert:

Für Linux, OS X oder Unix:

```
aws neptune modify-db-cluster \
    --db-cluster-identifier (your-neptune-cluster) \
    --engine-version 1.0.2.0 \
    --apply-immediately
```

Für Windows:

```
aws neptune modify-db-cluster ^
    --db-cluster-identifier (your-neptune-cluster) ^
    --engine-version 1.0.2.0 ^
    --apply-immediately
```

Updates werden auf alle Instances in einem DB-Cluster gleichzeitig angewendet. Ein Update erfordert einen Datenbankneustart auf diesen Instances. Daher kommt es zu einer Ausfallzeit von 20–30 Sekunden bis zu mehreren Minuten. Anschließend können Sie die Nutzung Ihres DB-Clusters fortsetzen.

Testen Sie immer vor dem Upgrade

Wenn eine neue Haupt- oder Nebenversion der Neptune-Engine veröffentlicht wird, testen Sie Ihre Neptune-Anwendungen immer zuerst dafür, bevor Sie sie dazu aktualisieren. Selbst ein Nebenversions-Upgrade könnte neue Features oder Verhaltensweisen einführen, die sich auf Ihren Code auswirken können.

Vergleichen Sie zunächst die Seiten mit den Versionshinweisen Ihrer aktuellen Version mit denen der Zielversion, um festzustellen, ob es Änderungen an den Versionen der Abfragesprache oder andere wichtige Änderungen geben wird.

Die beste Methode, eine neue Version zu testen, bevor Sie Ihren Produktions-DB-Cluster aktualisieren, besteht darin, den Produktions-Cluster zu klonen, so dass auf dem Klon die neue

Engine-Version ausgeführt wird. Sie können dann Abfragen auf dem Klon ausführen, ohne dass der Produktions-DB-Cluster davon betroffen wird.

Erstellen Sie vor einem Upgrade immer einen manuellen Snapshot

Bevor Sie ein Upgrade durchführen, wird dringend empfohlen, immer einen manuellen Snapshot Ihres DB-Clusters zu erstellen. Ein automatischer Snapshot bietet nur kurzfristigen Schutz, wohingegen ein manueller Snapshot verfügbar bleibt, bis Sie ihn explizit löschen.

In bestimmten Fällen erstellt Neptune im Rahmen des Upgrade-Prozesses einen manuellen Snapshot für Sie, aber Sie sollten sich nicht darauf verlassen und in jedem Fall Ihren eigenen manuellen Snapshot erstellen.

Wenn Sie sicher sind, dass Sie Ihren DB-Cluster nicht auf den Zustand vor dem Upgrade zurücksetzen müssen, können Sie den manuellen Snapshot, den Sie selbst erstellt haben, sowie den manuellen Snapshot, den Neptune möglicherweise erstellt hat, explizit löschen. Wenn Neptune einen manuellen Snapshot erstellt, hat dieser einen Namen, der mit `preupgrade` beginnt, gefolgt vom Namen Ihres DB-Clusters, der Quell-Engine-Version, der Ziel-Engine-Version und dem Datum.

> **Note**
>
> Wenn Sie versuchen, ein Upgrade durchzuführen, während eine ausstehende Aktion ausgeführt wird, kann ein Fehler wie der folgende auftreten:
>
> ```
> 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.
> ```
>
> Wenn dieser Fehler auftritt, warten Sie, bis die ausstehende Aktion abgeschlossen ist, oder starten Sie sofort ein Wartungsfenster, damit das vorherige Upgrade abgeschlossen werden kann.

Weitere Informationen zum Upgraden Ihrer Engine-Version finden Sie unter Warten eines Amazon-Neptune-DB-Clusters. Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Amazon Neptune Engine-Version 1.0.1.2 (10.06.2020)

WICHTIG: DIESE ENGINE-VERSION IST INZWISCHEN VERALTET

Ab dem 27.04.2021 werden keine neuen Instances erstellt, die diese Engine-Version verwenden.

Verbesserungen in dieser Engine-Version

- Neptune löst nun eine `IllegalArgumentException` aus, wenn Sie versuchen, auf eine nicht existierende Eigenschaft, einen Scheitelpunkt oder ein Edge zuzugreifen. Zuvor löste Neptune in dieser Situation eine `UnsupportedOperationException` aus.

  Wenn Sie beispielsweise versuchen, ein Edge hinzuzufügen, das auf einen nicht vorhandenen Scheitelpunkt verweist, wird nun eine `IllegalArgumentException` ausgelöst.

In diesem Engine-Version behobene Fehler

- Es wurde ein Fehler behoben, durch den Lexikon-und Benutzertransaktions-Commits nicht mehr in Ordnung waren, wenn zwei `value->id`-Zuweisungen gleichzeitig eingefügt wurden.
- Es wurde ein Fehler bei der Laststatus-Serialisierung behoben.
- Ein stochastischer Fehler beim Serverstart, der die Instance-Erstellung verzögert hat, wurde behoben.
- Ein Cursor-Leck wurde behoben.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.0.1.2 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprache kompatibel ist:

- Gremlin-Version: `3.4.1`
- SPARQL-Version: `1.1`

Amazon Neptune Engine-Version 1.0.1.1 (26.06.2020)

WICHTIG: DIESE ENGINE-VERSION IST INZWISCHEN VERALTET

Ab dem 27.04.2021 werden keine neuen Instances erstellt, die diese Engine-Version verwenden.

In diesem Engine-Version behobene Fehler

- Es wurde ein Fehler behoben, bei dem Commits nicht in der richtigen Reihenfolge waren, wenn sie gleichzeitig eingefügt wurden.
- Es wurde ein Fehler bei der Laststatus-Serialisierung behoben.
- Ein stochastischer Fehler beim Serverstart, der die Instance-Erstellung verzögert hat, wurde behoben.
- Ein Speicherleck wurde behoben.

In dieser Version unterstützte Versionen in Abfragesprache

Bevor Sie einen DB-Cluster auf Version 1.0.1.1 aktualisieren, stellen Sie sicher, dass Ihr Projekt mit den folgenden Versionen in Abfragesprache kompatibel ist:

- Gremlin-Version: `3.3.2`
- SPARQL-Version: `1.1`

Amazon Neptune Engine-Version 1.0.1.0 (02.07.2019)

WICHTIG: DIESE ENGINE-VERSION IST INZWISCHEN VERALTET

Ab dem 27.04.2021 werden keine neuen Instances erstellt, die diese Engine-Version verwenden.

Amazon Neptune Engine-Updates 31.10.2019

Version: 1.0.1.0.200502.0

WICHTIG: DIESE ENGINE-VERSION IST INZWISCHEN VERALTET

Ab dem 27.04.2021 werden keine neuen Instances erstellt, die diese Engine-Version verwenden.

In diesem Engine-Version behobene Fehler

- Es wurde ein Gremlin-Fehler in der Serialisierung der Antwort des Schrittes `tree()` behoben, wenn sich Clients unter Verwendung von `traversal().withRemote(...)` (also mit GLV-Bytecode) mit Neptune verbinden.

  Diese Version behebt ein Problem, bei dem Clients, die sich unter Verwendung von `traversal().withRemote(...)` mit Neptune verbunden haben, eine ungültige Antwort auf Gremlin-Abfragen erhalten haben, die einen `tree()`-Schritt enthielten.
- Es wurde ein SPARQL-Fehler in `DELETE WHERE LIMIT`-Abfragen behoben, bei dem der Abfragebeendungsprozess aufgrund einer Racebedingung abstürzte, wodurch die Abfrage zu einem Timeout führte.

Amazon Neptune Engine-Updates 15.10.2019

Version: 1.0.1.0.200463.0

WICHTIG: DIESE ENGINE-VERSION IST INZWISCHEN VERALTET

Ab dem 27.04.2021 werden keine neuen Instances erstellt, die diese Engine-Version verwenden.

Neue Features in dieser Engine-Version

- Eine Explain/Profile Gremlin-Funktion wurde hinzugefügt (siehe`Analysieren der Ausführung Neptune Neptun-Abfragen mit Gremlin Explain`).
- Hinzufügung von Unterstützung für skriptbasierte Gremlin-Sitzungen, um die Ausführung mehrerer Gremlin-Transversalen in einer einzigen Transaktion zu ermöglichen.
- Hinzufügung der Unterstützung für die SPARQL Federated Query-Erweiterung in (siehe SPARQL 1.1 Federated Query und `SPARQL-Verbundabfragen in Neptune mit der SERVICE-Erweiterung`).
- Hinzufügung einer Funktion, mit der Sie Ihre eigene `queryId` in eine Gremlin- oder SPARQL-Abfrage einfügen können, entweder über einen HTTP-URL-Parameter oder über einen SPARQL-`queryId`Abfragehinweis (siehe Einfügen einer benutzerdefinierten ID in eine Neptune-Gremlin- oder -SPARQL-Abfrage).
- Hinzufügung einer Labor-Modus-Funktion zu Neptune, mit der Sie zukünftige Features ausprobieren können, die noch nicht für die Verwendung in der Produktion bereit sind.

- Hinzufügung einer zukünftigen Neptune-Streams Funktion, die jede an Ihrer Datenbank vorgenommene Änderung zuverlässig in einem Stream protokolliert, der eine Woche lang bestehen bleibt. Diese Funktion steht nur im Labor-Modus zur Verfügung.
- Aktualisierung der formalen Semantik für gleichzeitige Transaktionen (siehe Transaktionssemantik in Neptune). Diese Funktion bietet Garantien nach Branchenstandard hinsichtlich der Gleichzeitigkeit.

  Standardmäßig ist diese Transaktionssemantik aktiviert. In einigen Szenarien kann diese Funktion das aktuelle Ladeverhalten ändern und die Ladeleistung reduzieren. Sie können den DB-Cluster-`neptune_lab_mode`-Parameter verwenden, um zur vorherigen Semantik zurückzukehren, indem Sie `ReadWriteConflictDetection=disabled` in den Parameterwert einschließen.

Verbesserungen in dieser Engine-Version

- Die Instance-Status API wurde verbessert, indem gemeldet wurde, welche Version von TinkerPop und welche Version von SPARQL die Engine verwendet.
- Verbesserung der Leistung des Gremlin-Subgraph-Operators.
- Verbesserung der Leistung der Gremlin-Antwortserialisierung.
- Verbesserung der Leistung im Gremlin Union-Schritt.
- Verbesserung der Latenz einfacher SPARQL-Abfragen.

In diesem Engine-Version behobene Fehler

- Behebung eines Gremlin-Fehlers, bei dem Timeout fälschlicherweise als interner Fehler zurückgegeben wurde.
- Behebung eines SPARQL-Fehlers, bei dem ORDER BY über einen Teil von Variablen einen internen Server-Fehler verursachte.

Amazon Neptune Engine-Updates 19.09.2019

WICHTIG: DIESE ENGINE-VERSION IST INZWISCHEN VERALTET

Ab dem 27.04.2021 werden keine neuen Instances erstellt, die diese Engine-Version verwenden.

Version: 1.0.1.0.200457.0

Amazon Neptune 1.0.1.0.200457.0 ist jetzt allgemein verfügbar. Alle neuen Neptune-DB-Cluster, einschließlich der aus Snapshots wiederhergestellten Cluster, werden in Neptune 1.0.1.0.200457.0 erstellt, sobald das Engine-Update für diese Region abgeschlossen ist.

Bestehende Cluster können sofort über die DB-Cluster-Operationen in der Konsole oder über das SDK auf diese Version aktualisiert werden. Mit dem folgenden CLI-Befehl können Sie einen DB-Cluster aktualisieren:

```
aws neptune apply-pending-maintenance-action \
    --apply-action system-update \
    --opt-in-type immediate \
    --resource-identifier arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

Updates werden auf alle Instances in einem DB-Cluster gleichzeitig angewendet. Ein Update erfordert einen Datenbankneustart auf allen Instances in einem DB-Cluster. Daher kommt es zu einer Ausfallzeit von 20–30 Sekunden bis zu mehreren Minuten. Anschließend können Sie die Nutzung Ihres DB-Clusters/Ihrer DB-Cluster fortsetzen. Sie können die Einstellungen für das Wartungszeitfenster in der Neptune-Konsole anzeigen und ändern.

Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

In diesem Engine-Version behobene Fehler

- Behebung eines Problems mit der Gremlin-Korrektheit behoben, das in der vorherigen Engine-Version (1.0.1.0.200369.0) auftrat, indem die Leistungsverbesserung bei der Verarbeitung von Bindeprädikaten, die dazu führte, entfernt wurde.
- Behebung eines SPARQL-Fehlers, der dazu führte, dass Abfragen mit `DISTINCT` und ein einzelnes in `OPTIONAL` verpacktes Muster eine `InternalServerError` generieren.

Amazon Neptune Engine-Updates 13.08.2019

WICHTIG: DIESE ENGINE-VERSION IST INZWISCHEN VERALTET

Ab dem 27.04.2021 werden keine neuen Instances erstellt, die diese Engine-Version verwenden.

Neue Features in dieser Engine-Version

- Hinzufügung einer `OVERSUBSCRIBE`-Option zu dem `parallelism`-Parameter von Neptune-Loader-Befehl, die bewirkt, dass der Neptune-Massenlader alle verfügbaren Threads und Ressourcen verwendet.

Verbesserungen in dieser Engine-Version

- Verbesserte Leistung von SPARQL-Filtern mit einfachen logischen ODER-Ausdrücken.
- Verbesserte Gremlin-Leistung bei der Bearbeitung von Bindeprädikaten.

In diesem Engine-Version behobene Fehler

- Es wurde ein SPARQL-Fehler behoben, der die Subtraktion eines `xsd:duration` von einem `xsd:date` verhinderte.
- Ein SPARQL-Fehler wurde behoben, der beim Vorliegen eines `UNION` zu unvollständigen Ergebnissen von statischem Inlining führte.
- Es wurde ein SPARQL-Fehler beim Abbruch von Abfragen behoben.
- Es wurde ein Gremlin-Fehler behoben, der während der Typenhochstufung einen Overflow verursachte.
- Es wurde ein Gremlin-Fehler bei der Verarbeitung von Scheitelpunktelementen bei `addE().from().to()`-Schritten behoben.
- Es wurde ein Gremlin-Fehler (am 26-07-2019 in veröffentlichte Engine-Version 1.0.1.0.200366.0) behoben, der die Verarbeitung von NaN-Doubles und Floats bei Single-Kardinalität-Einfügungen umfasste.
- Es wurde ein Fehler beim Generieren von Abfrageplänen mit eigenschaftsbasierten Suchen behoben.

Amazon Neptune Engine-Updates 26.07.2019

Version: 1.0.1.0.200366.0

WICHTIG: DIESE ENGINE-VERSION IST INZWISCHEN VERALTET

Ab dem 27.04.2021 werden keine neuen Instances erstellt, die diese Engine-Version verwenden.

Neue Features in dieser Engine-Version

- Auf TinkerPop 3.4.1 aktualisiert (siehe TinkerPop Upgrade-Informationen und TinkerPop3.4.1-Änderungsprotokoll).

  Diese Änderungen bieten Neptune-Kunden neue Features und Verbesserungen, wie z. B.:

  - `GraphBinary` ist jetzt als Serialisierungsformat verfügbar.
  - Ein Keep-Alive-Bug, der zu Speicherlecks im TinkerPop Java-Treiber führte, wurde behoben, sodass ein Workaround nicht mehr erforderlich ist.

  In einigen wenigen Fällen können sie sich jedoch auf den vorhandenen Gremlin-Code in Neptune auswirken. Zum Beispiel:

  - `valueMap()` gibt jetzt ein`Map<Object,Object>` anstelle eines `Map<String,Object>` zurück.
  - Inkonsistentes Verhalten des `within()`-Schritts wurde behoben, sodass er künftig konsistent mit anderen Schritten funktioniert. Zuvor mussten Typen übereinstimmen, damit Vergleiche funktionierten. Jetzt können Zahlen verschiedener Typen akkurat verglichen werden. Beispielsweise vergleicht `33` jetzt als gleich zu `33L`, was zuvor nicht der Fall war.
  - Ein Fehler in `ReducingBarrierStep` wurde behoben, sodass jetzt kein Wert zurückgegeben wird, wenn keine Elemente für die Ausgabe verfügbar sind.
  - Die Reihenfolge der geänderten `select()`-Bereiche (die Reihenfolge ist jetzt `maps`, `side-effects`, `paths`). Dadurch werden die Ergebnisse der seltenen Abfragen geändert, die `side-effects` und `select` mit demselben Schlüsselnamen für `side-effects` ebenso wie für `select` kombinieren.
  - `bulkSet()` ist jetzt Teil des GraphSON-Protokolls. Abfragen, die mit `toBulkSet()` enden, funktionieren nicht mit älteren Clients.
  - Eine Parameterisierung des `Submit()`-Schritts wurde vom 3.4-Client entfernt.

  Viele andere in TinkerPop 3.4 eingeführte Änderungen haben keinen Einfluss auf das aktuelle Verhalten von Neptune. Beispielsweise wurde Gremlin `io()` als ein Schritt zu `Traversal` hinzugefügt und ist jetzt in `Graph` veraltet, aber wurde nie in Neptune aktiviert.
- HInzufügung der Unterstützung für einzelne Kardinalitäts-Scheitelpunkteigenschaften zum Massenlader für Gremlin, um Eigenschaftsgraphdaten zu laden.
- Hinzufügung einer Option zum Überschreiben der vorhandenen Werte für eine einzelne Kardinalitäts-Eigenschaft im Massenlader.

- Hinzufügung der Möglichkeit, den Status einer Gremlin-Abfrage abzurufen und eine Gremlin-Abfrage abzubrechen.
- Hinzufügung eines Abfragehinweises für SPARQL-Abfrage-Zeitüberschreitungen.
- Hinzufügung der Möglichkeit, die Instance-Rolle in der Status-API anzuzeigen (siehe Instance-Status).
- Hinzufügung des Supports für das Klonen von Datenbanken (siehe Klonen von Datenbanken in Neptune).

Verbesserungen in dieser Engine-Version

- Verbesserung von SPARQL Query Explanation zur Anzeige von Graphvariablen aus FROM-Klauseln.
- Verbesserung der Leistung von SPARQL in Filtern, Equal-Filtern, VALUES-Klauseln und Bereichszählungen.
- Verbesserung der Leistung für die Schrittreihenfolge in Gremlin.
- Verbesserung der Leistung für `.repeat.dedup`-Traversalen in Gremlin.
- Verbesserung der Leistung von `valueMap()`- und `path().by()`-Traversalen in Gremlin.

In diesem Engine-Version behobene Fehler

- Behebung verschiedener Probleme mit SPARQL-Eigenschaftspfaden einschließlich Operationen mit benannten Graphen.
- Behebung eines Problems mit SPARQL CONSTRUCT-Abfragen, das zu Arbeitsspeicherproblemen führte.
- Behebung eines Problems mit dem Turtle RDF-Parser und lokalen Namen.
- Behebung eines Problems mit Fehlermeldungen, die Benutzern angezeigt werden.
- Behebung eines Problems mit `repeat()...drop()`-Traversalen in Gremlin.
- Behebung eines Problems mit dem `drop()`-Schritt in Gremlin.
- Behebung eines Problems mit Gremlin-Beschriftungsfiltern.
- Behebung eines Problems mit Gremlin-Abfragezeitüberschreitungen.

Amazon Neptune Engine-Updates 02.07.2019

WICHTIG: DIESE ENGINE-VERSION IST INZWISCHEN VERALTET

Ab dem 27.04.2021 werden keine neuen Instances erstellt, die diese Engine-Version verwenden.

In diesem Engine-Version behobene Fehler

- Behebung eines Fehlers, der dazu führte, dass bestimmte Muster mit einem Eigenschaftsnamen und einer Wertgrenze nicht optimiert werden.

Frühere Neptune Engine-Releases

Themen

- Amazon Neptune Engine-Updates 12.06.2019
- Amazon Neptune Engine-Updates 01.05.2019
- Amazon Neptune Engine-Updates 21.01.2019
- Amazon Neptune Engine-Updates 19.11.2018
- Amazon Neptune Engine-Updates 08.11.2018
- Amazon Neptune Engine-Updates 29.10.2018
- Amazon Neptune Engine-Updates 06.09.2018
- Amazon Neptune Engine-Updates 24.07.2018
- Amazon Neptune Engine-Updates 22.06.2018

Amazon Neptune Engine-Updates 12.06.2019

Version: 1.0.1.0.200310.0

Amazon Neptune 1.0.1.0.200310.0 ist jetzt allgemein verfügbar. Alle neuen Neptune-DB-Cluster, einschließlich der aus Snapshots wiederhergestellten Cluster, werden in Neptune 1.0.1.0.200310.0 erstellt, sobald das Engine-Update für diese Region abgeschlossen ist.

Bestehende Cluster können sofort über die DB-Cluster-Operationen in der Konsole oder über das SDK auf diese Version aktualisiert werden. Mit dem folgenden CLI-Befehl können Sie einen DB-Cluster aktualisieren, um dies sofort freizugeben:

```
aws neptune apply-pending-maintenance-action \
```

```
    --apply-action system-update \
    --opt-in-type immediate \
    --resource-identifier arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

Neptune-DB-Cluster werden während der Systemwartungsfenster automatisch auf Engine-Version 1.0.1.0.200310.0 aktualisiert. Zu welchem Zeitpunkt Updates angewendet werden, hängt von den Einstellungen für die Region und dem Wartungszeitfenster für den DB-Cluster sowie vom Typ des Updates ab.

> **Note**
>
> Das Instance-Wartungsfenster gilt nicht für Engine-Updates.

Updates werden auf alle Instances in einem DB-Cluster gleichzeitig angewendet. Ein Update erfordert einen Datenbankneustart auf allen Instances in einem DB-Cluster. Daher kommt es zu einer Ausfallzeit von 20–30 Sekunden bis zu mehreren Minuten. Anschließend können Sie die Nutzung Ihres DB-Clusters/Ihrer DB-Cluster fortsetzen. Sie können die Einstellungen für das Wartungszeitfenster in der Neptune-Konsole anzeigen und ändern.

Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Verbesserungen

- Behebung eines Fehlers, bei dem das gleichzeitige Einfügen und Löschen einer Grenze dazu führen kann, dass es mehrere Grenzen mit derselben ID gibt.
- Weitere kleinere Fehlerbehebungen und Verbesserungen.

Amazon Neptune Engine-Updates 01.05.2019

Version: 1.0.1.0.200296.0

Amazon Neptune 1.0.1.0.200296.0 ist jetzt allgemein verfügbar. Alle neuen Neptune-DB-Cluster, einschließlich der aus Snapshots wiederhergestellten Cluster, werden in Neptune 1.0.1.0.200296.0 erstellt, sobald das Engine-Update für diese Region abgeschlossen ist.

Bestehende Cluster können sofort über die DB-Cluster-Operationen in der Konsole oder über das SDK auf diese Version aktualisiert werden. Mit dem folgenden CLI-Befehl können Sie einen DB-Cluster aktualisieren, um dies sofort freizugeben:

```
aws neptune apply-pending-maintenance-action \
    --apply-action system-update \
    --opt-in-type immediate \
    --resource-identifier arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

Neptune-DB-Cluster werden während der Systemwartungsfenster automatisch auf Engine-Version 1.0.1.0.200296.0 aktualisiert. Zu welchem Zeitpunkt Updates angewendet werden, hängt von den Einstellungen für die Region und dem Wartungszeitfenster für den DB-Cluster sowie vom Typ des Updates ab.

> **Note**
>
> Das Instance-Wartungsfenster gilt nicht für Engine-Updates.

Updates werden auf alle Instances in einem DB-Cluster gleichzeitig angewendet. Ein Update erfordert einen Datenbankneustart auf allen Instances in einem DB-Cluster. Daher kommt es zu einer Ausfallzeit von 20–30 Sekunden bis zu mehreren Minuten. Anschließend können Sie die Nutzung Ihres DB-Clusters/Ihrer DB-Cluster fortsetzen. Sie können die Einstellungen für das Wartungszeitfenster in der Neptune-Konsole anzeigen und ändern.

Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Verbesserungen

- Das neue `explain`-Feature wurde den Neptune-SPARQL-Abfragen hinzugefügt, damit Sie den Abfrageplan visualisieren und ihn bei Bedarf optimieren können. Weitere Informationen finden Sie unter SPARQL `explain`.
- Die SPARQL-Leistung und Berichterstellung wurden auf verschiedene Weise verbessert.
- Die Gremlin-Leistung und das Verhalten wurden auf verschiedene Weise verbessert.
- Die Zeitüberschreitung bei langandauernden `drop( )`-Abfragen wurde verbessert.
- Die Leistung von `otherV( )`-Anfragen wurde verbessert.
- Den zurückgegebenen Informationen wurden bei der Abfrage des Neptune-Status eines DB-Clusters oder einer Instance zwei Felder hinzugefügt, nämlich die Engine-Versionsnummer und die Cluster- oder Instance-Startzeit. Siehe Instance-Status.

- Die `Get-Status`-Lader-API gibt jetzt ein `startTime`-Feld aus, in dem aufgezeichnet wird, wenn ein Auftrag gestartet wird.
- Der Lader-Befehl akzeptiert jetzt einen optionalen `parallelism`-Parameter, mit dem Sie die Anzahl der Threads beschränken können, die vom Lader verwendet werden.

Amazon Neptune Engine-Updates 21.01.2019

Version: 1.0.1.0.200267.0

Amazon Neptune 1.0.1.0.200267.0 ist jetzt allgemein verfügbar. Alle neuen Neptune-DB-Cluster, einschließlich der aus Snapshots wiederhergestellten Cluster, werden in Neptune 1.0.1.0.200267.0 erstellt, sobald das Engine-Update für diese Region abgeschlossen ist.

Bestehende Cluster können sofort über die DB-Cluster-Operationen in der Konsole oder über das SDK auf diese Version aktualisiert werden. Mit dem folgenden CLI-Befehl können Sie einen DB-Cluster aktualisieren, um dies sofort freizugeben:

```
aws neptune apply-pending-maintenance-action \
    --apply-action system-update \
    --opt-in-type immediate \
    --resource-identifier arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

Neptune-DB-Cluster werden während der Systemwartungsfenster automatisch auf Engine-Version 1.0.1.0.200267.0 aktualisiert. Zu welchem Zeitpunkt Updates angewendet werden, hängt von den Einstellungen für die Region und dem Wartungszeitfenster für den DB-Cluster sowie vom Typ des Updates ab.

> **Note**
>
> Das Instance-Wartungsfenster gilt nicht für Engine-Updates.

Updates werden auf alle Instances in einem DB-Cluster gleichzeitig angewendet. Ein Update erfordert einen Datenbankneustart auf allen Instances in einem DB-Cluster. Daher kommt es zu einer Ausfallzeit von 20–30 Sekunden bis zu mehreren Minuten. Anschließend können Sie die Nutzung Ihres DB-Clusters/Ihrer DB-Cluster fortsetzen. Sie können die Einstellungen für das Wartungszeitfenster in der Neptune-Konsole anzeigen und ändern.

Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Verbesserungen

- Neptune wartet länger (innerhalb des angegebenen Abfrage-Zeitlimits), damit Probleme behoben werden können. Dadurch tritt der Fehler "concurrent modifications exception", der vom Client behandelt werden muss, weniger häufig auf (siehe Abfragefehler).
- Es wurde ein Problem behoben, bei dem die Gremlin-Kardinalitätsdurchsetzung zu einem Neustart der Engine führte.
- Verbesserte Gremlin-Leistung für wiederholte `emit.times`-Abfragen.
- Es wurde ein Gremlin-Problem behoben, bei dem `repeat.until .emit`-Lösungen zuließ, die gefiltert werden sollten.
- Verbesserte Fehlerbehandlung in Gremlin.

Amazon Neptune Engine-Updates 19.11.2018

Version: 1.0.1.0.200264.0

Amazon Neptune 1.0.1.0.200264.0 ist jetzt allgemein verfügbar. Alle neuen Neptune-DB-Cluster, einschließlich der aus Snapshots wiederhergestellten Cluster, werden in Neptune 1.0.1.0.200264.0 erstellt, sobald das Engine-Update für diese Region abgeschlossen ist.

Bestehende Cluster können sofort über die DB-Cluster-Operationen in der Konsole oder über das SDK auf diese Version aktualisiert werden. Mit dem folgenden CLI-Befehl können Sie einen DB-Cluster aktualisieren, um dies sofort freizugeben:

```
aws neptune apply-pending-maintenance-action \
    --apply-action system-update \
    --opt-in-type immediate \
    --resource-identifier arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

Neptune-DB-Cluster werden während der Systemwartungsfenster automatisch auf Engine-Version 1.0.1.0.200264.0 aktualisiert. Zu welchem Zeitpunkt Updates angewendet werden, hängt von den Einstellungen für die Region und dem Wartungszeitfenster für den DB-Cluster sowie vom Typ des Updates ab.

> **Note**
>
> Das Instance-Wartungsfenster gilt nicht für Engine-Updates.

Updates werden auf alle Instances in einem DB-Cluster gleichzeitig angewendet. Ein Update erfordert einen Datenbankneustart auf allen Instances in einem DB-Cluster. Daher kommt es zu einer Ausfallzeit von 20–30 Sekunden bis zu mehreren Minuten. Anschließend können Sie die Nutzung Ihres DB-Clusters/Ihrer DB-Cluster fortsetzen. Sie können die Einstellungen für das Wartungszeitfenster in der Neptune-Konsole anzeigen und ändern.

Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Verbesserungen

- Unterstützung für the section called “Abfragehinweise” hinzugefügt.
- Verbesserung von Fehlermeldungen für die IAM-Authentifizierung. Weitere Informationen finden Sie unter the section called “IAM-Fehler”.
- Verbesserung der Leistung von SPARQL-Abfragen mit einer hohen Anzahl von Prädikaten.
- Verbesserung der Leistung des SPARQL-Eigenschaftspfads.
- Verbesserung der Gremlin-Leistung für bedingte Mutationen wie das Muster `fold().coalesce(unfold(), …)`, wenn sie mit den Schritten `addV()`, `addE()` und `property()` verwendet werden.
- Verbesserung der Leistung von Gremlin für die Modulationen `by()` und `sack()`.
- Verbesserung der Leistung von Gremlin für die Schritte `group()` und `groupCount()`.
- Verbesserung der Leistung von Gremlin für die Schritte `store()`, `sideEffect()` und `cap().unfold()`.
- Verbesserung der Unterstützung für Einschränkungen von Single-Kardinalität-Eigenschaften.
  - Verbesserung der Durchsetzung der Single-Kardinalität für Edge- und Eckpunkteigenschaften, die als Single-Kardinalität-Eigenschaften gekennzeichnet sind.
  - Einführung eines Fehlers, der angezeigt wird, wenn während Neptune-Ladeaufträgen zusätzliche Eigenschaftenwerte für eine vorhandene Edge-Eigenschaft angegeben werden.

Amazon Neptune Engine-Updates 08.11.2018

Version: 1.0.1.0.200258.0

Amazon Neptune 1.0.1.0.200255.0 ist jetzt allgemein verfügbar. Alle neuen Neptune-DB-Cluster, einschließlich der aus Snapshots wiederhergestellten Cluster, werden in Neptune 1.0.1.0.200258.0 erstellt, sobald das Engine-Update für diese Region abgeschlossen ist.

Bestehende Cluster können sofort über die DB-Cluster-Operationen in der Konsole oder über das SDK auf diese Version aktualisiert werden. Mit dem folgenden CLI-Befehl können Sie einen DB-Cluster aktualisieren, um dies sofort freizugeben:

```
aws neptune apply-pending-maintenance-action \
    --apply-action system-update \
    --opt-in-type immediate \
    --resource-identifier arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

Neptune-DB-Cluster werden während der Systemwartungsfenster automatisch auf Engine-Version 1.0.1.0.200258.0 aktualisiert. Zu welchem Zeitpunkt Updates angewendet werden, hängt von den Einstellungen für die Region und dem Wartungszeitfenster für den DB-Cluster sowie vom Typ des Updates ab.

> **Note**
>
> Das Instance-Wartungsfenster gilt nicht für Engine-Updates.

Updates werden auf alle Instances in einem DB-Cluster gleichzeitig angewendet. Ein Update erfordert einen Datenbankneustart auf allen Instances in einem DB-Cluster. Daher kommt es zu einer Ausfallzeit von 20–30 Sekunden bis zu mehreren Minuten. Anschließend können Sie die Nutzung Ihres DB-Clusters/Ihrer DB-Cluster fortsetzen. Sie können die Einstellungen für das Wartungszeitfenster in der Neptune-Konsole anzeigen und ändern.

Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Verbesserungen

- Unterstützung für SPARQL-Abfragehinweise hinzugefügt.
- Verbesserte Leistung für SPARQL FILTER (NOT) EXISTS-Abfragen.

- Verbesserte Leistung für SPARQL DESCRIBE-Abfragen.
- Verbesserte Leistung für das Muster „Wiederholen bis“ in Gremlin.
- Verbesserte Leistung für das Hinzufügen von Edges in Gremlin.
- Behebung eines Fehlers, bei dem SPARQL UPDATE DELETE-Abfragen in einigen Fällen fehlschlagen könnten.
- Ein Problem beim Umgang mit Timeouts auf dem WebSocket Gremlin-Server wurde behoben.

Amazon Neptune Engine-Updates 29.10.2018

Version: 1.0.1.0.200255.0

Amazon Neptune 1.0.1.0.200255.0 ist jetzt allgemein verfügbar. Alle neuen Neptune-DB-Cluster, einschließlich der aus Snapshots wiederhergestellten Cluster, werden in Neptune 1.0.1.0.200255.0 erstellt, sobald das Engine-Update für diese Region abgeschlossen ist.

Bestehende Cluster können sofort über die DB-Cluster-Operationen in der Konsole oder über das SDK auf diese Version aktualisiert werden. Mit dem folgenden CLI-Befehl können Sie einen DB-Cluster aktualisieren, um dies sofort freizugeben:

```
aws neptune apply-pending-maintenance-action \
    --apply-action system-update \
    --opt-in-type immediate \
    --resource-identifier arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

Neptune-DB-Cluster werden während der Systemwartungsfenster automatisch auf Engine-Version 1.0.1.0.200255.0 aktualisiert. Zu welchem Zeitpunkt Updates angewendet werden, hängt von den Einstellungen für die Region und dem Wartungszeitfenster für den DB-Cluster sowie vom Typ des Updates ab.

> **Note**
>
> Das Instance-Wartungsfenster gilt nicht für Engine-Updates.

Updates werden auf alle Instances in einem DB-Cluster gleichzeitig angewendet. Ein Update erfordert einen Datenbankneustart auf allen Instances in einem DB-Cluster. Daher kommt es zu einer Ausfallzeit von 20–30 Sekunden bis zu mehreren Minuten. Anschließend können Sie

die Nutzung Ihres DB-Clusters/Ihrer DB-Cluster fortsetzen. Sie können die Einstellungen für das Wartungszeitfenster in der Neptune-Konsole anzeigen und ändern.

Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Verbesserungen

- Hinzufügung von Informationen zur IAM-Authentifizierung zu Prüfprotokollen.
- Hinzufügung von Unterstützung für temporäre Anmeldeinformationen mittels IAM-Rollen und Instance-Profilen.
- Es wurde ein WebSocket Verbindungsabbruch für die IAM-Authentifizierung hinzugefügt, wenn die Erlaubnis widerrufen oder der IAM-Benutzer oder die IAM-Rolle gelöscht wird.
- Die maximale Anzahl von WebSocket Verbindungen wurde auf 60.000 pro Instanz begrenzt.
- Verbesserung der Massenladeleistung für kleinere Instance-Typen.
- Verbesserung der Leistung für Abfragen, die die Operatoren `and()`, `or()`, `not()`, `drop()` in Gremlin enthalten.
- Der NTriples Parser lehnt jetzt ungültige Objekte ab URIs, z. B. URIs die Leerzeichen enthalten.

Amazon Neptune Engine-Updates 06.09.2018

Version: 1.0.1.0.200237.0

Amazon Neptune 1.0.1.0.200237.0 ist jetzt allgemein verfügbar. Alle neuen Neptune-DB-Cluster, einschließlich der aus Snapshots wiederhergestellten Cluster, werden in Neptune 1.0.1.0.200237.0 erstellt, sobald das Engine-Update für diese Region abgeschlossen ist.

Bestehende Cluster können sofort über die DB-Cluster-Operationen in der Konsole oder über das SDK auf diese Version aktualisiert werden. Mit dem folgenden CLI-Befehl können Sie einen DB-Cluster aktualisieren, um dies sofort freizugeben:

```
aws neptune apply-pending-maintenance-action \
    --apply-action system-update \
    --opt-in-type immediate \
    --resource-identifier arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

Neptune-DB-Cluster werden während der Systemwartungsfenster automatisch auf Engine-Version 1.0.1.0.200237.0 aktualisiert. Zu welchem Zeitpunkt Updates angewendet werden, hängt von den

Einstellungen für die Region und dem Wartungszeitfenster für den DB-Cluster sowie vom Typ des Updates ab.

> **Note**
>
> Das Instance-Wartungsfenster gilt nicht für Engine-Updates.

Updates werden auf alle Instances in einem DB-Cluster gleichzeitig angewendet. Ein Update erfordert einen Datenbankneustart auf allen Instances in einem DB-Cluster. Daher kommt es zu einer Ausfallzeit von 20–30 Sekunden bis zu mehreren Minuten. Anschließend können Sie die Nutzung Ihres DB-Clusters/Ihrer DB-Cluster fortsetzen. Sie können die Einstellungen für das Wartungszeitfenster in der Neptune-Konsole anzeigen und ändern.

Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Verbesserungen

- Es wurde ein Problem behoben, bei dem `SPARQL COUNT(DISTINCT)`-Abfragen fehlgeschlagen sind.
- Es wurde ein Problem behoben, bei dem für `COUNT`-, `SUM`- und`MIN`-Abfragen mit einer `DISTINCT`-Klausel nicht mehr genügend Speicherplatz vorhanden war.
- Es wurde ein Problem behoben, bei dem Daten vom Typ `BLOB` zu einen Fehler beim Neptune-Lader führten.
- Es wurde ein Problem behoben, bei dem doppelte Einfügungen Transaktionsfehler verursachten.
- Es wurde ein Problem behoben, bei dem `DROP ALL`-Abfragen nicht abgebrochen werden konnten.
- Es wurde ein Problem behoben, bei dem Gremlin-Clients in unregelmäßigen Abständen abstürzen könnten.
- Es wurden alle Fehlercodes für Nutzlasten über 150.000 auf `HTTP 400` aktualisiert.
- Verbesserte Leistung und Genauigkeit von single-triple-pattern `COUNT()` Abfragen.
- Die Leistung von `SPARQL UNION`-Abfragen mit `BIND`-Klauseln wurde verbessert.

Amazon Neptune Engine-Updates 24.07.2018

Version: 1.0.1.0.200236.0

Amazon Neptune 1.0.1.0.200236.0 ist jetzt allgemein verfügbar. Alle neuen Neptune-DB-Cluster, einschließlich der aus Snapshots wiederhergestellten Cluster, werden in Neptune 1.0.1.0.200236.0 erstellt, sobald das Engine-Update für diese Region abgeschlossen ist.

Bestehende Cluster können sofort über die DB-Cluster-Operationen in der Konsole oder über das SDK auf diese Version aktualisiert werden. Mit dem folgenden CLI-Befehl können Sie einen DB-Cluster aktualisieren, um dies sofort freizugeben:

```
aws neptune apply-pending-maintenance-action \
    --apply-action system-update \
    --opt-in-type immediate \
    --resource-identifier arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

Neptune-DB-Cluster werden während der Systemwartungsfenster automatisch auf Engine-Version 1.0.1.0.200236.0 aktualisiert. Zu welchem Zeitpunkt Updates angewendet werden, hängt von den Einstellungen für die Region und dem Wartungszeitfenster für den DB-Cluster sowie vom Typ des Updates ab.

> **Note**
>
> Das Instance-Wartungsfenster gilt nicht für Engine-Updates.

Updates werden auf alle Instances in einem DB-Cluster gleichzeitig angewendet. Ein Update erfordert einen Datenbankneustart auf allen Instances in einem DB-Cluster. Daher kommt es zu einer Ausfallzeit von 20–30 Sekunden bis zu mehreren Minuten. Anschließend können Sie die Nutzung Ihres DB-Clusters/Ihrer DB-Cluster fortsetzen. Sie können die Einstellungen für das Wartungszeitfenster in der Neptune-Konsole anzeigen und ändern.

Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Verbesserungen

- Die SPARQL-Serialisierung wurde für den `xsd:string`-Datentyp aktualisiert. xsd:string ist im Einklang mit anderen Ausgabeformaten nun nicht mehr in der JSON-Serialisierung enthalten.
- Die Verarbeitung von `xsd:double`/`xsd:float` infinity wurde korrigiert. Die `-INF`-, `NaN`- und `INF`-Werte werden jetzt ordnungsgemäß erkannt und in allen SPARQL-Datenladerformaten, SPARQL 1.1 UPDATE und SPARQL 1.1 Query, verarbeitet.

- Behebung eines Fehlers, bei dem eine Gremlin-Abfrage mit leeren Zeichenfolgenwerten unerwartet fehlschlug.
- Behebung eines Fehlers, bei dem Gremlin `aggregate()` und `cap()` bei einem leeren Graphen unerwartet fehlschlugen.
- Behebung eines Fehlers, bei dem falsche Fehlermeldungen für Gremlin zurückgegeben werden, wenn die Kardinalitätsangabe ungültig ist, z. B. `.property(set,id,'10')` und `.property(single,id,'10')`.
- Es wurde ein Problem behoben, bei dem eine ungültige Gremlin-Syntax als zurückgegeben wurde. InternalFailureException
- Korrektur der Schreibweise von `TimeLimitExceeededException` in `TimeLimitExceededException` in Fehlermeldungen.
- Reaktion der SPARQL- und GREMLIN-Endpunkte vereinheitlicht, wenn kein Skript bereitgestellt wird.
- Fehlermeldungen für zu viele gleichzeitige Anforderungen präzisiert.

Amazon Neptune Engine-Updates 22.06.2018

Version: 1.0.1.0.200233.0

Amazon Neptune 1.0.1.0.200233.0 ist jetzt allgemein verfügbar. Alle neuen Neptune-DB-Cluster, einschließlich der aus Snapshots wiederhergestellten Cluster, werden in Neptune 1.0.1.0.200233.0 erstellt, sobald das Engine-Update für diese Region abgeschlossen ist.

Bestehende Cluster können sofort über die DB-Cluster-Operationen in der Konsole oder über das SDK auf diese Version aktualisiert werden. Mit dem folgenden CLI-Befehl können Sie einen DB-Cluster aktualisieren, um dies sofort freizugeben:

```
aws neptune apply-pending-maintenance-action \
    --apply-action system-update \
    --opt-in-type immediate \
    --resource-identifier arn:aws:rds:<region>:<account number>:<resourcetype>:<name>
```

Neptune-DB-Cluster werden während der Systemwartungsfenster automatisch auf Engine-Version 1.0.1.0.200233.0 aktualisiert. Zu welchem Zeitpunkt Updates angewendet werden, hängt von den Einstellungen für die Region und dem Wartungszeitfenster für den DB-Cluster sowie vom Typ des Updates ab.

> **Note**
>
> Das Instance-Wartungsfenster gilt nicht für Engine-Updates.

Updates werden auf alle Instances in einem DB-Cluster gleichzeitig angewendet. Ein Update erfordert einen Datenbankneustart auf allen Instances in einem DB-Cluster. Daher kommt es zu einer Ausfallzeit von 20–30 Sekunden bis zu mehreren Minuten. Anschließend können Sie die Nutzung Ihres DB-Clusters/Ihrer DB-Cluster fortsetzen. Sie können die Einstellungen für das Wartungszeitfenster in der Neptune-Konsole anzeigen und ändern.

Wenn Sie Fragen oder Bedenken haben, steht Ihnen das AWS Support-Team in den Community-Foren und über den AWS Premium-Support zur Verfügung.

Verbesserungen

- Es wurde ein Problem behoben, bei dem eine große Anzahl von Massenlade-Anforderungen in schneller Folge ausgestellt wurden, was zu einem Fehler führte.
- Es wurde ein datenabhängiges Problem behoben, bei dem eine Abfrage mit einem InternalServerError fehlschlagen konnte. Das folgende Beispiel zeigt die Art der betroffenen Abfrage.

```
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")
```

- Es wurde ein Problem behoben, bei dem ein Gremlin-Java-Client nach dem Timeout einer lang andauernden Abfrage keine WebSocket Verbindung zum Server über dieselbe Verbindung herstellen konnte.
- Es wurde ein Problem behoben, bei dem die Escape-Sequenzen, die als Teil der Gremlin-Abfrage über HTTP oder zeichenfolgenbasierte Abfragen über die Verbindung enthalten waren, nicht korrekt behandelt wurden. WebSocket

Einführung in die Verwendung von Amazon Neptune APIs

Das Amazon Neptune Neptune-Management APIs bietet SDK-Unterstützung für das Erstellen, Verwalten und Löschen von Neptune-DB-Clustern und -Instances, während die Neptune-Daten SDK-Unterstützung für das Laden von Daten in Ihr Diagramm, das Ausführen von Abfragen, das Abrufen von Informationen über die Daten in Ihrem Diagramm und das Ausführen von Machine-Learning-Vorgängen APIs bieten. Diese sind in allen SDK-Sprachen verfügbar. APIs Durch das automatische Signieren von API-Anfragen integrieren sie Neptune sehr einfach in Anwendungen.

Diese Seite enthält Informationen darüber, wie Sie diese verwenden können APIs.

IAM-Aktionen mit anderen Namen als ihre Gegenstücke zum Neptune Data API SDK

Wenn Sie eine Neptune-API-Methode auf einem Cluster aufrufen, für den die IAM-Authentifizierung aktiviert ist, muss dem Benutzer oder der Rolle, die die Aufrufe ausführt, eine IAM-Richtlinie zugewiesen sein, die Berechtigungen für die gewünschten Aktionen bereitstellt. Sie legen diese Berechtigungen in der Richtlinie mithilfe der entsprechenden IAM-Aktionen fest. Sie können die Aktionen, die ausgeführt werden können, auch mithilfe von IAM-Bedingungsschlüsseln einschränken.

Die meisten IAM-Aktionen haben denselben Namen wie die API-Methoden, denen sie entsprechen, aber einige Methoden in der Daten-API haben unterschiedliche Namen, da einige von mehreren Methoden gemeinsam genutzt werden. In der folgenden Tabelle sind Datenmethoden und die entsprechenden IAM-Aktionen aufgeführt:

Daten-API-Operationsname	IAM-Korrespondenzen
CancelGremlinQuery(cancel_gremlin_query)	Aktion: `neptune-db: CancelQuery`
CancelLoaderJob(Loader_Job stornieren)	Aktion: `neptune-db:CancelLoaderJob`
MLDataProcessingJobAbbrechen (cancel_ml_data_processing_job)	Aktion: `neptune-db:CancelMLDataProcessingJob`

Daten-API-Operationsname	IAM-Korrespondenzen
Abbrechen MLModel TrainingJob (cancel_ml_model_training_job)	Aktion: `neptune-db:CancelMLModelTrainingJob`
CancelOpenCypherQuery(cancel_open_cypher_query)	Aktion: `neptune-db:` **`CancelQuery`**
Erstellen MLEndpoint (create_ml_endpoint)	Aktion: `neptune-db:CreateMLEndpoint`
Löschen (delete_ml_endpoint) MLEndpoint	Aktion: `neptune-db:DeleteMLEndpoint`
DeletePropertygraphStatistics(delete_propertygraph_statistics)	Aktion: `neptune-db:` **`DeleteStatistics`**
DeleteSparqlStatistics(delete_sparql_statistics)	Aktion: `neptune-db:` **`DeleteStatistics`**
ExecuteFastResetfast_reset ausführen ()	Aktion: `neptune-db:` **`ResetDatabase`**
ExecuteGremlinExplainQuery(gremlin_erklären_abfrage ausführen)	Aktionen: • `neptune-db:` **`ReadDataViaQuery`** • `neptune-db:` **`WriteDataViaQuery`** • `neptune-db:` **`DeleteDataViaQuery`** Bedingungsschlüssel: `neptune-db:QueryLanguage:Gremlin`

Daten-API-Operationsname	IAM-Korrespondenzen	
ExecuteGremlinProfileQuery(gremlin_profile_query ausführen)	Aktion: `neptune-db: ReadDataViaQuery` Bedingungsschlüssel: `neptune-db:QueryLanguage:Gremlin`	
ExecuteGremlinQuery(Gremlin_Abfrage ausführen)	Aktionen: • `neptune-db: ReadDataViaQuery` • `neptune-db: WriteDataViaQuery` • `neptune-db: DeleteDataViaQuery` Bedingungsschlüssel: `neptune-db:QueryLanguage:Gremlin`	
ExecuteOpenCypherExplainQuery(Execute_Open_Cypher_Explain_Query)	Aktion: `neptune-db: ReadDataViaQuery` Bedingungsschlüssel: `neptune-db:QueryLanguage:OpenCypher`	

Daten-API-Operationsname	IAM-Korrespondenzen	
ExecuteOpenCypherQuery(Verschlüsselungsabfrage ausführen_öffnen_	Aktionen: • `neptune-db: ReadDataViaQuery` • `neptune-db: WriteDataViaQuery` • `neptune-db: DeleteDataViaQuery` Bedingungsschlüssel: `neptune-db:QueryLanguage:OpenCypher`	
GetEngineStatus(get_engine_status)	Aktion: `neptune-db:GetEngineStatus`	
GetGremlinQueryStatus(get_gremlin_query_status)	Aktion: `neptune-db: :GetQueryStatus` Bedingungsschlüssel: `neptune-db:QueryLanguage:Gremlin`	
GetLoaderJobStatus(get_loader_job_status)	Aktion: `neptune-db:GetLoaderJobStatus`	
MLDataProcessingJobHolen Sie sich (get_ml_data_processing_job)	Aktion: `neptune-db: GetMLDataProcessingJobStatus`	
Holen Sie sich MLEndpoint (get_ml_endpoint)	Aktion: `neptune-db: GetMLEndpointStatus`	

Daten-API-Operationsname	IAM-Korrespondenzen	
Holen Sie sich (get_ml_model_training_job) MLModel TrainingJob	Aktion: `neptune-db:` **`GetMLModelTrainingJobStatus`**	
Holen Sie sich MLModel TransformJob (get_ml_model_transform_job)	Aktion: `neptune-db:` **`GetMLModelTransformJobStatus`**	
GetOpenCypherQueryStatus(get_open_cypher_query_status)	Aktion: `neptune-db:` **`:GetQueryStatus`** Bedingungsschlüssel: `neptune-db:QueryLanguage:OpenCypher`	
GetPropertygraphStatistics(get_propertygraph_statistics)	Aktion: `neptune-db:` **`GetStatisticsStatus`**	
GetPropertygraphStream(get_propertygraph_stream)	Aktion: `neptune-db:` **`GetStreamRecords`** Bedingungsschlüssel: • `neptune-db:QueryLanguage:Gremlin` • `neptune-db:QueryLanguage:OpenCypher`	
GetPropertygraphSummary(get_propertygraph_summary)	Aktion: `neptune-db:` **`GetGraphSummary`**	
Zusammenfassung abrufen (get_rdf_graph_summary) RDFGraph	Aktion: `neptune-db:` **`GetGraphSummary`**	

Daten-API-Operationsname	IAM-Korrespondenzen	
GetSparqlStatistics(get_sparql_statistics)	Aktion: `neptune-db: GetStatisticsStatus`	
GetSparqlStream(get_sparql_stream)	Aktion: `neptune-db: GetStreamRecords` Bedingungsschlüssel: `neptune-db:QueryLanguage:Sparql`	
ListGremlinQueries(list_gremlin_queries)	Aktion: `neptune-db: GetQueryStatus` Bedingungsschlüssel: `neptune-db:QueryLanguage:Gremlin`	
MLEndpointsListe (list_ml_endpoints)	Aktion: `neptune-db:ListMLEndpoints`	
Liste MLModel TrainingJobs (list_ml_model_training_jobs)	Aktion: `neptune-db:ListMLModelTrainingJobs`	
Liste MLModel TransformJobs (list_ml_model_transform_jobs)	Aktion: `neptune-db:ListMLModelTransformJobs`	
ListOpenCypherQueries(list_open_cypher_queries)	Aktion: `neptune-db: GetQueryStatus` Bedingungsschlüssel: `neptune-db:QueryLanguage:OpenCypher`	

Daten-API-Operationsname	IAM-Korrespondenzen	
ManagePropertygraphStatistics(Propertygraph_Statistics verwalten)	Aktion: `neptune-db: ManageStatistics`	
ManageSparqlStatistics(sparql_statistics verwalten)	Aktion: `neptune-db: ManageStatistics`	
StartLoaderJob(start_loader_job)	Aktion: `neptune-db:StartLoaderJob`	
MLModelDataProcessingJobStarten (start_ml_data_processing_job)	Aktion: `neptune-db:StartMLModelDataProcessingJob`	
Starten MLModel TrainingJob (start_ml_model_training_job)	Aktion: `neptune-db:StartMLModelTrainingJob`	
Starten MLModel Transform Job (start_ml_model_transform_job)	Aktion: `neptune-db:StartMLModelTransformJob`	

Amazon Neptune Management-API-Referenz

In diesem Kapitel werden die Neptune-API-Operationen beschrieben, die Sie verwenden können, um Ihre Neptune-DB-Cluster zu verwalten und zu pflegen.

Neptune wird auf Clustern von Datenbankservern ausgeführt, die mittels Replikationstopologie verbunden sind. Daher beinhaltet die Verwaltung von Neptune häufig das Bereitstellen von Änderungen an mehreren Servern und das Sicherstellen, dass alle Neptune-Replicas beim primären Server aufrechterhalten werden.

Da Neptune den zugrunde liegenden Speicher bei wachsender Datenmenge transparent skaliert, erfordert die Verwaltung von Neptune relativ wenig Verwaltung von Festplattenspeicher. Da Neptune automatisch fortlaufend Sicherungen durchführt, erfordert ein Neptune-Cluster ebenfalls keine umfassende Planung und keine Ausfallzeiten für die Durchführung von Sicherungen.

Inhalt

- Neptune-DB-Cluster-API
  - Erstellen DBCluster (Aktion)
  - Löschen DBCluster (Aktion)
  - Ändern DBCluster (Aktion)
  - Start DBCluster (Aktion)
  - Stopp DBCluster (Aktion)
  - AddRoleToDBCluster (Aktion)
  - RemoveRoleFromDBCluster (Aktion)
  - Failover DBCluster (Aktion)
  - PromoteReadReplicaDBCluster (Aktion)
  - Beschreiben DBClusters (Aktion)
  - Strukturen:
  - DBCluster (Struktur)
  - DBClusterMitglied (Struktur)
  - DBClusterRolle (Struktur)
  - CloudwatchLogsExportConfiguration (Struktur)
  - PendingCloudwatchLogsExports (Struktur)

  - ClusterPendingModifiedValues (Struktur)
- Neptune Global Database-API
  - CreateGlobalCluster (Aktion)
  - DeleteGlobalCluster (Aktion)
  - ModifyGlobalCluster (Aktion)
  - DescribeGlobalClusters (Aktion)
  - FailoverGlobalCluster (Aktion)
  - RemoveFromGlobalCluster (Aktion)
  - Strukturen:
  - GlobalCluster (Struktur)
  - GlobalClusterMember (Struktur)
- Neptune-Instances-API
  - Erstellen DBInstance (Aktion)
  - Löschen DBInstance (Aktion)
  - Ändern DBInstance (Aktion)
  - Neustart DBInstance (Aktion)
  - Beschreiben Sie DBInstances (Aktion)
  - DescribeOrderableDBInstanceOptionen (Aktion)
  - DescribeValidDBInstanceÄnderungen (Aktion)
  - Strukturen:
  - DBInstance (Struktur)
  - DBInstanceStatusInfo (Struktur)
  - Bestellbare DBInstance Option (Struktur)
  - PendingModifiedValues (Struktur)
  - ValidStorageOptions (Struktur)
  - Gültig DBInstance ModificationsMessage (Struktur)
- Neptune-Parameter-API
  - DBParameterGruppe kopieren (Aktion)
  - Kopieren DBCluster ParameterGroup (Aktion)
  - DBParameterGruppe erstellen (Aktion)

- Erstellen DBCluster ParameterGroup (Aktion)
- DBParameterGruppe löschen (Aktion)
- Löschen DBCluster ParameterGroup (Aktion)
- DBParameterGruppe ändern (Aktion)
- Ändern DBCluster ParameterGroup (Aktion)
- DBParameterGruppe zurücksetzen (Aktion)
- Zurücksetzen DBCluster ParameterGroup (Aktion)
- Beschreiben Sie DBParameters (Aktion)
- DBParameterGruppen beschreiben (Aktion)
- Beschreiben Sie die DBCluster Parameter (Aktion)
- Beschreiben Sie DBCluster ParameterGroups (Aktion)
- DescribeEngineDefaultParameters (Aktion)
- DescribeEngineDefaultClusterParameters (Aktion)
- Strukturen:
- Parameter (Struktur)
- DBParameterGruppe (Struktur)
- DBClusterParameterGroup (Struktur)
- DBParameterGroupStatus (Struktur)

- Neptune-Subnetz-API
  - DBSubnetGruppe erstellen (Aktion)
  - DBSubnetGruppe löschen (Aktion)
  - DBSubnetGruppe ändern (Aktion)
  - DBSubnetGruppen beschreiben (Aktion)
  - Strukturen:
  - Subnetz (Struktur)
  - DBSubnetGruppe (Struktur)
- Neptune Snapshots-API
  - DBClusterSnapshot erstellen (Aktion)
  - DBClusterSnapshot löschen (Aktion)
  - DBClusterSchnappschuss kopieren (Aktion)

- Ändern DBCluster SnapshotAttribute (Aktion)
- Wiederherstellen DBCluster FromSnapshot (Aktion)
- Wiederherstellen DBCluster ToPointInTime (Aktion)
- Beschreiben Sie DBCluster Schnappschüsse (Aktion)
- Beschreiben Sie DBCluster SnapshotAttributes (Aktion)
- Strukturen:
- DBClusterSnapshot (Struktur)
- DBClusterSnapshotAttribute (Struktur)
- DBClusterSnapshotAttributesResult (Struktur)

- API von Neptune Ereignissen
  - CreateEventSubscription (Aktion)
  - DeleteEventSubscription (Aktion)
  - ModifyEventSubscription (Aktion)
  - DescribeEventSubscriptions (Aktion)
  - AddSourceIdentifierToSubscription (Aktion)
  - RemoveSourceIdentifierFromSubscription (Aktion)
  - DescribeEvents (Aktion)
  - DescribeEventCategories (Aktion)
  - Strukturen:
  - Ereignis (Struktur)
  - EventCategoriesMap (Struktur)
  - EventSubscription (Struktur)
- Anderer Neptune APIs
  - AddTagsToResource (Aktion)
  - ListTagsForResource (Aktion)
  - RemoveTagsFromResource (Aktion)
  - ApplyPendingMaintenanceAction (Aktion)
  - DescribePendingMaintenanceActions (Aktion)
  - DBEngineVersionen beschreiben (Aktion)
  - Strukturen:

- DBEngineVersion (Struktur)
- EngineDefaults (Struktur)
- PendingMaintenanceAction (Struktur)
- ResourcePendingMaintenanceActions (Struktur)
- UpgradeTarget (Struktur)
- Tag (Struktur)

- Häufige Neptune-Datentypen
  - AvailabilityZone (Struktur)
  - DBSecurityGroupMembership (Struktur)
  - DomainMembership (Struktur)
  - DoubleRange (Struktur)
  - Endpunkt (Struktur)
  - Filter (Struktur)
  - Bereich (Struktur)
  - ServerlessV2 ScalingConfiguration (Struktur)
  - ScalingConfigurationInfo ServerlessV2 (Struktur)
  - Zeitzone (Struktur)
  - VpcSecurityGroupMembership (Struktur)
- Individuelle Neptun-Ausnahmen APIs
  - AuthorizationAlreadyExistsFault (Struktur)
  - AuthorizationNotFoundFault (Struktur)
  - AuthorizationQuotaExceededFault (Struktur)
  - CertificateNotFoundFault (Struktur)
  - DBClusterAlreadyExistsFault (Struktur)
  - DBClusterNotFoundFault (Struktur)
  - DBClusterParameterGroupNotFoundFault (Struktur)
  - DBClusterQuotaExceededFault (Struktur)
  - DBClusterRoleAlreadyExistsFault (Struktur)
  - DBClusterRoleNotFoundFault (Struktur)
  - DBClusterRoleQuotaExceededFault (Struktur)

- DBClusterSnapshotAlreadyExistsFault (Struktur)
- DBClusterSnapshotNotFoundFault (Struktur)
- DBInstanceAlreadyExistsFault (Struktur)
- DBInstanceNotFoundFault (Struktur)
- DBLogFileNotFoundFault (Struktur)
- DBParameterGroupAlreadyExistsFault (Struktur)
- DBParameterGroupNotFoundFault (Struktur)
- DBParameterGroupQuotaExceededFault (Struktur)
- DBSecurityGroupAlreadyExistsFault (Struktur)
- DBSecurityGroupNotFoundFault (Struktur)
- DBSecurityGroupNotSupportedFault (Struktur)
- DBSecurityGroupQuotaExceededFault (Struktur)
- DBSnapshotAlreadyExistsFault (Struktur)
- DBSnapshotNotFoundFault (Struktur)
- DBSubnetGroupAlreadyExistsFault (Struktur)
- DBSubnetGroupDoesNotCoverEnoughAZs (Struktur)
- DBSubnetGroupNotAllowedFault (Struktur)
- DBSubnetGroupNotFoundFault (Struktur)
- DBSubnetGroupQuotaExceededFault (Struktur)
- DBSubnetQuotaExceededFault (Struktur)
- DBUpgradeDependencyFailureFault (Struktur)
- DomainNotFoundFault (Struktur)
- EventSubscriptionQuotaExceededFault (Struktur)
- GlobalClusterAlreadyExistsFault (Struktur)
- GlobalClusterNotFoundFault (Struktur)
- GlobalClusterQuotaExceededFault (Struktur)
- InstanceQuotaExceededFault (Struktur)
- Ungenügend DBCluster CapacityFault (Struktur)
- Ungenügend DBInstance CapacityFault (Struktur)
- InsufficientStorageClusterCapacityFault (Struktur)

- Ungültig DBCluster EndpointStateFault (Struktur)
- Ungültig DBCluster SnapshotStateFault (Struktur)
- Ungültig DBCluster StateFault (Struktur)
- Ungültig DBInstance StateFault (Struktur)
- Ungültig DBParameter GroupStateFault (Struktur)
- Ungültig DBSecurity GroupStateFault (Struktur)
- Ungültig DBSnapshot StateFault (Struktur)
- Ungültig DBSubnet GroupFault (Struktur)
- Ungültig DBSubnet GroupStateFault (Struktur)
- Ungültig DBSubnet StateFault (Struktur)
- InvalidEventSubscriptionStateFault (Struktur)
- InvalidGlobalClusterStateFault (Struktur)
- InvalidOptionGroupStateFault (Struktur)
- InvalidRestoreFault (Struktur)
- InvalidSubnet (Struktur)
- Ungültig VPCNetwork StateFault (Struktur)
- KMSKeyNotAccessibleFault (Struktur)
- OptionGroupNotFoundFault (Struktur)
- PointInTimeRestoreNotEnabledFault (Struktur)
- ProvisionedIopsNotAvailableInAZFault (Struktur)
- ResourceNotFoundFault (Struktur)
- SNSInvalidTopicFault (Struktur)
- SNSNoAuthorizationFault (Struktur)
- SNSTopicArnNotFoundFault (Struktur)
- SharedSnapshotQuotaExceededFault (Struktur)
- SnapshotQuotaExceededFault (Struktur)
- SourceNotFoundFault (Struktur)
- StorageQuotaExceededFault (Struktur)
- StorageTypeNotSupportedFault (Struktur)
- SubnetAlreadyInUse (Struktur)

- SubscriptionAlreadyExistFault (Struktur)
- SubscriptionCategoryNotFoundFault (Struktur)
- SubscriptionNotFoundFault (Struktur)

Neptune-DB-Cluster-API

Aktionen:

- Erstellen DBCluster (Aktion)
- Löschen DBCluster (Aktion)
- Ändern DBCluster (Aktion)
- Start DBCluster (Aktion)
- Stopp DBCluster (Aktion)
- AddRoleToDBCluster (Aktion)
- RemoveRoleFromDBCluster (Aktion)
- Failover DBCluster (Aktion)
- PromoteReadReplicaDBCluster (Aktion)
- Beschreiben DBClusters (Aktion)

Strukturen:

- DBCluster (Struktur)
- DBClusterMitglied (Struktur)
- DBClusterRolle (Struktur)
- CloudwatchLogsExportConfiguration (Struktur)
- PendingCloudwatchLogsExports (Struktur)
- ClusterPendingModifiedValues (Struktur)

Erstellen DBCluster (Aktion)

> Der AWS CLI-Name für diese API lautet:`create-db-cluster`.

Erstellt einen neuen Amazon Neptune-DB-Cluster.

Sie können den Parameter `ReplicationSourceIdentifier` verwenden, um den DB-Cluster als Read Replica eines anderen DB-Clusters oder einer Amazon Neptune-DB-Instance zu erstellen.

Beachten Sie, dass beim direkten Erstellen eines neuen Clusters mit `CreateDBCluster` der Löschschutz standardmäßig deaktiviert ist (wenn Sie einen neuen Produktions-Cluster in der Konsole erstellen, ist der Löschschutz standardmäßig aktiviert). Sie können einen DB-Cluster nur löschen, wenn sein `DeletionProtection`-Feld auf `false` gesetzt ist.

Anforderung

- AvailabilityZones (in der CLI: `--availability-zones`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Liste von EC2 Availability Zones, in denen Instances im DB-Cluster erstellt werden können.
- BackupRetentionPeriod(in der CLI:`--backup-retention-period`) — ein IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Die Anzahl von Tagen, über die hinweg automatische Sicherungen aufbewahrt werden. Sie müssen einen Mindestwert von 1 angeben.

  Standard: 1

  Einschränkungen:
  - Muss ein Wert zwischen 1 und 35 sein
- CopyTagsToSnapshot(in der CLI:`--copy-tags-to-snapshot`) — a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Wenn auf *`true`* gesetzt, werden Tags in jeden Snapshot des DB-Clusters kopiert, der erstellt wird.
- DatabaseName (in der CLI: `--database-name`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Name für Ihre Datenbank mit bis zu 64 alphanumerischen Zeichen. Wenn Sie keinen Namen angeben, erstellt Amazon Neptune nicht automatisch eine Datenbank im DB-Cluster, den Sie erstellen.
- DBClusterIdentifier (in der CLI: `--db-cluster-identifier`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die DB-Cluster-ID. Dieser Parameter wird als Zeichenfolge in Kleinbuchstaben gespeichert.

Einschränkungen:

- Muss zwischen 1 und 63 Buchstaben, Ziffern oder Bindestriche enthalten.
- Muss mit einem Buchstaben beginnen.
- Darf nicht mit einem Bindestrich enden oder zwei aufeinanderfolgende Bindestriche enthalten.

Beispiel: `my-cluster1`

- DBClusterParameterGroupName (in der CLI: `--db-cluster-parameter-group-name`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Name der DB-Cluster-Parametergruppe, die mit diesem DB-Cluster verknüpft werden soll. Wenn dieses Argument weggelassen wird, wird der Standardwert verwendet.

  Einschränkungen:

  - Falls angegeben, muss er mit dem Namen eines vorhandenen übereinstimmen. DBCluster ParameterGroup
- DBSubnetGroupName (in der CLI: `--db-subnet-group-name`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine DB-Subnetzgruppe, die diesem DB-Cluster zugeordnet werden soll.

  Einschränkungen: Muss mit dem Namen einer vorhandenen DBSubnet Gruppe übereinstimmen. Der Name darf nicht default sein.

  Beispiel: `mySubnetgroup`
- DeletionProtection(in der CLI:`--deletion-protection`) — a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Ein Wert, der angibt, ob der Löschschutz des DB-Clusters aktiviert ist. Die Datenbank kann nicht gelöscht werden, wenn der Löschschutz aktiviert ist. Standardmäßig ist der Löschschutz deaktiviert.
- EnableCloudwatchLogsExports (in der CLI: `--enable-cloudwatch-logs-exports`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Liste der Protokolltypen, die dieser DB-Cluster in Logs exportieren soll CloudWatch. Gültige Protokolltypen sind: `audit` (zum Veröffentlichen von Audit-Logs) und `slowquery` (zum Veröffentlichen von Logs mit langsamen Abfragen). Weitere Informationen finden Sie unter Neptune-Protokolle in CloudWatch Amazon-Protokollen veröffentlichen.

- EnableIAMDatabaseAuthentication(in der CLI:`--enable-iam-database-authentication`) — a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Wenn auf `true` gesetzt, wird die Amazon Identity and Access Management (IAM)-Authentifizierung für den gesamten DB-Cluster aktiviert (dies kann nicht auf Instance-Ebene festgelegt werden).

  Standard: `false`.
- Engine (in der CLI: `--engine`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Name der Datenbank-Engine, die für diesen DB-Cluster verwendet werden soll.

  Zulässige Werte: `neptune`
- EngineVersion (in der CLI: `--engine-version`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Versionsnummer der Datenbank-Engine, die für den neuen DB-Cluster verwendet werden soll.

  Beispiel: `1.2.1.0`
- GlobalClusterIdentifier(in der CLI:`--global-cluster-identifier`) — a GlobalClusterIdentifier, vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge), nicht weniger als 1 oder mehr als 255? st? s, entspricht diesem regulären Ausdruck:`[A-Za-z][0-9A-Za-z-:._]*`.

  Die ID der globalen Neptune-Datenbank, zu der dieser neue DB-Cluster hinzugefügt werden soll.
- KmsKeyId (in der CLI: `--kms-key-id`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Kennung des Amazon-KMS-Schlüssels für einen verschlüsselten DB-Cluster.

  Die Kennung für den KMS-Schlüssel ist der Amazon-Ressourcenname (ARN) für den KMS-Verschlüsselungsschlüssel. Wenn Sie einen DB-Cluster mit demselben Amazon-Konto erstellen, das über den KMS-Verschlüsselungsschlüssel verfügt, der für die Verschlüsselung des neuen DB-Clusters verwendet wurde, können Sie den KMS-Schlüssel-Alias anstelle des ARN für den KMS-Verschlüsselungsschlüssel verwenden.

  Wenn ein Verschlüsselungsschlüssel in `KmsKeyId` nicht angegeben ist:

  - Wenn `ReplicationSourceIdentifier` eine verschlüsselte Quelle identifiziert, dann verwendet Amazon Neptune den Verschlüsselungsschlüssel zum Verschlüsseln der Quelle. Andernfalls nutzt Amazon Neptune Ihren Standardverschlüsselungsschlüssel.

- Wenn der Parameter `StorageEncrypted` auf "true" festgelegt ist und `ReplicationSourceIdentifier` nicht angegeben ist, dann verwendet Amazon Neptune Ihren Standardverschlüsselungsschlüssel.

  Amazon-KMS erstellt den Standardverschlüsselungsschlüssel für Ihr Amazon-Konto. Ihr Amazon-Konto verfügt für jede Amazon-Region über einen anderen Standardverschlüsselungsschlüssel.

  Wenn Sie eine Read Replica eines verschlüsselten DB-Clusters in einer anderen Amazon-Region erstellen, müssen Sie `KmsKeyId` auf eine KMS-Schlüssel-ID festlegen, die in der Amazon-Zielregion gültig ist. Dieser Schlüssel wird verwendet, um die Read Replica in der entsprechenden Amazon-Region zu verschlüsseln.

- Port(in der CLI:`--port`) — ein IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Die Nummer des Ports, an dem die Instances im DB-Cluster Verbindungen akzeptieren können.

  Standard: `8182`

- PreferredBackupWindow (in der CLI: `--preferred-backup-window`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der tägliche Zeitraum, in dem automatische Sicherungen erstellt werden, wenn diese mit dem Parameter `BackupRetentionPeriod` aktiviert sind.

  Der Standardwert ist ein 30-minütiges Fenster, das zufällig aus einem 8-Stunden-Zeitraum pro Amazon-Region ausgewählt wird. Informationen zu den verfügbaren Zeiträumen finden Sie unter Neptune-Wartungsfenster im Amazon Neptune-Benutzerhandbuch.

  Einschränkungen:

  - Muss im Format `hh24:mi-hh24:mi` angegeben werden.
  - Muss in Universal Coordinated Time (UTC) angegeben werden.
  - Darf nicht mit dem bevorzugten Wartungsfenster in Konflikt treten.
  - Muss mindestens 30 Minuten betragen.

- PreferredMaintenanceWindow (in der CLI: `--preferred-maintenance-window`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der wöchentliche Zeitraum, in dem Systemwartungen durchgeführt werden können, in UTC (Universal Coordinated Time).

Format: `ddd:hh24:mi-ddd:hh24:mi`

Der Standardwert ist ein 30-minütiges Fenster, das zufällig aus einem 8-Stunden-Zeitraum für jede Amazon-Region an einem zufälligen Wochentag ausgewählt wird. Informationen zu den verfügbaren Zeiträumen finden Sie unter Neptune-Wartungsfenster im Amazon Neptune-Benutzerhandbuch.

Gültige Tage: Mo, Di, Mi, Do, Fr, Sa, So.

Einschränkungen: mindestens 30-Minuten-Zeitfenster.

- PreSignedUrl (in der CLI: `--pre-signed-url`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Dieser Parameter wird derzeit nicht unterstützt.

- ReplicationSourceIdentifier (in der CLI: `--replication-source-identifier`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Amazon-Ressourcenname (ARN) der Quell-DB-Instance oder des DB-Clusters, wenn dieser DB-Cluster als Read Replica erstellt wird.

- ServerlessV2ScalingConfiguration (in der CLI: `--serverless-v2-scaling-configuration`) – Ein Serverless V2 ScalingConfiguration-Objekt.

  Enthält die Skalierungskonfiguration eines Neptune Serverless DB-Clusters.

  Weitere Informationen finden Sie unter Verwendung von Amazon Neptune Serverless im Amazon Neptune-Benutzerhandbuch.

- StorageEncrypted(in der CLI:`--storage-encrypted`) — a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Gibt an, ob der DB-Cluster verschlüsselt ist.

- StorageType (in der CLI: `--storage-type`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Speichertyp für den DB-Cluster.

  Zulässige Werte:

- **`standard`**— (Standard) Konfiguriert kostengünstigen Datenbankspeicher für Anwendungen mit mäßiger bis geringer Nutzung. I/O Wenn dies auf `standard` festgelegt ist, wird der Speichertyp in der Antwort nicht zurückgegeben.
- **`iopt1`**— Ermöglicht I/O-optimierten Speicher, der auf die Durchsatzanforderungen zugeschnitten ist. I/O-intensive graph workloads that require predictable pricing with low I/O latency and consistent I/O

  E/A-optimierter Neptune-Speicher ist erst ab Engine-Version 1.3.0.0 verfügbar.

- Tags (in der CLI: `--tags`) – Ein Array von Markierung Objekten.

  Die Tags, die dem neuen DB-Cluster zugewiesen werden sollen.

- VpcSecurityGroupIds (in der CLI: `--vpc-security-group-ids`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Liste von EC2 VPC-Sicherheitsgruppen, die diesem DB-Cluster zugeordnet werden sollen.

Antwort

Enthält die Details eines Amazon Neptune-DB-Clusters.

Dieser Datentyp wird als Antwortelement in der the section called “Beschreiben DBClusters” verwendet.

- AllocatedStorage— ein IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  `AllocatedStorage` gibt immer 1 zurück, da die Neptune-DB-Cluster-Speichergröße nicht fest ist, sondern sich bei Bedarf automatisch anpasst.

- AssociatedRoles – Ein Array mit DBClusterRolle-Objekten.

  Bietet eine Liste der Amazon Identity and Access Management (IAM)-Rollen, die dem DB-Cluster zugeordnet sind. IAM-Rollen, die einem DB-Cluster zugeordnet sind, erteilen dem DB-Cluster die Berechtigung, auf andere Amazon-Services in Ihrem Namen zuzugreifen.

- AutomaticRestartTime— a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Offset von Mitternacht am 01.01.1970).

  Zeitpunkt, zu dem der DB-Cluster automatisch neu gestartet wird.

- AvailabilityZones – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Stellt die Liste der EC2 Availability Zones bereit, in denen Instances im DB-Cluster erstellt werden können.

- BacktrackConsumedChangeRecords— a LongOptional, vom Typ: `long` (eine 64-Bit-Ganzzahl mit Vorzeichen).

  Wird von Neptune nicht unterstützt.
- BacktrackWindow— a LongOptional, vom Typ: `long` (eine 64-Bit-Ganzzahl mit Vorzeichen).

  Wird von Neptune nicht unterstützt.
- BackupRetentionPeriod— an IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Gibt die Anzahl der Tage an, für die automatische DB-Snapshots aufbewahrt werden.
- Capacity— an IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Wird von Neptune nicht unterstützt.
- CloneGroupId – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Identifiziert die Clone-Gruppe, mit der der DB-Cluster verknüpft ist.
- ClusterCreateTime— a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Offset von Mitternacht am 01.01.1970).

  Gibt den Zeitpunkt an, an dem der DB-Cluster erstellt wurde, in UTC (Universal Coordinated Time).
- CopyTagsToSnapshot— a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Wenn auf *`true`* gesetzt, werden Tags in jeden Snapshot des DB-Clusters kopiert, der erstellt wird.
- CrossAccountClone— a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Wenn auf `true` gesetzt, kann der DB-Cluster kontenübergreifend geklont werden.
- DatabaseName – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Enthält den Namen der initialen Datenbank dieses DB-Clusters, die zum Erstellungszeitpunkt bereitgestellt wurde, sofern eine angegeben wurde, als der DB-Cluster erstellt wurde. Derselbe Name wird über die Lebensdauer des DB-Clusters zurückgegeben.
- DBClusterArn – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Der Amazon-Ressourcenname (ARN) für den DB-Cluster.

- DBClusterIdentifier – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Enthält eine vom Benutzer bereitgestellte DB-Cluster-Kennung. Diese Kennung ist der eindeutige Schlüssel zur Identifizierung eines DB-Clusters.

- DBClusterMembers – Ein Array mit DBClusterMitglied-Objekten.

  Enthält eine Liste der Instances, aus denen der DB-Cluster besteht.

- DBClusterParameterGroup – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Namen der DB-Cluster-Parametergruppe für den DB-Cluster an.

- DbClusterResourceId – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die für die Amazon-Region eindeutige, unveränderliche Kennung für den DB-Cluster. Diese Kennung wird in CloudTrail Amazon-Protokolleinträgen immer dann gefunden, wenn auf den Amazon KMS-Schlüssel für den DB-Cluster zugegriffen wird.

- DBSubnetGroup – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt Informationen zu der Subnetzgruppe an, die dem DB-Cluster zugeordnet ist, einschließlich dem Namen, der Beschreibung und der Subnetze in der Subnetzgruppe.

- DeletionProtection— a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Gibt an, ob der Löschschutz des DB-Clusters aktiviert ist. Die Datenbank kann nicht gelöscht werden, wenn der Löschschutz aktiviert ist.

- EarliestBacktrackTime— a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Offset von Mitternacht 1970-01-01).

  Wird von Neptune nicht unterstützt.

- EarliestRestorableTime— a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Versatz von Mitternacht am 01.01.1970).

  Gibt den frühesten Zeitpunkt an, zu dem eine Datenbank mit point-in-time restore wiederhergestellt werden kann.

- EnabledCloudwatchLogsExports – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Eine Liste der Protokolltypen, für deren Export dieser DB-Cluster in CloudWatch Logs konfiguriert ist. Gültige Protokolltypen sind: `audit` (zum Veröffentlichen von Audit-Logs CloudWatch) und slowquery (zum Veröffentlichen von Slow-Query-Logs). CloudWatch Weitere Informationen finden Sie unter Neptune-Protokolle in CloudWatch Amazon-Protokollen veröffentlichen.

- Endpoint – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Verbindungsendpunkt für die primäre Instance des DB-Clusters an.

- Engine – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Namen der Datenbank-Engine an, die für diesen DB-Cluster verwendet werden soll.

- EngineVersion – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt die Version der Datenbank-Engine an.

- GlobalClusterIdentifier— a GlobalClusterIdentifier, vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge), nicht weniger als 1 oder mehr als 255? st? s, entspricht diesem regulären Ausdruck:`[A-Za-z][0-9A-Za-z-:._]*`.

  Enthält eine vom Benutzer bereitgestellte gelobale Datenbank-Cluster-ID. Diese Kennung ist der eindeutige Schlüssel zur Identifizierung einer globalen Datenbank.

- HostedZoneId – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt die ID an, die Amazon Route 53 zuweist, wenn Sie eine gehostete Zone erstellen.

- IAMDatabaseAuthenticationEnabled – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Dies ist „true“, wenn die Zuweisung von Amazon Identity and Access Management (IAM)-Konten zu Datenbank-Konten aktiviert ist. Andernfalls „false“.

- IOOptimizedNextAllowedModificationTime— a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Offset von Mitternacht 1970-01-01).

  Beim nächsten Mal können Sie den DB-Cluster so ändern, dass der Speichertyp `iopt1` verwendet wird.

- KmsKeyId – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Wenn `StorageEncrypted` „true“ ist, ist dies die Amazon-KMS-Schlüsselkennung für das verschlüsselte DB-Cluster.

- LatestRestorableTime— a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Versatz von Mitternacht am 01.01.1970).

  Gibt den letzten Zeitpunkt an, zu dem eine Datenbank mit point-in-time restore wiederhergestellt werden kann.

- MultiAZ – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Gibt an, ob der DB-Cluster über Instances in mehreren Availability Zones verfügt.

- PendingModifiedValues – Ein ClusterPendingModifiedValues-Objekt.

  Dieser Datentyp wird als Antwortelement in der `ModifyDBCluster` Operation verwendet und enthält Änderungen, die während des nächsten Wartungsfensters angewendet werden.

- PercentProgress – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Fortschritt der Operation als Prozentsatz an.

- Port— ein IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Gibt die Portnummer an, die von der Datenbank-Engine überwacht wird.

- PreferredBackupWindow – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den täglichen Zeitraum in koordinierter Weltzeit (UTC) an, in dem automatische Sicherungen erstellt werden, wenn automatische Sicherungen aktiviert sind, gemäß `BackupRetentionPeriod`.

- PreferredMaintenanceWindow – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den wöchentlichen Zeitraum, in dem Systemwartungen durchgeführt werden können, in UTC (Universal Coordinated Time) an.

- ReaderEndpoint – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Leser-Endpunkt für den DB-Cluster. Der Reader-Endpunkt für ein DB-Cluster gleicht die Verbindungslasten zwischen den Read Replicas aus, die in einem DB-Cluster verfügbar sind. Während Clients neue Verbindungsanfragen an den Reader-Endpunkt tätigen, verteilt Neptune die Verbindungsanfragen zwischen den Read Replicas im DB-Cluster. Diese Funktionalität sorgt dafür, dass die Workload-Auslastung für Lesevorgänge auf mehrere Read Replicas im DB-Cluster verteilt wird.

Wenn ein Failover auftritt und das Read Replica, mit dem Sie verbunden sind, als primäre Instance hochgestuft wird, wird Ihre Verbindung unterbrochen. Verbinden Sie sich erneut mit dem Reader-Endpunkt, um mit dem Senden Ihres Workloads für Lesevorgänge an andere Read Replicas im Cluster fortzufahren.

- ReadReplicaIdentifiers – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Enthält eine oder mehrere Kennungen der mit diesem DB-Cluster verbundenen Read Replicas.

- ReplicationSourceIdentifier – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Wird von Neptune nicht unterstützt.

- ReplicationType – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Wird von Neptune nicht unterstützt.

- ServerlessV2ScalingConfiguration – Ein Serverless V2 ScalingConfigurationInfo-Objekt.

  Zeigt die Skalierungskonfiguration für einen Neptune Serverless DB-Cluster.

  Weitere Informationen finden Sie unter Verwendung von Amazon Neptune Serverless im Amazon Neptune-Benutzerhandbuch.

- Status – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den aktuellen Status dieses DB-Clusters an.

- StorageEncrypted – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Gibt an, ob der DB-Cluster verschlüsselt ist.

- StorageType – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Speichertyp, der vom DB-Cluster verwendet wird.

  Zulässige Werte:

  - **`standard`**— (Standard) Bietet kostengünstigen Datenbankspeicher für Anwendungen mit mäßiger bis geringer I/O Nutzung.
  - **`iopt1`**— Ermöglicht I/O-optimierten Speicher, der auf die Durchsatzanforderungen zugeschnitten ist. I/O-intensive graph workloads that require predictable pricing with low I/O latency and consistent I/O

    E/A-optimierter Neptune-Speicher ist erst ab Engine-Version 1.3.0.0 verfügbar.

- VpcSecurityGroups – Ein Array mit VpcSecurityGroupMembership-Objekten.

  Stellt eine Liste der VPC-Sicherheitsgruppen zur Verfügung, zu denen das DB-Cluster gehört.

Fehler

- DBClusterAlreadyExistsFault
- InsufficientStorageClusterCapacityFault
- DBClusterQuotaExceededFault
- StorageQuotaExceededFault
- DBSubnetGroupNotFoundFault
- Ungültig VPCNetwork StateFault
- Ungültig DBCluster StateFault
- Ungültig DBSubnet GroupStateFault
- InvalidSubnet
- Ungültig DBInstance StateFault
- DBClusterParameterGroupNotFoundFault
- KMSKeyNotAccessibleFault
- DBClusterNotFoundFault
- DBInstanceNotFoundFault
- DBSubnetGroupDoesNotCoverEnoughAZs
- GlobalClusterNotFoundFault
- InvalidGlobalClusterStateFault

Löschen DBCluster (Aktion)

Der AWS CLI-Name für diese API lautet:`delete-db-cluster`.

Die DBCluster Aktion Löschen löscht einen zuvor bereitgestellten DB-Cluster. Beim Löschen eines DB-Clusters werden alle automatisch erstellten Sicherungen für diesen DB-Cluster ebenfalls gelöscht und können nicht wiederhergestellt werden. Manuelle DB-Cluster-Snapshots des angegebenen DB-Clusters werden nicht gelöscht.

Der DB-Cluster kann bei aktiviertem Löschschutz nicht gelöscht werden. Um ihn zu löschen, müssen Sie zuerst sein `DeletionProtection`- Feld auf `False` setzen.

Anforderung

- DBClusterIdentifier (in der CLI: `--db-cluster-identifier`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die DB-Cluster-Kennung des zu löschenden DB-Clusters. Bei diesem Parameter wird nicht zwischen Groß- und Kleinschreibung unterschieden.

  Einschränkungen:

  - Muss mit einem vorhandenen DBCluster Identifier übereinstimmen.
- FinalDBSnapshotIdentifier (in der CLI: `--final-db-snapshot-identifier`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Kennung des DB-Cluster-Snapshots des neuen DB-Cluster-Snapshots, das erstellt wird, wenn `SkipFinalSnapshot` auf `false` festgelegt ist.

  > **Note**
  >
  > Wenn dieser Parameter als auch der Parameter `SkipFinalShapshot` auf "true" gesetzt werden, tritt ein Fehler auf.

  Einschränkungen:

  - Muss zwischen 1 und 255 Buchstaben, Zahlen oder Bindestriche enthalten.
  - Muss mit einem Buchstaben beginnen
  - Darf nicht mit einem Bindestrich enden oder zwei aufeinanderfolgende Bindestriche enthalten
- SkipFinalSnapshot (in der CLI: `--skip-final-snapshot`) – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Legt fest, ob vor dem Löschen des DB-Clusters ein finaler DB-Cluster-Snapshot erstellt wird. Wenn `true` angegeben ist, wird kein DB-Cluster-Snapshot erstellt. Wenn `false` festgelegt ist, wird vor dem Löschen des DB-Clusters ein DB-Cluster-Snapshot erstellt.

> **Note**
>
> Sie müssen einen `FinalDBSnapshotIdentifier`-Parameter angeben, wenn `SkipFinalSnapshot` `false` ist.

Standard: `false`

Antwort

Enthält die Details eines Amazon Neptune-DB-Clusters.

Dieser Datentyp wird als Antwortelement in der the section called “Beschreiben DBClusters” verwendet.

- AllocatedStorage— ein IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  `AllocatedStorage` gibt immer 1 zurück, da die Neptune-DB-Cluster-Speichergröße nicht fest ist, sondern sich bei Bedarf automatisch anpasst.
- AssociatedRoles – Ein Array mit DBClusterRolle-Objekten.

  Bietet eine Liste der Amazon Identity and Access Management (IAM)-Rollen, die dem DB-Cluster zugeordnet sind. IAM-Rollen, die einem DB-Cluster zugeordnet sind, erteilen dem DB-Cluster die Berechtigung, auf andere Amazon-Services in Ihrem Namen zuzugreifen.
- AutomaticRestartTime— a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Offset von Mitternacht am 01.01.1970).

  Zeitpunkt, zu dem der DB-Cluster automatisch neu gestartet wird.
- AvailabilityZones – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Stellt die Liste der EC2 Availability Zones bereit, in denen Instances im DB-Cluster erstellt werden können.
- BacktrackConsumedChangeRecords— a LongOptional, vom Typ: `long` (eine 64-Bit-Ganzzahl mit Vorzeichen).

  Wird von Neptune nicht unterstützt.
- BacktrackWindow— a LongOptional, vom Typ: `long` (eine 64-Bit-Ganzzahl mit Vorzeichen).

Wird von Neptune nicht unterstützt.

- BackupRetentionPeriod— an IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Gibt die Anzahl der Tage an, für die automatische DB-Snapshots aufbewahrt werden.

- Capacity— an IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Wird von Neptune nicht unterstützt.

- CloneGroupId – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Identifiziert die Clone-Gruppe, mit der der DB-Cluster verknüpft ist.

- ClusterCreateTime— a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Offset von Mitternacht am 01.01.1970).

  Gibt den Zeitpunkt an, an dem der DB-Cluster erstellt wurde, in UTC (Universal Coordinated Time).

- CopyTagsToSnapshot— a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Wenn auf *`true`* gesetzt, werden Tags in jeden Snapshot des DB-Clusters kopiert, der erstellt wird.

- CrossAccountClone— a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Wenn auf `true` gesetzt, kann der DB-Cluster kontenübergreifend geklont werden.

- DatabaseName – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Enthält den Namen der initialen Datenbank dieses DB-Clusters, die zum Erstellungszeitpunkt bereitgestellt wurde, sofern eine angegeben wurde, als der DB-Cluster erstellt wurde. Derselbe Name wird über die Lebensdauer des DB-Clusters zurückgegeben.

- DBClusterArn – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Amazon-Ressourcenname (ARN) für den DB-Cluster.

- DBClusterIdentifier – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Enthält eine vom Benutzer bereitgestellte DB-Cluster-Kennung. Diese Kennung ist der eindeutige Schlüssel zur Identifizierung eines DB-Clusters.

- DBClusterMembers – Ein Array mit DBClusterMitglied-Objekten.

Enthält eine Liste der Instances, aus denen der DB-Cluster besteht.

- DBClusterParameterGroup – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Namen der DB-Cluster-Parametergruppe für den DB-Cluster an.

- DbClusterResourceId – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die für die Amazon-Region eindeutige, unveränderliche Kennung für den DB-Cluster. Diese Kennung wird in CloudTrail Amazon-Protokolleinträgen immer dann gefunden, wenn auf den Amazon KMS-Schlüssel für den DB-Cluster zugegriffen wird.

- DBSubnetGroup – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt Informationen zu der Subnetzgruppe an, die dem DB-Cluster zugeordnet ist, einschließlich dem Namen, der Beschreibung und der Subnetze in der Subnetzgruppe.

- DeletionProtection— a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Gibt an, ob der Löschschutz des DB-Clusters aktiviert ist. Die Datenbank kann nicht gelöscht werden, wenn der Löschschutz aktiviert ist.

- EarliestBacktrackTime— a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Offset von Mitternacht 1970-01-01).

  Wird von Neptune nicht unterstützt.

- EarliestRestorableTime— a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Versatz von Mitternacht am 01.01.1970).

  Gibt den frühesten Zeitpunkt an, zu dem eine Datenbank mit point-in-time restore wiederhergestellt werden kann.

- EnabledCloudwatchLogsExports – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Liste der Protokolltypen, für deren Export dieser DB-Cluster in CloudWatch Logs konfiguriert ist. Gültige Protokolltypen sind: `audit` (zum Veröffentlichen von Audit-Logs CloudWatch) und slowquery (zum Veröffentlichen von Slow-Query-Logs). CloudWatch Weitere Informationen finden Sie unter Neptune-Protokolle in CloudWatch Amazon-Protokollen veröffentlichen.

- Endpoint – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Verbindungsendpunkt für die primäre Instance des DB-Clusters an.

- Engine – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Namen der Datenbank-Engine an, die für diesen DB-Cluster verwendet werden soll.
- EngineVersion – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt die Version der Datenbank-Engine an.
- GlobalClusterIdentifier— a GlobalClusterIdentifier, vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge), nicht weniger als 1 oder mehr als 255? st? s, entspricht diesem regulären Ausdruck:`[A-Za-z][0-9A-Za-z-:._]*`.

  Enthält eine vom Benutzer bereitgestellte gelobale Datenbank-Cluster-ID. Diese Kennung ist der eindeutige Schlüssel zur Identifizierung einer globalen Datenbank.
- HostedZoneId – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt die ID an, die Amazon Route 53 zuweist, wenn Sie eine gehostete Zone erstellen.
- IAMDatabaseAuthenticationEnabled – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Dies ist „true“, wenn die Zuweisung von Amazon Identity and Access Management (IAM)-Konten zu Datenbank-Konten aktiviert ist. Andernfalls „false“.
- IOOptimizedNextAllowedModificationTime— a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Offset von Mitternacht 1970-01-01).

  Beim nächsten Mal können Sie den DB-Cluster so ändern, dass der Speichertyp `iopt1` verwendet wird.
- KmsKeyId – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Wenn `StorageEncrypted` „true“ ist, ist dies die Amazon-KMS-Schlüsselkennung für das verschlüsselte DB-Cluster.
- LatestRestorableTime— a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Versatz von Mitternacht am 01.01.1970).

  Gibt den letzten Zeitpunkt an, zu dem eine Datenbank mit point-in-time restore wiederhergestellt werden kann.
- MultiAZ – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Gibt an, ob der DB-Cluster über Instances in mehreren Availability Zones verfügt.
- PendingModifiedValues – Ein ClusterPendingModifiedValues-Objekt.

Dieser Datentyp wird als Antwortelement in der `ModifyDBCluster` Operation verwendet und enthält Änderungen, die während des nächsten Wartungsfensters angewendet werden.

- PercentProgress – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Fortschritt der Operation als Prozentsatz an.

- Port— ein IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Gibt die Portnummer an, die von der Datenbank-Engine überwacht wird.

- PreferredBackupWindow – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den täglichen Zeitraum in koordinierter Weltzeit (UTC) an, in dem automatische Sicherungen erstellt werden, wenn automatische Sicherungen aktiviert sind, gemäß `BackupRetentionPeriod`.

- PreferredMaintenanceWindow – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den wöchentlichen Zeitraum, in dem Systemwartungen durchgeführt werden können, in UTC (Universal Coordinated Time) an.

- ReaderEndpoint – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Leser-Endpunkt für den DB-Cluster. Der Reader-Endpunkt für ein DB-Cluster gleicht die Verbindungslasten zwischen den Read Replicas aus, die in einem DB-Cluster verfügbar sind. Während Clients neue Verbindungsanfragen an den Reader-Endpunkt tätigen, verteilt Neptune die Verbindungsanfragen zwischen den Read Replicas im DB-Cluster. Diese Funktionalität sorgt dafür, dass die Workload-Auslastung für Lesevorgänge auf mehrere Read Replicas im DB-Cluster verteilt wird.

  Wenn ein Failover auftritt und das Read Replica, mit dem Sie verbunden sind, als primäre Instance hochgestuft wird, wird Ihre Verbindung unterbrochen. Verbinden Sie sich erneut mit dem Reader-Endpunkt, um mit dem Senden Ihres Workloads für Lesevorgänge an andere Read Replicas im Cluster fortzufahren.

- ReadReplicaIdentifiers – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Enthält eine oder mehrere Kennungen der mit diesem DB-Cluster verbundenen Read Replicas.

- ReplicationSourceIdentifier – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Wird von Neptune nicht unterstützt.

- ReplicationType – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Wird von Neptune nicht unterstützt.

- ServerlessV2ScalingConfiguration – Ein Serverless V2 ScalingConfigurationInfo-Objekt.

  Zeigt die Skalierungskonfiguration für einen Neptune Serverless DB-Cluster.

  Weitere Informationen finden Sie unter Verwendung von Amazon Neptune Serverless im Amazon Neptune-Benutzerhandbuch.

- Status – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den aktuellen Status dieses DB-Clusters an.

- StorageEncrypted – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Gibt an, ob der DB-Cluster verschlüsselt ist.

- StorageType – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Speichertyp, der vom DB-Cluster verwendet wird.

  Zulässige Werte:

  - **`standard`**— (Standard) Bietet kostengünstigen Datenbankspeicher für Anwendungen mit mäßiger bis geringer I/O Nutzung.
  - **`iopt1`**— Ermöglicht I/O-optimierten Speicher, der auf die Durchsatzanforderungen zugeschnitten ist. I/O-intensive graph workloads that require predictable pricing with low I/O latency and consistent I/O

    E/A-optimierter Neptune-Speicher ist erst ab Engine-Version 1.3.0.0 verfügbar.

- VpcSecurityGroups – Ein Array mit VpcSecurityGroupMembership-Objekten.

  Stellt eine Liste der VPC-Sicherheitsgruppen zur Verfügung, zu denen das DB-Cluster gehört.

Fehler

- DBClusterNotFoundFault
- Ungültig DBCluster StateFault
- DBClusterSnapshotAlreadyExistsFault
- SnapshotQuotaExceededFault

- Ungültig DBCluster SnapshotStateFault

Ändern DBCluster (Aktion)

Der AWS CLI-Name für diese API lautet:`modify-db-cluster`.

Ändern Sie eine Einstellung für einen DB-Cluster. Sie können einen oder mehrere Datenbank-Konfigurationsparameter ändern, indem Sie diese Parameter und die neuen Werte in der Anforderung angeben.

Anforderung

- AllowMajorVersionUpgrade (in der CLI: `--allow-major-version-upgrade`) – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Ein Wert, der angibt, ob Upgrades zwischen verschiedenen Hauptversionen erlaubt sind.

  Einschränkungen: Sie müssen das allow-major-version-upgrade Flag setzen, wenn Sie einen `EngineVersion` Parameter angeben, der eine andere Hauptversion als die aktuelle Version des DB-Clusters verwendet.
- ApplyImmediately (in der CLI: `--apply-immediately`) – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Ein Wert, der angibt, ob die Änderungen in dieser Anforderung und sämtliche ausstehenden Änderungen so bald wie möglich asynchron angewendet werden, und zwar unabhängig von der Einstellung `PreferredMaintenanceWindow` für den DB-Cluster. Wenn dieser Parameter auf `false` festgelegt ist, werden Änderungen am DB-Cluster im nächsten Wartungsfenster angewendet.

  Der Parameter `ApplyImmediately` wirkt sich nur auf die `NewDBClusterIdentifier`-Werte aus. Wenn Sie den Wert des Parameters `ApplyImmediately` auf „false“ festlegen, werden Änderungen an den `NewDBClusterIdentifier`-Werten im nächsten Wartungsfenster übernommen. Alle anderen Änderungen werden sofort übernommen, unabhängig von dem Wert des Parameters `ApplyImmediately`.

  Standard: `false`
- BackupRetentionPeriod(in der CLI:`--backup-retention-period`) — ein IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

Die Anzahl von Tagen, über die hinweg automatische Sicherungen aufbewahrt werden. Sie müssen einen Mindestwert von 1 angeben.

Standard: 1

Einschränkungen:

- Muss ein Wert zwischen 1 und 35 sein

- CloudwatchLogsExportConfiguration (in der CLI: `--cloudwatch-logs-export-configuration`) – Ein CloudwatchLogsExportConfiguration-Objekt.

  Die Konfigurationseinstellung für die Protokolltypen, die für den Export in CloudWatch Logs für einen bestimmten DB-Cluster aktiviert werden sollen. Siehe Verwenden der CLI zum Veröffentlichen von Neptune-Auditprotokollen in Logs. CloudWatch

- CopyTagsToSnapshot(in der CLI:`--copy-tags-to-snapshot`) — a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Wenn auf *`true`* gesetzt, werden Tags in jeden Snapshot des DB-Clusters kopiert, der erstellt wird.

- DBClusterIdentifier (in der CLI: `--db-cluster-identifier`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die DB-Cluster-Kennung des zu ändernden DB-Clusters. Bei diesem Parameter wird nicht zwischen Groß- und Kleinschreibung unterschieden.

  Einschränkungen:

  - Muss mit dem Bezeichner eines vorhandenen Bezeichners übereinstimmen. DBCluster

- DBClusterParameterGroupName (in der CLI: `--db-cluster-parameter-group-name`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Name der DB-Cluster-Parametergruppe, die für den DB-Cluster verwendet werden soll.

- DBInstanceParameterGroupName (in der CLI: `--db-instance-parameter-group-name`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Name der DB-Parametergruppe, die auf alle Instances des DB-Clusters angewendet wird.

> **Note**
>
> Wenn Sie eine Parametergruppe mithilfe von `DBInstanceParameterGroupName` anwenden, werden Parameteränderungen nicht im nächsten Wartungsfenster angewendet, sondern sofort angewendet.

Standard: Die vorhandene Namenseinstellung

Einschränkungen:

- Die DB-Parametergruppe muss zur gleichen DB-Parametergruppenfamilie gehören wie die Ziel-DB-Cluster-Version.
- Der Parameter `DBInstanceParameterGroupName` ist nur in Kombination mit dem Parameter `AllowMajorVersionUpgrade` gültig.

- DeletionProtection(in der CLI:`--deletion-protection`) — a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Ein Wert, der angibt, ob der Löschschutz des DB-Clusters aktiviert ist. Die Datenbank kann nicht gelöscht werden, wenn der Löschschutz aktiviert ist. Der Löschschutz ist standardmäßig deaktiviert.
- EnableIAMDatabaseAuthentication(in der CLI:`--enable-iam-database-authentication`) — a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  „True“, um die Zuweisung von Amazon Identity and Access Management (IAM)-Konten zu Datenbankkonten zu aktivieren; andernfalls „false“.

  Standard: `false`
- EngineVersion (in der CLI: `--engine-version`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Versionsnummer der Datenbank-Engine, auf die ein Upgrade durchgeführt werden soll. Das Ändern dieses Parameters führt zu einem Nutzungsausfall. Die Änderung wird im nächsten Wartungsfenster übernommen, es sei denn, der Parameter `ApplyImmediately` ist auf „true“ festgelegt.

  Eine Liste der gültigen Engine-Versionen finden Sie unter Engine-Releases für Amazon Neptune, oder rufen Sie the section called “DBEngineVersionen beschreiben” auf.

- NewDBClusterIdentifier (in der CLI: `--new-db-cluster-identifier`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die neue DB-Cluster-Kennung für den DB-Cluster, wenn ein DB-Cluster umbenannt wird. Dieser Wert wird als Zeichenfolge in Kleinbuchstaben gespeichert.

  Einschränkungen:

  - Muss zwischen 1 und 63 Buchstaben, Ziffern oder Bindestriche enthalten.
  - Das erste Zeichen muss ein Buchstabe sein.
  - Darf nicht mit einem Bindestrich enden oder zwei aufeinanderfolgende Bindestriche enthalten

  Beispiel: `my-cluster2`
- Port(in der CLI:`--port`) — ein IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Die Nummer des Ports, an dem das DB-Cluster Verbindungen akzeptiert.

  Einschränkungen: Der Wert muss `1150-65535` lauten.

  Standard: Der gleiche Port wie der ursprüngliche DB-Cluster.
- PreferredBackupWindow (in der CLI: `--preferred-backup-window`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der tägliche Zeitraum, in dem automatische Sicherungen erstellt werden, wenn diese mit dem Parameter `BackupRetentionPeriod` aktiviert sind.

  Der Standardwert ist ein 30-minütiges Fenster, das zufällig aus einem 8-Stunden-Zeitraum pro Amazon-Region ausgewählt wird.

  Einschränkungen:

  - Muss im Format `hh24:mi-hh24:mi` angegeben werden.
  - Muss in Universal Coordinated Time (UTC) angegeben werden.
  - Darf nicht mit dem bevorzugten Wartungsfenster in Konflikt treten.
  - Muss mindestens 30 Minuten betragen.
- PreferredMaintenanceWindow (in der CLI: `--preferred-maintenance-window`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Der wöchentliche Zeitraum, in dem Systemwartungen durchgeführt werden können, in UTC (Universal Coordinated Time).

Format: `ddd:hh24:mi-ddd:hh24:mi`

Der Standardwert ist ein 30-minütiges Fenster, das zufällig aus einem 8-Stunden-Zeitraum für jede Amazon-Region an einem zufälligen Wochentag ausgewählt wird.

Gültige Tage: Mo, Di, Mi, Do, Fr, Sa, So.

Einschränkungen: mindestens 30-Minuten-Zeitfenster.

- ServerlessV2ScalingConfiguration (in der CLI: `--serverless-v2-scaling-configuration`) – Ein Serverless V2 ScalingConfiguration-Objekt.

  Enthält die Skalierungskonfiguration eines Neptune Serverless DB-Clusters.

  Weitere Informationen finden Sie unter Verwendung von Amazon Neptune Serverless im Amazon Neptune-Benutzerhandbuch.

- StorageType (in der CLI: `--storage-type`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Speichertyp, der dem DB-Cluster zugeordnet werden soll.

  Zulässige Werte:

  - **`standard`**— (Standard) Konfiguriert kostengünstigen Datenbankspeicher für Anwendungen mit mäßiger bis geringer I/O Nutzung.
  - **`iopt1`**— Ermöglicht I/O-optimierten Speicher, der auf die Durchsatzanforderungen zugeschnitten ist. I/O-intensive graph workloads that require predictable pricing with low I/O latency and consistent I/O

    E/A-optimierter Neptune-Speicher ist erst ab Engine-Version 1.3.0.0 verfügbar.

- VpcSecurityGroupIds (in der CLI: `--vpc-security-group-ids`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Liste der VPC-Sicherheitsgruppen, zu denen der DB-Cluster gehören wird.

Antwort

Enthält die Details eines Amazon Neptune-DB-Clusters.

Dieser Datentyp wird als Antwortelement in der the section called “Beschreiben DBClusters” verwendet.

- AllocatedStorage— an IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  `AllocatedStorage` gibt immer 1 zurück, da die Neptune-DB-Cluster-Speichergröße nicht fest ist, sondern sich bei Bedarf automatisch anpasst.
- AssociatedRoles – Ein Array mit DBClusterRolle-Objekten.

  Bietet eine Liste der Amazon Identity and Access Management (IAM)-Rollen, die dem DB-Cluster zugeordnet sind. IAM-Rollen, die einem DB-Cluster zugeordnet sind, erteilen dem DB-Cluster die Berechtigung, auf andere Amazon-Services in Ihrem Namen zuzugreifen.
- AutomaticRestartTime— a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Offset von Mitternacht am 01.01.1970).

  Zeitpunkt, zu dem der DB-Cluster automatisch neu gestartet wird.
- AvailabilityZones – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Stellt die Liste der EC2 Availability Zones bereit, in denen Instances im DB-Cluster erstellt werden können.
- BacktrackConsumedChangeRecords— a LongOptional, vom Typ: `long` (eine 64-Bit-Ganzzahl mit Vorzeichen).

  Wird von Neptune nicht unterstützt.
- BacktrackWindow— a LongOptional, vom Typ: `long` (eine 64-Bit-Ganzzahl mit Vorzeichen).

  Wird von Neptune nicht unterstützt.
- BackupRetentionPeriod— an IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Gibt die Anzahl der Tage an, für die automatische DB-Snapshots aufbewahrt werden.
- Capacity— an IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Wird von Neptune nicht unterstützt.
- CloneGroupId – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Identifiziert die Clone-Gruppe, mit der der DB-Cluster verknüpft ist.

- ClusterCreateTime— a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Offset von Mitternacht am 01.01.1970).

  Gibt den Zeitpunkt an, an dem der DB-Cluster erstellt wurde, in UTC (Universal Coordinated Time).

- CopyTagsToSnapshot— a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Wenn auf *`true`* gesetzt, werden Tags in jeden Snapshot des DB-Clusters kopiert, der erstellt wird.

- CrossAccountClone— a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Wenn auf `true` gesetzt, kann der DB-Cluster kontenübergreifend geklont werden.

- DatabaseName – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Enthält den Namen der initialen Datenbank dieses DB-Clusters, die zum Erstellungszeitpunkt bereitgestellt wurde, sofern eine angegeben wurde, als der DB-Cluster erstellt wurde. Derselbe Name wird über die Lebensdauer des DB-Clusters zurückgegeben.

- DBClusterArn – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Amazon-Ressourcenname (ARN) für den DB-Cluster.

- DBClusterIdentifier – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Enthält eine vom Benutzer bereitgestellte DB-Cluster-Kennung. Diese Kennung ist der eindeutige Schlüssel zur Identifizierung eines DB-Clusters.

- DBClusterMembers – Ein Array mit DBClusterMitglied-Objekten.

  Enthält eine Liste der Instances, aus denen der DB-Cluster besteht.

- DBClusterParameterGroup – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Namen der DB-Cluster-Parametergruppe für den DB-Cluster an.

- DbClusterResourceId – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die für die Amazon-Region eindeutige, unveränderliche Kennung für den DB-Cluster. Diese Kennung wird in CloudTrail Amazon-Protokolleinträgen immer dann gefunden, wenn auf den Amazon KMS-Schlüssel für den DB-Cluster zugegriffen wird.

- DBSubnetGroup – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Gibt Informationen zu der Subnetzgruppe an, die dem DB-Cluster zugeordnet ist, einschließlich dem Namen, der Beschreibung und der Subnetze in der Subnetzgruppe.

- DeletionProtection— a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Gibt an, ob der Löschschutz des DB-Clusters aktiviert ist. Die Datenbank kann nicht gelöscht werden, wenn der Löschschutz aktiviert ist.

- EarliestBacktrackTime— a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Offset von Mitternacht 1970-01-01).

  Wird von Neptune nicht unterstützt.

- EarliestRestorableTime— a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Versatz von Mitternacht am 01.01.1970).

  Gibt den frühesten Zeitpunkt an, zu dem eine Datenbank mit point-in-time restore wiederhergestellt werden kann.

- EnabledCloudwatchLogsExports – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Liste der Protokolltypen, für deren Export dieser DB-Cluster in CloudWatch Logs konfiguriert ist. Gültige Protokolltypen sind: `audit` (zum Veröffentlichen von Audit-Logs CloudWatch) und slowquery (zum Veröffentlichen von Slow-Query-Logs). CloudWatch Weitere Informationen finden Sie unter Neptune-Protokolle in CloudWatch Amazon-Protokollen veröffentlichen.

- Endpoint – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Verbindungsendpunkt für die primäre Instance des DB-Clusters an.

- Engine – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Namen der Datenbank-Engine an, die für diesen DB-Cluster verwendet werden soll.

- EngineVersion – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt die Version der Datenbank-Engine an.

- GlobalClusterIdentifier— a GlobalClusterIdentifier, vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge), nicht weniger als 1 oder mehr als 255? st? s, entspricht diesem regulären Ausdruck:`[A-Za-z][0-9A-Za-z-:._]*`.

Enthält eine vom Benutzer bereitgestellte gelobale Datenbank-Cluster-ID. Diese Kennung ist der eindeutige Schlüssel zur Identifizierung einer globalen Datenbank.

- HostedZoneId – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt die ID an, die Amazon Route 53 zuweist, wenn Sie eine gehostete Zone erstellen.

- IAMDatabaseAuthenticationEnabled – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Dies ist „true“, wenn die Zuweisung von Amazon Identity and Access Management (IAM)-Konten zu Datenbank-Konten aktiviert ist. Andernfalls „false“.

- IOOptimizedNextAllowedModificationTime— a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Offset von Mitternacht 1970-01-01).

  Beim nächsten Mal können Sie den DB-Cluster so ändern, dass der Speichertyp `iopt1` verwendet wird.

- KmsKeyId – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Wenn `StorageEncrypted` „true“ ist, ist dies die Amazon-KMS-Schlüsselkennung für das verschlüsselte DB-Cluster.

- LatestRestorableTime— a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Versatz von Mitternacht am 01.01.1970).

  Gibt den letzten Zeitpunkt an, zu dem eine Datenbank mit point-in-time restore wiederhergestellt werden kann.

- MultiAZ – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Gibt an, ob der DB-Cluster über Instances in mehreren Availability Zones verfügt.

- PendingModifiedValues – Ein ClusterPendingModifiedValues-Objekt.

  Dieser Datentyp wird als Antwortelement in der `ModifyDBCluster` Operation verwendet und enthält Änderungen, die während des nächsten Wartungsfensters angewendet werden.

- PercentProgress – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Fortschritt der Operation als Prozentsatz an.

- Port— ein IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Gibt die Portnummer an, die von der Datenbank-Engine überwacht wird.

- PreferredBackupWindow – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den täglichen Zeitraum in koordinierter Weltzeit (UTC) an, in dem automatische Sicherungen erstellt werden, wenn automatische Sicherungen aktiviert sind, gemäß `BackupRetentionPeriod`.

- PreferredMaintenanceWindow – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den wöchentlichen Zeitraum, in dem Systemwartungen durchgeführt werden können, in UTC (Universal Coordinated Time) an.

- ReaderEndpoint – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Leser-Endpunkt für den DB-Cluster. Der Reader-Endpunkt für ein DB-Cluster gleicht die Verbindungslasten zwischen den Read Replicas aus, die in einem DB-Cluster verfügbar sind. Während Clients neue Verbindungsanfragen an den Reader-Endpunkt tätigen, verteilt Neptune die Verbindungsanfragen zwischen den Read Replicas im DB-Cluster. Diese Funktionalität sorgt dafür, dass die Workload-Auslastung für Lesevorgänge auf mehrere Read Replicas im DB-Cluster verteilt wird.

  Wenn ein Failover auftritt und das Read Replica, mit dem Sie verbunden sind, als primäre Instance hochgestuft wird, wird Ihre Verbindung unterbrochen. Verbinden Sie sich erneut mit dem Reader-Endpunkt, um mit dem Senden Ihres Workloads für Lesevorgänge an andere Read Replicas im Cluster fortzufahren.

- ReadReplicaIdentifiers – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Enthält eine oder mehrere Kennungen der mit diesem DB-Cluster verbundenen Read Replicas.

- ReplicationSourceIdentifier – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Wird von Neptune nicht unterstützt.

- ReplicationType – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Wird von Neptune nicht unterstützt.

- ServerlessV2ScalingConfiguration – Ein Serverless V2 ScalingConfigurationInfo-Objekt.

  Zeigt die Skalierungskonfiguration für einen Neptune Serverless DB-Cluster.

  Weitere Informationen finden Sie unter Verwendung von Amazon Neptune Serverless im Amazon Neptune-Benutzerhandbuch.

- Status – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den aktuellen Status dieses DB-Clusters an.

- StorageEncrypted – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Gibt an, ob der DB-Cluster verschlüsselt ist.

- StorageType – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Speichertyp, der vom DB-Cluster verwendet wird.

  Zulässige Werte:

  - **`standard`**— (Standard) Bietet kostengünstigen Datenbankspeicher für Anwendungen mit mäßiger bis geringer I/O Nutzung.
  - **`iopt1`**— Ermöglicht I/O-optimierten Speicher, der auf die Durchsatzanforderungen zugeschnitten ist. I/O-intensive graph workloads that require predictable pricing with low I/O latency and consistent I/O

    E/A-optimierter Neptune-Speicher ist erst ab Engine-Version 1.3.0.0 verfügbar.

- VpcSecurityGroups – Ein Array mit VpcSecurityGroupMembership-Objekten.

  Stellt eine Liste der VPC-Sicherheitsgruppen zur Verfügung, zu denen das DB-Cluster gehört.

Fehler

- DBClusterNotFoundFault
- Ungültig DBCluster StateFault
- StorageQuotaExceededFault
- DBSubnetGroupNotFoundFault
- Ungültig VPCNetwork StateFault
- Ungültig DBSubnet GroupStateFault
- InvalidSubnet
- DBClusterParameterGroupNotFoundFault
- Ungültig DBSecurity GroupStateFault
- Ungültig DBInstance StateFault
- DBClusterAlreadyExistsFault

- StorageTypeNotSupportedFault

Start DBCluster (Aktion)

Der AWS CLI-Name für diese API lautet:`start-db-cluster`.

Startet einen Amazon Neptune Neptune-DB-Cluster, der mit der Amazon-Konsole, dem Amazon stop-db-cluster CLI-Befehl oder der DBCluster Stop-API gestoppt wurde.

Anforderung

- DBClusterIdentifier (in der CLI: `--db-cluster-identifier`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die DB-Clusterkennung des zu startenden Neptune DB-Clusters. Dieser Parameter wird als Zeichenfolge in Kleinbuchstaben gespeichert.

Antwort

Enthält die Details eines Amazon Neptune-DB-Clusters.

Dieser Datentyp wird als Antwortelement in der the section called “Beschreiben DBClusters” verwendet.

- AllocatedStorage— ein IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  `AllocatedStorage` gibt immer 1 zurück, da die Neptune-DB-Cluster-Speichergröße nicht fest ist, sondern sich bei Bedarf automatisch anpasst.
- AssociatedRoles – Ein Array mit DBClusterRolle-Objekten.

  Bietet eine Liste der Amazon Identity and Access Management (IAM)-Rollen, die dem DB-Cluster zugeordnet sind. IAM-Rollen, die einem DB-Cluster zugeordnet sind, erteilen dem DB-Cluster die Berechtigung, auf andere Amazon-Services in Ihrem Namen zuzugreifen.
- AutomaticRestartTime— a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Offset von Mitternacht am 01.01.1970).

  Zeitpunkt, zu dem der DB-Cluster automatisch neu gestartet wird.
- AvailabilityZones – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Stellt die Liste der EC2 Availability Zones bereit, in denen Instances im DB-Cluster erstellt werden können.

- BacktrackConsumedChangeRecords— a LongOptional, vom Typ: `long` (eine 64-Bit-Ganzzahl mit Vorzeichen).

  Wird von Neptune nicht unterstützt.

- BacktrackWindow— a LongOptional, vom Typ: `long` (eine 64-Bit-Ganzzahl mit Vorzeichen).

  Wird von Neptune nicht unterstützt.

- BackupRetentionPeriod— an IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Gibt die Anzahl der Tage an, für die automatische DB-Snapshots aufbewahrt werden.

- Capacity— an IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Wird von Neptune nicht unterstützt.

- CloneGroupId – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Identifiziert die Clone-Gruppe, mit der der DB-Cluster verknüpft ist.

- ClusterCreateTime— a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Offset von Mitternacht am 01.01.1970).

  Gibt den Zeitpunkt an, an dem der DB-Cluster erstellt wurde, in UTC (Universal Coordinated Time).

- CopyTagsToSnapshot— a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Wenn auf *`true`* gesetzt, werden Tags in jeden Snapshot des DB-Clusters kopiert, der erstellt wird.

- CrossAccountClone— a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Wenn auf `true` gesetzt, kann der DB-Cluster kontenübergreifend geklont werden.

- DatabaseName – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Enthält den Namen der initialen Datenbank dieses DB-Clusters, die zum Erstellungszeitpunkt bereitgestellt wurde, sofern eine angegeben wurde, als der DB-Cluster erstellt wurde. Derselbe Name wird über die Lebensdauer des DB-Clusters zurückgegeben.

- DBClusterArn – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Der Amazon-Ressourcenname (ARN) für den DB-Cluster.

- DBClusterIdentifier – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Enthält eine vom Benutzer bereitgestellte DB-Cluster-Kennung. Diese Kennung ist der eindeutige Schlüssel zur Identifizierung eines DB-Clusters.

- DBClusterMembers – Ein Array mit DBClusterMitglied-Objekten.

  Enthält eine Liste der Instances, aus denen der DB-Cluster besteht.

- DBClusterParameterGroup – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Namen der DB-Cluster-Parametergruppe für den DB-Cluster an.

- DbClusterResourceId – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die für die Amazon-Region eindeutige, unveränderliche Kennung für den DB-Cluster. Diese Kennung wird in CloudTrail Amazon-Protokolleinträgen immer dann gefunden, wenn auf den Amazon KMS-Schlüssel für den DB-Cluster zugegriffen wird.

- DBSubnetGroup – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt Informationen zu der Subnetzgruppe an, die dem DB-Cluster zugeordnet ist, einschließlich dem Namen, der Beschreibung und der Subnetze in der Subnetzgruppe.

- DeletionProtection— a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Gibt an, ob der Löschschutz des DB-Clusters aktiviert ist. Die Datenbank kann nicht gelöscht werden, wenn der Löschschutz aktiviert ist.

- EarliestBacktrackTime— a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Offset von Mitternacht 1970-01-01).

  Wird von Neptune nicht unterstützt.

- EarliestRestorableTime— a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Versatz von Mitternacht am 01.01.1970).

  Gibt den frühesten Zeitpunkt an, zu dem eine Datenbank mit point-in-time restore wiederhergestellt werden kann.

- EnabledCloudwatchLogsExports – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Eine Liste der Protokolltypen, für deren Export dieser DB-Cluster in CloudWatch Logs konfiguriert ist. Gültige Protokolltypen sind: `audit` (zum Veröffentlichen von Audit-Logs CloudWatch) und slowquery (zum Veröffentlichen von Slow-Query-Logs). CloudWatch Weitere Informationen finden Sie unter Neptune-Protokolle in CloudWatch Amazon-Protokollen veröffentlichen.

- Endpoint – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Verbindungsendpunkt für die primäre Instance des DB-Clusters an.

- Engine – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Namen der Datenbank-Engine an, die für diesen DB-Cluster verwendet werden soll.

- EngineVersion – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt die Version der Datenbank-Engine an.

- GlobalClusterIdentifier— a GlobalClusterIdentifier, vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge), nicht weniger als 1 oder mehr als 255? st? s, entspricht diesem regulären Ausdruck:`[A-Za-z][0-9A-Za-z-:._]*`.

  Enthält eine vom Benutzer bereitgestellte gelobale Datenbank-Cluster-ID. Diese Kennung ist der eindeutige Schlüssel zur Identifizierung einer globalen Datenbank.

- HostedZoneId – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt die ID an, die Amazon Route 53 zuweist, wenn Sie eine gehostete Zone erstellen.

- IAMDatabaseAuthenticationEnabled – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Dies ist „true“, wenn die Zuweisung von Amazon Identity and Access Management (IAM)-Konten zu Datenbank-Konten aktiviert ist. Andernfalls „false“.

- IOOptimizedNextAllowedModificationTime— a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Offset von Mitternacht 1970-01-01).

  Beim nächsten Mal können Sie den DB-Cluster so ändern, dass der Speichertyp `iopt1` verwendet wird.

- KmsKeyId – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Wenn `StorageEncrypted` „true“ ist, ist dies die Amazon-KMS-Schlüsselkennung für das verschlüsselte DB-Cluster.

- LatestRestorableTime— a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Versatz von Mitternacht am 01.01.1970).

  Gibt den letzten Zeitpunkt an, zu dem eine Datenbank mit point-in-time restore wiederhergestellt werden kann.

- MultiAZ – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Gibt an, ob der DB-Cluster über Instances in mehreren Availability Zones verfügt.

- PendingModifiedValues – Ein ClusterPendingModifiedValues-Objekt.

  Dieser Datentyp wird als Antwortelement in der `ModifyDBCluster` Operation verwendet und enthält Änderungen, die während des nächsten Wartungsfensters angewendet werden.

- PercentProgress – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Fortschritt der Operation als Prozentsatz an.

- Port— ein IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Gibt die Portnummer an, die von der Datenbank-Engine überwacht wird.

- PreferredBackupWindow – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den täglichen Zeitraum in koordinierter Weltzeit (UTC) an, in dem automatische Sicherungen erstellt werden, wenn automatische Sicherungen aktiviert sind, gemäß `BackupRetentionPeriod`.

- PreferredMaintenanceWindow – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den wöchentlichen Zeitraum, in dem Systemwartungen durchgeführt werden können, in UTC (Universal Coordinated Time) an.

- ReaderEndpoint – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Leser-Endpunkt für den DB-Cluster. Der Reader-Endpunkt für ein DB-Cluster gleicht die Verbindungslasten zwischen den Read Replicas aus, die in einem DB-Cluster verfügbar sind. Während Clients neue Verbindungsanfragen an den Reader-Endpunkt tätigen, verteilt Neptune die Verbindungsanfragen zwischen den Read Replicas im DB-Cluster. Diese Funktionalität sorgt dafür, dass die Workload-Auslastung für Lesevorgänge auf mehrere Read Replicas im DB-Cluster verteilt wird.

Wenn ein Failover auftritt und das Read Replica, mit dem Sie verbunden sind, als primäre Instance hochgestuft wird, wird Ihre Verbindung unterbrochen. Verbinden Sie sich erneut mit dem Reader-Endpunkt, um mit dem Senden Ihres Workloads für Lesevorgänge an andere Read Replicas im Cluster fortzufahren.

- ReadReplicaIdentifiers – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Enthält eine oder mehrere Kennungen der mit diesem DB-Cluster verbundenen Read Replicas.

- ReplicationSourceIdentifier – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Wird von Neptune nicht unterstützt.

- ReplicationType – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Wird von Neptune nicht unterstützt.

- ServerlessV2ScalingConfiguration – Ein Serverless V2 ScalingConfigurationInfo-Objekt.

  Zeigt die Skalierungskonfiguration für einen Neptune Serverless DB-Cluster.

  Weitere Informationen finden Sie unter Verwendung von Amazon Neptune Serverless im Amazon Neptune-Benutzerhandbuch.

- Status – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den aktuellen Status dieses DB-Clusters an.

- StorageEncrypted – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Gibt an, ob der DB-Cluster verschlüsselt ist.

- StorageType – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Speichertyp, der vom DB-Cluster verwendet wird.

  Zulässige Werte:

  - **`standard`**— (Standard) Bietet kostengünstigen Datenbankspeicher für Anwendungen mit mäßiger bis geringer I/O Nutzung.
  - **`iopt1`**— Ermöglicht I/O-optimierten Speicher, der auf die Durchsatzanforderungen zugeschnitten ist. I/O-intensive graph workloads that require predictable pricing with low I/O latency and consistent I/O

    E/A-optimierter Neptune-Speicher ist erst ab Engine-Version 1.3.0.0 verfügbar.

- VpcSecurityGroups – Ein Array mit VpcSecurityGroupMembership-Objekten.

  Stellt eine Liste der VPC-Sicherheitsgruppen zur Verfügung, zu denen das DB-Cluster gehört.

Fehler

- DBClusterNotFoundFault
- Ungültig DBCluster StateFault
- Ungültig DBInstance StateFault

Stopp DBCluster (Aktion)

Der AWS CLI-Name für diese API lautet:`stop-db-cluster`.

Stoppt einen Amazon Neptune DB-Cluster. Wenn Sie einen DB-Cluster stoppen, behält Neptune die Metadaten des DB-Clusters einschließlich seiner Endpunkte und DB-Parametergruppen bei.

Neptune speichert auch die Transaktionsprotokolle, sodass Sie bei Bedarf eine point-in-time Wiederherstellung durchführen können.

Anforderung

- DBClusterIdentifier (in der CLI: `--db-cluster-identifier`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die DB-Clusterkennung des Neptune DB-Clusters, der gestoppt werden soll. Dieser Parameter wird als Zeichenfolge in Kleinbuchstaben gespeichert.

Antwort

Enthält die Details eines Amazon Neptune-DB-Clusters.

Dieser Datentyp wird als Antwortelement in der the section called “Beschreiben DBClusters” verwendet.

- AllocatedStorage— ein IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  `AllocatedStorage` gibt immer 1 zurück, da die Neptune-DB-Cluster-Speichergröße nicht fest ist, sondern sich bei Bedarf automatisch anpasst.

- AssociatedRoles – Ein Array mit DBClusterRolle-Objekten.

  Bietet eine Liste der Amazon Identity and Access Management (IAM)-Rollen, die dem DB-Cluster zugeordnet sind. IAM-Rollen, die einem DB-Cluster zugeordnet sind, erteilen dem DB-Cluster die Berechtigung, auf andere Amazon-Services in Ihrem Namen zuzugreifen.
- AutomaticRestartTime— a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Offset von Mitternacht am 01.01.1970).

  Zeitpunkt, zu dem der DB-Cluster automatisch neu gestartet wird.
- AvailabilityZones – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Stellt die Liste der EC2 Availability Zones bereit, in denen Instances im DB-Cluster erstellt werden können.
- BacktrackConsumedChangeRecords— a LongOptional, vom Typ: `long` (eine 64-Bit-Ganzzahl mit Vorzeichen).

  Wird von Neptune nicht unterstützt.
- BacktrackWindow— a LongOptional, vom Typ: `long` (eine 64-Bit-Ganzzahl mit Vorzeichen).

  Wird von Neptune nicht unterstützt.
- BackupRetentionPeriod— an IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Gibt die Anzahl der Tage an, für die automatische DB-Snapshots aufbewahrt werden.
- Capacity— an IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Wird von Neptune nicht unterstützt.
- CloneGroupId – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Identifiziert die Clone-Gruppe, mit der der DB-Cluster verknüpft ist.
- ClusterCreateTime— a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Offset von Mitternacht am 01.01.1970).

  Gibt den Zeitpunkt an, an dem der DB-Cluster erstellt wurde, in UTC (Universal Coordinated Time).
- CopyTagsToSnapshot— a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Wenn auf *`true`* gesetzt, werden Tags in jeden Snapshot des DB-Clusters kopiert, der erstellt wird.

- CrossAccountClone— a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Wenn auf `true` gesetzt, kann der DB-Cluster kontenübergreifend geklont werden.
- DatabaseName – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Enthält den Namen der initialen Datenbank dieses DB-Clusters, die zum Erstellungszeitpunkt bereitgestellt wurde, sofern eine angegeben wurde, als der DB-Cluster erstellt wurde. Derselbe Name wird über die Lebensdauer des DB-Clusters zurückgegeben.
- DBClusterArn – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Amazon-Ressourcenname (ARN) für den DB-Cluster.
- DBClusterIdentifier – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Enthält eine vom Benutzer bereitgestellte DB-Cluster-Kennung. Diese Kennung ist der eindeutige Schlüssel zur Identifizierung eines DB-Clusters.
- DBClusterMembers – Ein Array mit DBClusterMitglied-Objekten.

  Enthält eine Liste der Instances, aus denen der DB-Cluster besteht.
- DBClusterParameterGroup – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Namen der DB-Cluster-Parametergruppe für den DB-Cluster an.
- DbClusterResourceId – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die für die Amazon-Region eindeutige, unveränderliche Kennung für den DB-Cluster. Diese Kennung wird in CloudTrail Amazon-Protokolleinträgen immer dann gefunden, wenn auf den Amazon KMS-Schlüssel für den DB-Cluster zugegriffen wird.
- DBSubnetGroup – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt Informationen zu der Subnetzgruppe an, die dem DB-Cluster zugeordnet ist, einschließlich dem Namen, der Beschreibung und der Subnetze in der Subnetzgruppe.
- DeletionProtection— a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Gibt an, ob der Löschschutz des DB-Clusters aktiviert ist. Die Datenbank kann nicht gelöscht werden, wenn der Löschschutz aktiviert ist.
- EarliestBacktrackTime— a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Offset von Mitternacht 1970-01-01).

Wird von Neptune nicht unterstützt.

- EarliestRestorableTime— a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Versatz von Mitternacht am 01.01.1970).

  Gibt den frühesten Zeitpunkt an, zu dem eine Datenbank mit point-in-time restore wiederhergestellt werden kann.

- EnabledCloudwatchLogsExports – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Liste der Protokolltypen, für deren Export dieser DB-Cluster in CloudWatch Logs konfiguriert ist. Gültige Protokolltypen sind: `audit` (zum Veröffentlichen von Audit-Logs CloudWatch) und slowquery (zum Veröffentlichen von Slow-Query-Logs). CloudWatch Weitere Informationen finden Sie unter Neptune-Protokolle in CloudWatch Amazon-Protokollen veröffentlichen.

- Endpoint – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Verbindungsendpunkt für die primäre Instance des DB-Clusters an.

- Engine – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Namen der Datenbank-Engine an, die für diesen DB-Cluster verwendet werden soll.

- EngineVersion – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt die Version der Datenbank-Engine an.

- GlobalClusterIdentifier— a GlobalClusterIdentifier, vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge), nicht weniger als 1 oder mehr als 255? st? s, entspricht diesem regulären Ausdruck:`[A-Za-z][0-9A-Za-z-:._]*`.

  Enthält eine vom Benutzer bereitgestellte gelobale Datenbank-Cluster-ID. Diese Kennung ist der eindeutige Schlüssel zur Identifizierung einer globalen Datenbank.

- HostedZoneId – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt die ID an, die Amazon Route 53 zuweist, wenn Sie eine gehostete Zone erstellen.

- IAMDatabaseAuthenticationEnabled – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Dies ist „true“, wenn die Zuweisung von Amazon Identity and Access Management (IAM)-Konten zu Datenbank-Konten aktiviert ist. Andernfalls „false“.

- IOOptimizedNextAllowedModificationTime— a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Offset von Mitternacht 1970-01-01).

  Beim nächsten Mal können Sie den DB-Cluster so ändern, dass der Speichertyp `iopt1` verwendet wird.
- KmsKeyId – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Wenn `StorageEncrypted` „true“ ist, ist dies die Amazon-KMS-Schlüsselkennung für das verschlüsselte DB-Cluster.
- LatestRestorableTime— a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Versatz von Mitternacht am 01.01.1970).

  Gibt den letzten Zeitpunkt an, zu dem eine Datenbank mit point-in-time restore wiederhergestellt werden kann.
- MultiAZ – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Gibt an, ob der DB-Cluster über Instances in mehreren Availability Zones verfügt.
- PendingModifiedValues – Ein ClusterPendingModifiedValues-Objekt.

  Dieser Datentyp wird als Antwortelement in der `ModifyDBCluster` Operation verwendet und enthält Änderungen, die während des nächsten Wartungsfensters angewendet werden.
- PercentProgress – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Fortschritt der Operation als Prozentsatz an.
- Port— ein IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Gibt die Portnummer an, die von der Datenbank-Engine überwacht wird.
- PreferredBackupWindow – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den täglichen Zeitraum in koordinierter Weltzeit (UTC) an, in dem automatische Sicherungen erstellt werden, wenn automatische Sicherungen aktiviert sind, gemäß `BackupRetentionPeriod`.
- PreferredMaintenanceWindow – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den wöchentlichen Zeitraum, in dem Systemwartungen durchgeführt werden können, in UTC (Universal Coordinated Time) an.
- ReaderEndpoint – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Der Leser-Endpunkt für den DB-Cluster. Der Reader-Endpunkt für ein DB-Cluster gleicht die Verbindungslasten zwischen den Read Replicas aus, die in einem DB-Cluster verfügbar sind. Während Clients neue Verbindungsanfragen an den Reader-Endpunkt tätigen, verteilt Neptune die Verbindungsanfragen zwischen den Read Replicas im DB-Cluster. Diese Funktionalität sorgt dafür, dass die Workload-Auslastung für Lesevorgänge auf mehrere Read Replicas im DB-Cluster verteilt wird.

Wenn ein Failover auftritt und das Read Replica, mit dem Sie verbunden sind, als primäre Instance hochgestuft wird, wird Ihre Verbindung unterbrochen. Verbinden Sie sich erneut mit dem Reader-Endpunkt, um mit dem Senden Ihres Workloads für Lesevorgänge an andere Read Replicas im Cluster fortzufahren.

- ReadReplicaIdentifiers – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Enthält eine oder mehrere Kennungen der mit diesem DB-Cluster verbundenen Read Replicas.

- ReplicationSourceIdentifier – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Wird von Neptune nicht unterstützt.

- ReplicationType – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Wird von Neptune nicht unterstützt.

- ServerlessV2ScalingConfiguration – Ein Serverless V2 ScalingConfigurationInfo-Objekt.

  Zeigt die Skalierungskonfiguration für einen Neptune Serverless DB-Cluster.

  Weitere Informationen finden Sie unter Verwendung von Amazon Neptune Serverless im Amazon Neptune-Benutzerhandbuch.

- Status – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den aktuellen Status dieses DB-Clusters an.

- StorageEncrypted – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Gibt an, ob der DB-Cluster verschlüsselt ist.

- StorageType – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Speichertyp, der vom DB-Cluster verwendet wird.

  Zulässige Werte:

- **`standard`**— (Standard) Bietet kostengünstigen Datenbankspeicher für Anwendungen mit mäßiger bis geringer I/O Nutzung.
- **`iopt1`**— Ermöglicht I/O-optimierten Speicher, der auf die Durchsatzanforderungen zugeschnitten ist. I/O-intensive graph workloads that require predictable pricing with low I/O latency and consistent I/O

  E/A-optimierter Neptune-Speicher ist erst ab Engine-Version 1.3.0.0 verfügbar.

- VpcSecurityGroups – Ein Array mit VpcSecurityGroupMembership-Objekten.

  Stellt eine Liste der VPC-Sicherheitsgruppen zur Verfügung, zu denen das DB-Cluster gehört.

Fehler

- DBClusterNotFoundFault
- Ungültig DBCluster StateFault
- Ungültig DBInstance StateFault

AddRoleToDBCluster (Aktion)

Der AWS CLI-Name für diese API lautet:`add-role-to-db-cluster`.

Ordnet eine Identity and Access Management (IAM)-Rolle einem Neptune DB-Cluster zu

Anforderung

- DBClusterIdentifier (in der CLI: `--db-cluster-identifier`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Name des DB-Clusters, mit dem die IAM-Rolle verknüpft werden soll.

- FeatureName (in der CLI: `--feature-name`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Name des Features für den Neptune DB-Cluster, dem die IAM-Rolle zugeordnet werden soll. Eine Liste der unterstützten Funktionsnamen finden Sie unter the section called “DBEngineVersion”.

- RoleArn (in der CLI: `--role-arn`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Der Amazon-Ressourcenname (ARN) der IAM-Rolle, der dem Neptune DB-Cluster zugeordnet werden soll, z. B. `arn:aws:iam::123456789012:role/NeptuneAccessRole`.

Antwort

- Keine Antwortparameter.

Fehler

- DBClusterNotFoundFault
- DBClusterRoleAlreadyExistsFault
- Ungültig DBCluster StateFault
- DBClusterRoleQuotaExceededFault

RemoveRoleFromDBCluster (Aktion)

Der AWS CLI-Name für diese API lautet:`remove-role-from-db-cluster`.

Hebt die Zuordnung der Identity and Access Management (IAM)-Rolle zu einem DB-Cluster auf.

Anforderung

- DBClusterIdentifier (in der CLI: `--db-cluster-identifier`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Name des DB-Clusters, dessen Zuordnung zur IAM-Rolle aufgehoben werden soll.
- FeatureName (in der CLI: `--feature-name`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Name des Features für den DB-Cluster, von dem die IAM-Rolle abgekoppelt werden soll. Eine Liste der unterstützten Funktionsnamen finden Sie unter the section called “DBEngineVersionen beschreiben”.
- RoleArn (in der CLI: `--role-arn`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Amazon-Ressourcenname (ARN) der IAM-Rolle, deren Zuordnung zum DB-Cluster aufgehoben werden soll, z. B. `arn:aws:iam::123456789012:role/NeptuneAccessRole`.

Antwort

- Keine Antwortparameter.

Fehler

- DBClusterNotFoundFault
- DBClusterRoleNotFoundFault
- Ungültig DBCluster StateFault

Failover DBCluster (Aktion)

Der AWS CLI-Name für diese API lautet:`failover-db-cluster`.

Erzwingt einen Failover für einen DB-Cluster.

Ein Failover für einen DB-Cluster stuft eine der Read Replicas (Read-Only-Instances) im DB-Cluster zu einer primären Instance (den Cluster-Writer) hoch.

Amazon Neptune führt automatisch einen Failover auf eine Read Replica aus (falls eine existiert), sobald die primäre Instance ausfällt. Sie können ein Failover erzwingen, wenn Sie einen Ausfall einer primären Instance zum Testen simulieren möchten. Da jede Instance in einem DB-Cluster ihre eigene Endpunktadresse hat, müssen Sie alle bestehenden Verbindungen, die diese Endpunktadressen verwenden, bereinigen und wiederherstellen, wenn der Failover abgeschlossen ist.

Anforderung

- DBClusterIdentifier (in der CLI: `--db-cluster-identifier`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine DB-Cluster-Kennung, für die ein Failover erzwungen wird. Bei diesem Parameter wird nicht zwischen Groß- und Kleinschreibung unterschieden.

  Einschränkungen:

  - Muss mit dem Bezeichner einer vorhandenen Datei übereinstimmen DBCluster.
- TargetDBInstanceIdentifier (in der CLI: `--target-db-instance-identifier`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Name der Instance, die zur primären Instance hochgestuft werden soll.

Sie müssen die Instance-Kennung für eine Read Replica im DB-Cluster angeben. Beispiel, `mydbcluster-replica1`.

Antwort

Enthält die Details eines Amazon Neptune-DB-Clusters.

Dieser Datentyp wird als Antwortelement in der the section called “Beschreiben DBClusters” verwendet.

- AllocatedStorage— ein IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  `AllocatedStorage` gibt immer 1 zurück, da die Neptune-DB-Cluster-Speichergröße nicht fest ist, sondern sich bei Bedarf automatisch anpasst.
- AssociatedRoles – Ein Array mit DBClusterRolle-Objekten.

  Bietet eine Liste der Amazon Identity and Access Management (IAM)-Rollen, die dem DB-Cluster zugeordnet sind. IAM-Rollen, die einem DB-Cluster zugeordnet sind, erteilen dem DB-Cluster die Berechtigung, auf andere Amazon-Services in Ihrem Namen zuzugreifen.
- AutomaticRestartTime— a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Offset von Mitternacht am 01.01.1970).

  Zeitpunkt, zu dem der DB-Cluster automatisch neu gestartet wird.
- AvailabilityZones – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Stellt die Liste der EC2 Availability Zones bereit, in denen Instances im DB-Cluster erstellt werden können.
- BacktrackConsumedChangeRecords— a LongOptional, vom Typ: `long` (eine 64-Bit-Ganzzahl mit Vorzeichen).

  Wird von Neptune nicht unterstützt.
- BacktrackWindow— a LongOptional, vom Typ: `long` (eine 64-Bit-Ganzzahl mit Vorzeichen).

  Wird von Neptune nicht unterstützt.
- BackupRetentionPeriod— an IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Gibt die Anzahl der Tage an, für die automatische DB-Snapshots aufbewahrt werden.

- Capacity— an IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Wird von Neptune nicht unterstützt.

- CloneGroupId – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Identifiziert die Clone-Gruppe, mit der der DB-Cluster verknüpft ist.

- ClusterCreateTime— a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Offset von Mitternacht am 01.01.1970).

  Gibt den Zeitpunkt an, an dem der DB-Cluster erstellt wurde, in UTC (Universal Coordinated Time).

- CopyTagsToSnapshot— a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Wenn auf *`true`* gesetzt, werden Tags in jeden Snapshot des DB-Clusters kopiert, der erstellt wird.

- CrossAccountClone— a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Wenn auf `true` gesetzt, kann der DB-Cluster kontenübergreifend geklont werden.

- DatabaseName – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Enthält den Namen der initialen Datenbank dieses DB-Clusters, die zum Erstellungszeitpunkt bereitgestellt wurde, sofern eine angegeben wurde, als der DB-Cluster erstellt wurde. Derselbe Name wird über die Lebensdauer des DB-Clusters zurückgegeben.

- DBClusterArn – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Amazon-Ressourcenname (ARN) für den DB-Cluster.

- DBClusterIdentifier – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Enthält eine vom Benutzer bereitgestellte DB-Cluster-Kennung. Diese Kennung ist der eindeutige Schlüssel zur Identifizierung eines DB-Clusters.

- DBClusterMembers – Ein Array mit DBClusterMitglied-Objekten.

  Enthält eine Liste der Instances, aus denen der DB-Cluster besteht.

- DBClusterParameterGroup – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Namen der DB-Cluster-Parametergruppe für den DB-Cluster an.

- DbClusterResourceId – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Die für die Amazon-Region eindeutige, unveränderliche Kennung für den DB-Cluster. Diese Kennung wird in CloudTrail Amazon-Protokolleinträgen immer dann gefunden, wenn auf den Amazon KMS-Schlüssel für den DB-Cluster zugegriffen wird.

- DBSubnetGroup – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt Informationen zu der Subnetzgruppe an, die dem DB-Cluster zugeordnet ist, einschließlich dem Namen, der Beschreibung und der Subnetze in der Subnetzgruppe.

- DeletionProtection— a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Gibt an, ob der Löschschutz des DB-Clusters aktiviert ist. Die Datenbank kann nicht gelöscht werden, wenn der Löschschutz aktiviert ist.

- EarliestBacktrackTime— a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Offset von Mitternacht 1970-01-01).

  Wird von Neptune nicht unterstützt.

- EarliestRestorableTime— a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Versatz von Mitternacht am 01.01.1970).

  Gibt den frühesten Zeitpunkt an, zu dem eine Datenbank mit point-in-time restore wiederhergestellt werden kann.

- EnabledCloudwatchLogsExports – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Liste der Protokolltypen, für deren Export dieser DB-Cluster in CloudWatch Logs konfiguriert ist. Gültige Protokolltypen sind: `audit` (zum Veröffentlichen von Audit-Logs CloudWatch) und slowquery (zum Veröffentlichen von Slow-Query-Logs). CloudWatch Weitere Informationen finden Sie unter Neptune-Protokolle in CloudWatch Amazon-Protokollen veröffentlichen.

- Endpoint – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Verbindungsendpunkt für die primäre Instance des DB-Clusters an.

- Engine – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Namen der Datenbank-Engine an, die für diesen DB-Cluster verwendet werden soll.

- EngineVersion – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt die Version der Datenbank-Engine an.

- GlobalClusterIdentifier— a GlobalClusterIdentifier, vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge), nicht weniger als 1 oder mehr als 255? st? s, entspricht diesem regulären Ausdruck:`[A-Za-z][0-9A-Za-z-:._]*`.

  Enthält eine vom Benutzer bereitgestellte gelobale Datenbank-Cluster-ID. Diese Kennung ist der eindeutige Schlüssel zur Identifizierung einer globalen Datenbank.
- HostedZoneId – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt die ID an, die Amazon Route 53 zuweist, wenn Sie eine gehostete Zone erstellen.
- IAMDatabaseAuthenticationEnabled – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Dies ist „true“, wenn die Zuweisung von Amazon Identity and Access Management (IAM)-Konten zu Datenbank-Konten aktiviert ist. Andernfalls „false“.
- IOOptimizedNextAllowedModificationTime— a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Offset von Mitternacht 1970-01-01).

  Beim nächsten Mal können Sie den DB-Cluster so ändern, dass der Speichertyp `iopt1` verwendet wird.
- KmsKeyId – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Wenn `StorageEncrypted` „true“ ist, ist dies die Amazon-KMS-Schlüsselkennung für das verschlüsselte DB-Cluster.
- LatestRestorableTime— a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Versatz von Mitternacht am 01.01.1970).

  Gibt den letzten Zeitpunkt an, zu dem eine Datenbank mit point-in-time restore wiederhergestellt werden kann.
- MultiAZ – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Gibt an, ob der DB-Cluster über Instances in mehreren Availability Zones verfügt.
- PendingModifiedValues – Ein ClusterPendingModifiedValues-Objekt.

  Dieser Datentyp wird als Antwortelement in der `ModifyDBCluster` Operation verwendet und enthält Änderungen, die während des nächsten Wartungsfensters angewendet werden.
- PercentProgress – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Fortschritt der Operation als Prozentsatz an.

- Port— ein IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Gibt die Portnummer an, die von der Datenbank-Engine überwacht wird.

- PreferredBackupWindow – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den täglichen Zeitraum in koordinierter Weltzeit (UTC) an, in dem automatische Sicherungen erstellt werden, wenn automatische Sicherungen aktiviert sind, gemäß `BackupRetentionPeriod`.

- PreferredMaintenanceWindow – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den wöchentlichen Zeitraum, in dem Systemwartungen durchgeführt werden können, in UTC (Universal Coordinated Time) an.

- ReaderEndpoint – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Leser-Endpunkt für den DB-Cluster. Der Reader-Endpunkt für ein DB-Cluster gleicht die Verbindungslasten zwischen den Read Replicas aus, die in einem DB-Cluster verfügbar sind. Während Clients neue Verbindungsanfragen an den Reader-Endpunkt tätigen, verteilt Neptune die Verbindungsanfragen zwischen den Read Replicas im DB-Cluster. Diese Funktionalität sorgt dafür, dass die Workload-Auslastung für Lesevorgänge auf mehrere Read Replicas im DB-Cluster verteilt wird.

  Wenn ein Failover auftritt und das Read Replica, mit dem Sie verbunden sind, als primäre Instance hochgestuft wird, wird Ihre Verbindung unterbrochen. Verbinden Sie sich erneut mit dem Reader-Endpunkt, um mit dem Senden Ihres Workloads für Lesevorgänge an andere Read Replicas im Cluster fortzufahren.

- ReadReplicaIdentifiers – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Enthält eine oder mehrere Kennungen der mit diesem DB-Cluster verbundenen Read Replicas.

- ReplicationSourceIdentifier – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Wird von Neptune nicht unterstützt.

- ReplicationType – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Wird von Neptune nicht unterstützt.

- ServerlessV2ScalingConfiguration – Ein Serverless V2 ScalingConfigurationInfo-Objekt.

  Zeigt die Skalierungskonfiguration für einen Neptune Serverless DB-Cluster.

Weitere Informationen finden Sie unter Verwendung von Amazon Neptune Serverless im Amazon Neptune-Benutzerhandbuch.

- Status – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den aktuellen Status dieses DB-Clusters an.

- StorageEncrypted – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Gibt an, ob der DB-Cluster verschlüsselt ist.

- StorageType – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Speichertyp, der vom DB-Cluster verwendet wird.

  Zulässige Werte:

  - **`standard`**— (Standard) Bietet kostengünstigen Datenbankspeicher für Anwendungen mit mäßiger bis geringer I/O Nutzung.
  - **`iopt1`**— Ermöglicht I/O-optimierten Speicher, der auf die Durchsatzanforderungen zugeschnitten ist. I/O-intensive graph workloads that require predictable pricing with low I/O latency and consistent I/O

    E/A-optimierter Neptune-Speicher ist erst ab Engine-Version 1.3.0.0 verfügbar.

- VpcSecurityGroups – Ein Array mit VpcSecurityGroupMembership-Objekten.

  Stellt eine Liste der VPC-Sicherheitsgruppen zur Verfügung, zu denen das DB-Cluster gehört.

Fehler

- DBClusterNotFoundFault
- Ungültig DBCluster StateFault
- Ungültig DBInstance StateFault

PromoteReadReplicaDBCluster (Aktion)

Der AWS CLI-Name für diese API lautet:`promote-read-replica-db-cluster`.

Nicht unterstützt

Anforderung

- DBClusterIdentifier (in der CLI: `--db-cluster-identifier`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Nicht unterstützt

Antwort

Enthält die Details eines Amazon Neptune-DB-Clusters.

Dieser Datentyp wird als Antwortelement in der the section called “Beschreiben DBClusters” verwendet.

- AllocatedStorage— ein IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  `AllocatedStorage` gibt immer 1 zurück, da die Neptune-DB-Cluster-Speichergröße nicht fest ist, sondern sich bei Bedarf automatisch anpasst.
- AssociatedRoles – Ein Array mit DBClusterRolle-Objekten.

  Bietet eine Liste der Amazon Identity and Access Management (IAM)-Rollen, die dem DB-Cluster zugeordnet sind. IAM-Rollen, die einem DB-Cluster zugeordnet sind, erteilen dem DB-Cluster die Berechtigung, auf andere Amazon-Services in Ihrem Namen zuzugreifen.
- AutomaticRestartTime— a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Offset von Mitternacht am 01.01.1970).

  Zeitpunkt, zu dem der DB-Cluster automatisch neu gestartet wird.
- AvailabilityZones – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Stellt die Liste der EC2 Availability Zones bereit, in denen Instances im DB-Cluster erstellt werden können.
- BacktrackConsumedChangeRecords— a LongOptional, vom Typ: `long` (eine 64-Bit-Ganzzahl mit Vorzeichen).

  Wird von Neptune nicht unterstützt.
- BacktrackWindow— a LongOptional, vom Typ: `long` (eine 64-Bit-Ganzzahl mit Vorzeichen).

  Wird von Neptune nicht unterstützt.
- BackupRetentionPeriod— an IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

Gibt die Anzahl der Tage an, für die automatische DB-Snapshots aufbewahrt werden.

- Capacity— an IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Wird von Neptune nicht unterstützt.

- CloneGroupId – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Identifiziert die Clone-Gruppe, mit der der DB-Cluster verknüpft ist.

- ClusterCreateTime— a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Offset von Mitternacht am 01.01.1970).

  Gibt den Zeitpunkt an, an dem der DB-Cluster erstellt wurde, in UTC (Universal Coordinated Time).

- CopyTagsToSnapshot— a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Wenn auf *`true`* gesetzt, werden Tags in jeden Snapshot des DB-Clusters kopiert, der erstellt wird.

- CrossAccountClone— a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Wenn auf `true` gesetzt, kann der DB-Cluster kontenübergreifend geklont werden.

- DatabaseName – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Enthält den Namen der initialen Datenbank dieses DB-Clusters, die zum Erstellungszeitpunkt bereitgestellt wurde, sofern eine angegeben wurde, als der DB-Cluster erstellt wurde. Derselbe Name wird über die Lebensdauer des DB-Clusters zurückgegeben.

- DBClusterArn – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Amazon-Ressourcenname (ARN) für den DB-Cluster.

- DBClusterIdentifier – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Enthält eine vom Benutzer bereitgestellte DB-Cluster-Kennung. Diese Kennung ist der eindeutige Schlüssel zur Identifizierung eines DB-Clusters.

- DBClusterMembers – Ein Array mit DBClusterMitglied-Objekten.

  Enthält eine Liste der Instances, aus denen der DB-Cluster besteht.

- DBClusterParameterGroup – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Namen der DB-Cluster-Parametergruppe für den DB-Cluster an.

- DbClusterResourceId – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die für die Amazon-Region eindeutige, unveränderliche Kennung für den DB-Cluster. Diese Kennung wird in CloudTrail Amazon-Protokolleinträgen immer dann gefunden, wenn auf den Amazon KMS-Schlüssel für den DB-Cluster zugegriffen wird.

- DBSubnetGroup – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt Informationen zu der Subnetzgruppe an, die dem DB-Cluster zugeordnet ist, einschließlich dem Namen, der Beschreibung und der Subnetze in der Subnetzgruppe.

- DeletionProtection— a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Gibt an, ob der Löschschutz des DB-Clusters aktiviert ist. Die Datenbank kann nicht gelöscht werden, wenn der Löschschutz aktiviert ist.

- EarliestBacktrackTime— a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Offset von Mitternacht 1970-01-01).

  Wird von Neptune nicht unterstützt.

- EarliestRestorableTime— a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Versatz von Mitternacht am 01.01.1970).

  Gibt den frühesten Zeitpunkt an, zu dem eine Datenbank mit point-in-time restore wiederhergestellt werden kann.

- EnabledCloudwatchLogsExports – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Liste der Protokolltypen, für deren Export dieser DB-Cluster in CloudWatch Logs konfiguriert ist. Gültige Protokolltypen sind: `audit` (zum Veröffentlichen von Audit-Logs CloudWatch) und slowquery (zum Veröffentlichen von Slow-Query-Logs). CloudWatch Weitere Informationen finden Sie unter Neptune-Protokolle in CloudWatch Amazon-Protokollen veröffentlichen.

- Endpoint – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Verbindungsendpunkt für die primäre Instance des DB-Clusters an.

- Engine – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Namen der Datenbank-Engine an, die für diesen DB-Cluster verwendet werden soll.

- EngineVersion – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Gibt die Version der Datenbank-Engine an.

- GlobalClusterIdentifier— a GlobalClusterIdentifier, vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge), nicht weniger als 1 oder mehr als 255? st? s, entspricht diesem regulären Ausdruck:`[A-Za-z][0-9A-Za-z-:._]*`.

  Enthält eine vom Benutzer bereitgestellte gelobale Datenbank-Cluster-ID. Diese Kennung ist der eindeutige Schlüssel zur Identifizierung einer globalen Datenbank.
- HostedZoneId – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt die ID an, die Amazon Route 53 zuweist, wenn Sie eine gehostete Zone erstellen.
- IAMDatabaseAuthenticationEnabled – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Dies ist „true“, wenn die Zuweisung von Amazon Identity and Access Management (IAM)-Konten zu Datenbank-Konten aktiviert ist. Andernfalls „false“.
- IOOptimizedNextAllowedModificationTime— a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Offset von Mitternacht 1970-01-01).

  Beim nächsten Mal können Sie den DB-Cluster so ändern, dass der Speichertyp `iopt1` verwendet wird.
- KmsKeyId – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Wenn `StorageEncrypted` „true“ ist, ist dies die Amazon-KMS-Schlüsselkennung für das verschlüsselte DB-Cluster.
- LatestRestorableTime— a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Versatz von Mitternacht am 01.01.1970).

  Gibt den letzten Zeitpunkt an, zu dem eine Datenbank mit point-in-time restore wiederhergestellt werden kann.
- MultiAZ – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Gibt an, ob der DB-Cluster über Instances in mehreren Availability Zones verfügt.
- PendingModifiedValues – Ein ClusterPendingModifiedValues-Objekt.

  Dieser Datentyp wird als Antwortelement in der `ModifyDBCluster` Operation verwendet und enthält Änderungen, die während des nächsten Wartungsfensters angewendet werden.
- PercentProgress – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Gibt den Fortschritt der Operation als Prozentsatz an.

- Port— ein IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Gibt die Portnummer an, die von der Datenbank-Engine überwacht wird.

- PreferredBackupWindow – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den täglichen Zeitraum in koordinierter Weltzeit (UTC) an, in dem automatische Sicherungen erstellt werden, wenn automatische Sicherungen aktiviert sind, gemäß `BackupRetentionPeriod`.

- PreferredMaintenanceWindow – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den wöchentlichen Zeitraum, in dem Systemwartungen durchgeführt werden können, in UTC (Universal Coordinated Time) an.

- ReaderEndpoint – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Leser-Endpunkt für den DB-Cluster. Der Reader-Endpunkt für ein DB-Cluster gleicht die Verbindungslasten zwischen den Read Replicas aus, die in einem DB-Cluster verfügbar sind. Während Clients neue Verbindungsanfragen an den Reader-Endpunkt tätigen, verteilt Neptune die Verbindungsanfragen zwischen den Read Replicas im DB-Cluster. Diese Funktionalität sorgt dafür, dass die Workload-Auslastung für Lesevorgänge auf mehrere Read Replicas im DB-Cluster verteilt wird.

  Wenn ein Failover auftritt und das Read Replica, mit dem Sie verbunden sind, als primäre Instance hochgestuft wird, wird Ihre Verbindung unterbrochen. Verbinden Sie sich erneut mit dem Reader-Endpunkt, um mit dem Senden Ihres Workloads für Lesevorgänge an andere Read Replicas im Cluster fortzufahren.

- ReadReplicaIdentifiers – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Enthält eine oder mehrere Kennungen der mit diesem DB-Cluster verbundenen Read Replicas.

- ReplicationSourceIdentifier – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Wird von Neptune nicht unterstützt.

- ReplicationType – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Wird von Neptune nicht unterstützt.

- ServerlessV2ScalingConfiguration – Ein Serverless V2 ScalingConfigurationInfo-Objekt.

Zeigt die Skalierungskonfiguration für einen Neptune Serverless DB-Cluster.

Weitere Informationen finden Sie unter Verwendung von Amazon Neptune Serverless im Amazon Neptune-Benutzerhandbuch.

- Status – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den aktuellen Status dieses DB-Clusters an.

- StorageEncrypted – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Gibt an, ob der DB-Cluster verschlüsselt ist.

- StorageType – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Speichertyp, der vom DB-Cluster verwendet wird.

  Zulässige Werte:

  - **`standard`**— (Standard) Bietet kostengünstigen Datenbankspeicher für Anwendungen mit mäßiger bis geringer I/O Nutzung.
  - **`iopt1`**— Ermöglicht I/O-optimierten Speicher, der auf die Durchsatzanforderungen zugeschnitten ist. I/O-intensive graph workloads that require predictable pricing with low I/O latency and consistent I/O

    E/A-optimierter Neptune-Speicher ist erst ab Engine-Version 1.3.0.0 verfügbar.

- VpcSecurityGroups – Ein Array mit VpcSecurityGroupMembership-Objekten.

  Stellt eine Liste der VPC-Sicherheitsgruppen zur Verfügung, zu denen das DB-Cluster gehört.

Fehler

- DBClusterNotFoundFault
- Ungültig DBCluster StateFault

Beschreiben DBClusters (Aktion)

Der AWS CLI-Name für diese API lautet:`describe-db-clusters`.

Gibt Informationen zu bereitgestellten DB-Clustern zurück und unterstützt die Paginierung.

> **Note**
>
> Über diese Operation können auch Informationen für Amazon-RDS-Cluster und Amazon-DocDB-Cluster zurückgegeben werden.

Anforderung

- DBClusterIdentifier (in der CLI: `--db-cluster-identifier`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die vom Benutzer angegebene Kennung für den DB-Cluster. Wenn dieser Parameter angegeben ist, werden nur die Daten aus dem spezifischen DB-Cluster zurückgegeben. Bei diesem Parameter wird nicht zwischen Groß- und Kleinschreibung unterschieden.

  Einschränkungen:

  - Falls angegeben, muss er mit einem vorhandenen DBCluster Bezeichner übereinstimmen.
- Filters (in der CLI: `--filters`) – Ein Array von Filter Objekten.

  Ein Filter, der einen oder mehrere DB-Cluster zur Beschreibung angibt.

  Unterstützte Filter:

  - `db-cluster-id`- Akzeptiert DB-Cluster-Identifikatoren und DB-Cluster-Amazon-Ressourcennamen (ARNs). Die Ergebnisliste wird nur Informationen über die DB-Cluster enthalten, die durch diese ARNs identifiziert wurden.
  - `engine` – Akzeptiert einen Engine-Namen (z. B. `neptune`) und beschränkt die Ergebnisliste auf DB-Cluster, die von dieser Engine erstellt wurden.

  Um diese API beispielsweise über die Amazon CLI aufzurufen und so zu filtern, dass nur Neptune-DB-Cluster zurückgegeben werden, können Sie den folgenden Befehl verwenden:

  Example

  ```
  aws neptune describe-db-clusters \
              --filters  Name=engine,Values=neptune
  ```
- Marker (in der CLI: `--marker`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Ein optionales Paginierungstoken, das von einer vorherigen the section called “Beschreiben DBClusters”-Anforderung bereitgestellt wird. Wenn Sie diesen Parameter angeben, enthält die Antwort nur die Datensätze zwischen der Markierung und dem durch `MaxRecords` angegebenen Wert.

- MaxRecords(in der CLI:`--max-records`) — ein IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Die maximale Anzahl der in der Antwort zurückgegebenen Datensätze. Wenn mehrere Datensätze vorhanden sind, als der Wert `MaxRecords` angibt, ist ein Paginierungstoken mit dem Namen einer Markierung in der Antwort enthalten, sodass die verbleibenden Ergebnisse abgerufen werden können.

  Standard: 100

  Einschränkungen: Mindestwert 20, Höchstwert 100.

Antwort

- DBClusters – Ein Array mit DBCluster-Objekten.

  Enthält eine Liste der DB-Cluster für den Benutzer.

- Marker – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Ein Paginierungstoken, das in einer nachfolgenden DBClusters Describe-Anfrage verwendet werden kann.

Fehler

- DBClusterNotFoundFault

Strukturen:

DBCluster (Struktur)

Enthält die Details eines Amazon Neptune-DB-Clusters.

Dieser Datentyp wird als Antwortelement in der the section called “Beschreiben DBClusters” verwendet.

Felder

- AllocatedStorage— Dies ist eine IntegerOptional 32-Bit-Ganzzahl vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  `AllocatedStorage` gibt immer 1 zurück, da die Neptune-DB-Cluster-Speichergröße nicht fest ist, sondern sich bei Bedarf automatisch anpasst.
- AssociatedRoles – Dies ist ein Array von DBClusterRolle-Objekten.

  Bietet eine Liste der Amazon Identity and Access Management (IAM)-Rollen, die dem DB-Cluster zugeordnet sind. IAM-Rollen, die einem DB-Cluster zugeordnet sind, erteilen dem DB-Cluster die Berechtigung, auf andere Amazon-Services in Ihrem Namen zuzugreifen.
- AutomaticRestartTime— Dies ist ein TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Offset von Mitternacht 1970-01-01).

  Zeitpunkt, zu dem der DB-Cluster automatisch neu gestartet wird.
- AvailabilityZones – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Stellt die Liste der EC2 Availability Zones bereit, in denen Instances im DB-Cluster erstellt werden können.
- BacktrackConsumedChangeRecords— Dies ist ein LongOptional Wert vom Typ: `long` (eine 64-Bit-Ganzzahl mit Vorzeichen).

  Wird von Neptune nicht unterstützt.
- BacktrackWindow— Dies ist eine LongOptional, vom Typ: `long` (eine 64-Bit-Ganzzahl mit Vorzeichen).

  Wird von Neptune nicht unterstützt.
- BackupRetentionPeriod— Dies ist eine IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Gibt die Anzahl der Tage an, für die automatische DB-Snapshots aufbewahrt werden.
- Capacity— Dies ist eine IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

Wird von Neptune nicht unterstützt.

- CloneGroupId – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Identifiziert die Clone-Gruppe, mit der der DB-Cluster verknüpft ist.

- ClusterCreateTime— Dies ist ein TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Offset von Mitternacht 1970-01-01).

  Gibt den Zeitpunkt an, an dem der DB-Cluster erstellt wurde, in UTC (Universal Coordinated Time).

- CopyTagsToSnapshot— Dies ist ein BooleanOptional Wert vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Wenn auf *`true`* gesetzt, werden Tags in jeden Snapshot des DB-Clusters kopiert, der erstellt wird.

- CrossAccountClone— Dies ist ein BooleanOptional Wert vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Wenn auf `true` gesetzt, kann der DB-Cluster kontenübergreifend geklont werden.

- DatabaseName – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Enthält den Namen der initialen Datenbank dieses DB-Clusters, die zum Erstellungszeitpunkt bereitgestellt wurde, sofern eine angegeben wurde, als der DB-Cluster erstellt wurde. Derselbe Name wird über die Lebensdauer des DB-Clusters zurückgegeben.

- DBClusterArn – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Amazon-Ressourcenname (ARN) für den DB-Cluster.

- DBClusterIdentifier – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Enthält eine vom Benutzer bereitgestellte DB-Cluster-Kennung. Diese Kennung ist der eindeutige Schlüssel zur Identifizierung eines DB-Clusters.

- DBClusterMembers – Dies ist ein Array von DBClusterMitglied-Objekten.

  Enthält eine Liste der Instances, aus denen der DB-Cluster besteht.

- DBClusterParameterGroup – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Namen der DB-Cluster-Parametergruppe für den DB-Cluster an.

- DbClusterResourceId – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Die für die Amazon-Region eindeutige, unveränderliche Kennung für den DB-Cluster. Diese Kennung wird in CloudTrail Amazon-Protokolleinträgen immer dann gefunden, wenn auf den Amazon KMS-Schlüssel für den DB-Cluster zugegriffen wird.

- DBSubnetGroup – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt Informationen zu der Subnetzgruppe an, die dem DB-Cluster zugeordnet ist, einschließlich dem Namen, der Beschreibung und der Subnetze in der Subnetzgruppe.

- DeletionProtection— Dies ist ein BooleanOptional Wert vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Gibt an, ob der Löschschutz des DB-Clusters aktiviert ist. Die Datenbank kann nicht gelöscht werden, wenn der Löschschutz aktiviert ist.

- EarliestBacktrackTime— Dies ist ein Wert TStamp vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Offset von Mitternacht 1970-01-01).

  Wird von Neptune nicht unterstützt.

- EarliestRestorableTime— Dies ist ein TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als ein Offset von Mitternacht 1970-01-01).

  Gibt den frühesten Zeitpunkt an, zu dem eine Datenbank mit point-in-time restore wiederhergestellt werden kann.

- EnabledCloudwatchLogsExports – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Liste der Protokolltypen, für deren Export dieser DB-Cluster in CloudWatch Logs konfiguriert ist. Gültige Protokolltypen sind: `audit` (zum Veröffentlichen von Audit-Logs CloudWatch) und slowquery (zum Veröffentlichen von Slow-Query-Logs). CloudWatch Weitere Informationen finden Sie unter Neptune-Protokolle in CloudWatch Amazon-Protokollen veröffentlichen.

- Endpoint – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Verbindungsendpunkt für die primäre Instance des DB-Clusters an.

- Engine – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Namen der Datenbank-Engine an, die für diesen DB-Cluster verwendet werden soll.

- EngineVersion – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt die Version der Datenbank-Engine an.

- GlobalClusterIdentifier— Dies ist eine GlobalClusterIdentifier Zeichenfolge vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge), nicht weniger als 1 oder mehr als 255? st? s, entspricht diesem regulären Ausdruck:`[A-Za-z][0-9A-Za-z-:._]*`.

  Enthält eine vom Benutzer bereitgestellte gelobale Datenbank-Cluster-ID. Diese Kennung ist der eindeutige Schlüssel zur Identifizierung einer globalen Datenbank.
- HostedZoneId – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt die ID an, die Amazon Route 53 zuweist, wenn Sie eine gehostete Zone erstellen.
- IAMDatabaseAuthenticationEnabled – Dies ist ein boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Dies ist „true“, wenn die Zuweisung von Amazon Identity and Access Management (IAM)-Konten zu Datenbank-Konten aktiviert ist. Andernfalls „false“.
- IOOptimizedNextAllowedModificationTime— Das ist ein TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Offset von Mitternacht 1970-01-01).

  Beim nächsten Mal können Sie den DB-Cluster so ändern, dass der Speichertyp `iopt1` verwendet wird.
- KmsKeyId – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Wenn `StorageEncrypted` „true“ ist, ist dies die Amazon-KMS-Schlüsselkennung für das verschlüsselte DB-Cluster.
- LatestRestorableTime— Dies ist ein TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als ein Offset von Mitternacht 1970-01-01).

  Gibt den letzten Zeitpunkt an, zu dem eine Datenbank mit point-in-time restore wiederhergestellt werden kann.
- MultiAZ – Dies ist ein boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Gibt an, ob der DB-Cluster über Instances in mehreren Availability Zones verfügt.
- PendingModifiedValues – Dies ist ein ClusterPendingModifiedValues-Objekt.

  Dieser Datentyp wird als Antwortelement in der `ModifyDBCluster` Operation verwendet und enthält Änderungen, die während des nächsten Wartungsfensters angewendet werden.
- PercentProgress – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Fortschritt der Operation als Prozentsatz an.

- Port— Dies ist ein IntegerOptional Wert vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Gibt die Portnummer an, die von der Datenbank-Engine überwacht wird.

- PreferredBackupWindow – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den täglichen Zeitraum in koordinierter Weltzeit (UTC) an, in dem automatische Sicherungen erstellt werden, wenn automatische Sicherungen aktiviert sind, gemäß `BackupRetentionPeriod`.

- PreferredMaintenanceWindow – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den wöchentlichen Zeitraum, in dem Systemwartungen durchgeführt werden können, in UTC (Universal Coordinated Time) an.

- ReaderEndpoint – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Leser-Endpunkt für den DB-Cluster. Der Reader-Endpunkt für ein DB-Cluster gleicht die Verbindungslasten zwischen den Read Replicas aus, die in einem DB-Cluster verfügbar sind. Während Clients neue Verbindungsanfragen an den Reader-Endpunkt tätigen, verteilt Neptune die Verbindungsanfragen zwischen den Read Replicas im DB-Cluster. Diese Funktionalität sorgt dafür, dass die Workload-Auslastung für Lesevorgänge auf mehrere Read Replicas im DB-Cluster verteilt wird.

  Wenn ein Failover auftritt und das Read Replica, mit dem Sie verbunden sind, als primäre Instance hochgestuft wird, wird Ihre Verbindung unterbrochen. Verbinden Sie sich erneut mit dem Reader-Endpunkt, um mit dem Senden Ihres Workloads für Lesevorgänge an andere Read Replicas im Cluster fortzufahren.

- ReadReplicaIdentifiers – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Enthält eine oder mehrere Kennungen der mit diesem DB-Cluster verbundenen Read Replicas.

- ReplicationSourceIdentifier – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Wird von Neptune nicht unterstützt.

- ReplicationType – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Wird von Neptune nicht unterstützt.

- ServerlessV2ScalingConfiguration – Dies ist ein Serverless V2 ScalingConfigurationInfo-Objekt.

  Zeigt die Skalierungskonfiguration für einen Neptune Serverless DB-Cluster.

  Weitere Informationen finden Sie unter Verwendung von Amazon Neptune Serverless im Amazon Neptune-Benutzerhandbuch.
- Status – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den aktuellen Status dieses DB-Clusters an.
- StorageEncrypted – Dies ist ein boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Gibt an, ob der DB-Cluster verschlüsselt ist.
- StorageType – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Speichertyp, der vom DB-Cluster verwendet wird.

  Zulässige Werte:
  - **`standard`**— (Standardeinstellung) Bietet kostengünstigen Datenbankspeicher für Anwendungen mit mäßiger bis geringer I/O Nutzung.
  - **`iopt1`**— Ermöglicht I/O-optimierten Speicher, der auf die Durchsatzanforderungen zugeschnitten ist. I/O-intensive graph workloads that require predictable pricing with low I/O latency and consistent I/O

    E/A-optimierter Neptune-Speicher ist erst ab Engine-Version 1.3.0.0 verfügbar.
- VpcSecurityGroups – Dies ist ein Array von VpcSecurityGroupMembership-Objekten.

  Stellt eine Liste der VPC-Sicherheitsgruppen zur Verfügung, zu denen das DB-Cluster gehört.

`DBCluster` wird als Antwortelement verwendet für:

- Erstellen DBCluster
- Löschen DBCluster
- Failover DBCluster
- Modifizieren DBCluster
- PromoteReadReplicaDBCluster
- Wiederherstellen DBCluster FromSnapshot

- Wiederherstellen DBCluster ToPointInTime
- Starten DBCluster
- Stopp DBCluster

DBClusterMitglied (Struktur)

Enthält Informationen zu einer Instance, die Teil eines DB-Clusters ist.

Felder

- DBClusterParameterGroupStatus – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Status der DB-Cluster-Parametergruppe für dieses Mitglied des DB-Clusters an.
- DBInstanceIdentifier – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt die Instance-Kennung für dieses Mitglied des DB-Clusters an.
- IsClusterWriter – Dies ist ein boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Wert, der `true` lautet, wenn das Cluster-Mitglied die primäre Instance für den DB-Cluster ist. Andernfalls lautet der Wert `false`.
- PromotionTier— Dies ist ein IntegerOptional Wert vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Ein Wert, der die Reihenfolge angibt, in der eine Read Replica zur primären Instance hochgestuft wird, wenn die vorhandene primäre Instance ausfällt.

DBClusterRolle (Struktur)

Beschreibt eine Amazon Identity and Access Management (IAM)-Rolle, die dem DB-Cluster zugeordnet ist.

Felder

- FeatureName – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Der Name der Funktion, die der Amazon Identity and Access Management (IAM)-Rolle zugeordnet ist. Eine Liste der unterstützten Funktionsnamen finden Sie unter the section called “DBEngineVersionen beschreiben”.

- RoleArn – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Amazon-Ressourcenname (ARN) der IAM-Rolle, die dem DB-Cluster zugeordnet ist.

- Status – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Beschreibt den Status der Zuordnung zwischen der IAM-Rolle und dem DB-Cluster. Die Statuseigenschaft kann einen der folgenden Werte annehmen:

  - `ACTIVE` – Der IAM-Rollen-ARN wird dem DB-Cluster zugeordnet und kann für den Zugriff auf andere Amazon-Services in Ihrem Auftrag verwendet werden.
  - `PENDING` – Der IAM-Rollen-ARN wird dem DB-Cluster zugeordnet.
  - `INVALID` – Der IAM-Rollen-ARN wird dem DB-Cluster zugeordnet, der DB-Cluster ist jedoch nicht in der Lage, die IAM-Rolle anzunehmen, um auf andere Amazon-Services in Ihrem Auftrag zuzugreifen.

CloudwatchLogsExportConfiguration (Struktur)

Die Konfigurationseinstellung für die Protokolltypen, die für den Export in CloudWatch Logs für eine bestimmte DB-Instance oder einen bestimmten DB-Cluster aktiviert werden sollen.

Die `DisableLogTypes` Arrays `EnableLogTypes` und bestimmen, welche Protokolle in Logs exportiert (oder nicht exportiert) werden CloudWatch.

Gültige Protokolltypen sind: `audit` (zum Veröffentlichen von Audit-Logs) und `slowquery` (zum Veröffentlichen von Logs mit langsamen Abfragen). Weitere Informationen finden Sie unter Neptune-Protokolle in CloudWatch Amazon-Protokollen veröffentlichen.

Felder

- DisableLogTypes – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Liste der Protokolltypen, die deaktiviert werden sollen.

- EnableLogTypes – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Liste der Protokolltypen, die aktiviert werden sollen.

PendingCloudwatchLogsExports (Struktur)

Eine Liste der Protokolltypen, deren Konfiguration noch aussteht. Mit anderen Worten, diese Protokolltypen werden derzeit aktiviert oder deaktiviert.

Gültige Protokolltypen sind: `audit` (zum Veröffentlichen von Audit-Logs) und `slowquery` (zum Veröffentlichen von Logs mit langsamen Abfragen). Weitere Informationen finden Sie unter Neptune-Protokolle in CloudWatch Amazon-Protokollen veröffentlichen.

Felder

- LogTypesToDisable – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Protokolltypen, die gerade aktiviert werden. Nachdem sie aktiviert wurden, werden diese Protokolltypen in Logs exportiert CloudWatch .
- LogTypesToEnable – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Protokolltypen, die gerade deaktiviert werden. Nachdem sie deaktiviert wurden, werden diese Protokolltypen nicht in CloudWatch Logs exportiert.

ClusterPendingModifiedValues (Struktur)

Dieser Datentyp wird als Antwortelement in der `ModifyDBCluster` Operation verwendet und enthält Änderungen, die während des nächsten Wartungsfensters angewendet werden.

Felder

- AllocatedStorage— Dies ist eine IntegerOptional 32-Bit-Ganzzahl vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Die zugewiesene Speichergröße in Gibibyte (GiB) für Datenbank-Engines. Für Neptune gibt `AllocatedStorage` immer 1 zurück, da die Neptune-DB-Cluster-Speichergröße nicht fest ist, sondern sich bei Bedarf automatisch anpasst.
- BackupRetentionPeriod— Dies ist eine IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Die Anzahl der Tage, für die automatische DB-Snapshots aufbewahrt werden.
- DBClusterIdentifier – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Der DBCluster Identifier-Wert für den DB-Cluster.

- EngineVersion – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Version der Datenbank-Engine.

- IAMDatabaseAuthenticationEnabled— Dies ist ein BooleanOptional Wert vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Ein Wert, der angibt, ob die Zuordnung von AWS Identity and Access Management (IAM) -Konten zu Datenbankkonten aktiviert ist.

- Iops— Dies ist ein IntegerOptional Wert vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Der bereitgestellte IOPS (E/A-Operationen pro Sekunde)-Wert. Diese Einstellung gilt nur für Multi-AZ-DB-Cluster.

- PendingCloudwatchLogsExports – Dies ist ein PendingCloudwatchLogsExports-Objekt.

  Diese `PendingCloudwatchLogsExports` Struktur spezifiziert ausstehende Änderungen, welche CloudWatch Protokolle aktiviert und welche deaktiviert sind.

- StorageType – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die ausstehende Änderung des Speichertyps für den DB-Cluster. Zulässige Werte:

  - **`standard`**— (Standard) Konfiguriert kostengünstigen Datenbankspeicher für Anwendungen mit mäßiger bis geringer I/O Nutzung.
  - **`iopt1`**— Ermöglicht I/O-optimierten Speicher, der auf die Durchsatzanforderungen zugeschnitten ist. I/O-intensive graph workloads that require predictable pricing with low I/O latency and consistent I/O

    E/A-optimierter Neptune-Speicher ist erst ab Engine-Version 1.3.0.0 verfügbar.

Neptune Global Database-API

Aktionen:

- CreateGlobalCluster (Aktion)
- DeleteGlobalCluster (Aktion)
- ModifyGlobalCluster (Aktion)
- DescribeGlobalClusters (Aktion)

- FailoverGlobalCluster (Aktion)
- RemoveFromGlobalCluster (Aktion)

Strukturen:

- GlobalCluster (Struktur)
- GlobalClusterMember (Struktur)

CreateGlobalCluster (Aktion)

Der AWS CLI-Name für diese API lautet:`create-global-cluster`.

Erstellt eine globale Neptune-Datenbank, die über mehrere Amazon-Regionen verteilt ist. Die globale Datenbank enthält einen einzelnen primären Cluster mit Lese-Schreib-Funktion und schreibgeschützte sekundäre Cluster, die durch schnelle Replikation des Neptune-Speichersubsystems Daten vom primären Cluster empfangen.

Sie können eine leere globale Datenbank erstellen und dann einen primären Cluster und sekundäre Cluster hinzufügen, oder Sie können während des Erstellungsvorgangs einen vorhandenen Neptune-Cluster angeben, der der primäre Cluster der globalen Datenbank werden soll.

Anforderung

- DatabaseName (in der CLI: `--database-name`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Name für die neue globale Datenbank (bis zu 64 alphanumerische Zeichen).
- DeletionProtection(in der CLI:`--deletion-protection`) — a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Die Löschschutz-Einstellung für die neue globale Datenbank. Die globale Datenbank kann nicht gelöscht werden, wenn der Löschschutz aktiviert ist.
- Engine (in der CLI: `--engine`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Name der Datenbank-Engine, die in der globalen -Datenbank verwendet werden soll.

  Zulässige Werte: `neptune`

- EngineVersion (in der CLI: `--engine-version`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Neptune-Engine-Version, die von der globalen Datenbank verwendet werden soll.

  Gültige Werte: `1.2.0.0` oder höher.

- GlobalClusterIdentifier(in der CLI:`--global-cluster-identifier`) — Erforderlich: a GlobalClusterIdentifier, vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge), nicht weniger als 1 oder mehr als 255? st? s, entspricht diesem regulären Ausdruck:`[A-Za-z][0-9A-Za-z-:._]*`.

  Die Cluster-Kennung des neuen globalen Datenbank-Clusters.

- SourceDBClusterIdentifier (in der CLI: `--source-db-cluster-identifier`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  (Optional) Der Amazon-Ressourcenname (ARN) eines bestehenden Neptune DB-Clusters, der als primärer Cluster der neuen globalen Datenbank verwendet werden soll.

- StorageEncrypted(in der CLI:`--storage-encrypted`) — a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Die Speicherverschlüsselung für den globalen Datenbank-Cluster.

Antwort

Enthält die Details einer globalen Amazon Neptune Neptune-Datenbank.

Dieser Datentyp wird als Antwortelement in den Aktionen the section called “CreateGlobalCluster”, the section called “DescribeGlobalClusters”, the section called “ModifyGlobalCluster”, the section called “DeleteGlobalCluster”, the section called “FailoverGlobalCluster” und the section called “RemoveFromGlobalCluster” verwendet.

- DeletionProtection— a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Die Löschschutz-Einstellung für die globale Datenbank.

- Engine – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Neptune-Datenbank-Engine, die von der globalen Datenbank verwendet wird (`"neptune"`).

- EngineVersion – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Die Neptune-Engine-Version, die von der globalen Datenbank verwendet wird.

- GlobalClusterArn – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Amazon-Ressourcenname (ARN) für die globale Datenbank.

- GlobalClusterIdentifier— a GlobalClusterIdentifier, vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge), nicht weniger als 1 oder mehr als 255? st? s, entspricht diesem regulären Ausdruck:`[A-Za-z][0-9A-Za-z-:._]*`.

  Enthält eine vom Benutzer bereitgestellte gelobale Datenbank-Cluster-ID. Diese Kennung ist der eindeutige Schlüssel zur Identifizierung einer globalen Datenbank.

- GlobalClusterMembers – Ein Array mit GlobalClusterMember-Objekten.

  Eine Liste von Clustern ARNs und Instanzen ARNs für alle DB-Cluster, die Teil der globalen Datenbank sind.

- GlobalClusterResourceId – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine unveränderliche ID für die globale Datenbank, der in allen Regionen einzigartig ist. Diese Kennung befindet CloudTrail sich in Protokolleinträgen, wenn auf den KMS-Schlüssel für den DB-Cluster zugegriffen wird.

- Status – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den aktuellen Status dieser globalen Datenbank an.

- StorageEncrypted— a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Die Speicherverschlüsselung für die globale Datenbank.

Fehler

- GlobalClusterAlreadyExistsFault
- GlobalClusterQuotaExceededFault
- Ungültig DBCluster StateFault
- DBClusterNotFoundFault

DeleteGlobalCluster (Aktion)

Der AWS CLI-Name für diese API lautet:`delete-global-cluster`.

Löscht eine globale Datenbank. Der primäre und alle sekundären Cluster müssen bereits getrennt oder zuerst gelöscht werden.

Anforderung

- GlobalClusterIdentifier(in der CLI:`--global-cluster-identifier`) — Erforderlich: a GlobalClusterIdentifier, vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge), nicht weniger als 1 oder mehr als 255? st? s, entspricht diesem regulären Ausdruck:`[A-Za-z][0-9A-Za-z-:._]*`.

  Die Cluster-Kennung des gelöschten globalen Datenbank-Clusters.

Antwort

Enthält die Details einer globalen Amazon Neptune Neptune-Datenbank.

Dieser Datentyp wird als Antwortelement in den Aktionen the section called “CreateGlobalCluster”, the section called “DescribeGlobalClusters”, the section called “ModifyGlobalCluster”, the section called “DeleteGlobalCluster”, the section called “FailoverGlobalCluster” und the section called “RemoveFromGlobalCluster” verwendet.

- DeletionProtection— a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Die Löschschutz-Einstellung für die globale Datenbank.
- Engine – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Neptune-Datenbank-Engine, die von der globalen Datenbank verwendet wird (`"neptune"`).
- EngineVersion – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Neptune-Engine-Version, die von der globalen Datenbank verwendet wird.
- GlobalClusterArn – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Amazon-Ressourcenname (ARN) für die globale Datenbank.
- GlobalClusterIdentifier— a GlobalClusterIdentifier, vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge), nicht weniger als 1 oder mehr als 255? st? s, entspricht diesem regulären Ausdruck:`[A-Za-z][0-9A-Za-z-:._]*`.

Enthält eine vom Benutzer bereitgestellte gelobale Datenbank-Cluster-ID. Diese Kennung ist der eindeutige Schlüssel zur Identifizierung einer globalen Datenbank.

- GlobalClusterMembers – Ein Array mit GlobalClusterMember-Objekten.

  Eine Liste von Clustern ARNs und Instanzen ARNs für alle DB-Cluster, die Teil der globalen Datenbank sind.

- GlobalClusterResourceId – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine unveränderliche ID für die globale Datenbank, der in allen Regionen einzigartig ist. Diese Kennung befindet CloudTrail sich in Protokolleinträgen, wenn auf den KMS-Schlüssel für den DB-Cluster zugegriffen wird.

- Status – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den aktuellen Status dieser globalen Datenbank an.

- StorageEncrypted— a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Die Speicherverschlüsselung für die globale Datenbank.

Fehler

- GlobalClusterNotFoundFault
- InvalidGlobalClusterStateFault

ModifyGlobalCluster (Aktion)

Der AWS CLI-Name für diese API lautet:`modify-global-cluster`.

Ändert eine Einstellung für einen globalen Amazon Neptune-Cluster Sie können einen oder mehrere Datenbank-Konfigurationsparameter ändern, indem Sie diese Parameter und deren neuen Werte in der Anforderung angeben.

Anforderung

- AllowMajorVersionUpgrade(in der CLI:`--allow-major-version-upgrade`) — a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

Ein Wert, der angibt, ob Major-Versionsupgrades erlaubt sind.

Einschränkungen: Sie müssen Hauptversions-Upgrades zulassen, wenn Sie für den Parameter `EngineVersion` einen Wert angeben, der sich von der aktuellen Hauptversion des DB-Clusters unterscheidet.

Wenn Sie die Hauptversion einer globalen Datenbank aktualisieren, werden die Cluster- und DB-Instance-Parametergruppen auf die Standardparametergruppen für die neue Version gesetzt, sodass Sie nach Abschluss des Upgrades alle benutzerdefinierten Parametergruppen anwenden müssen.

- DeletionProtection(in der CLI:`--deletion-protection`) — a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Gibt an, ob der Löschschutz der globalen Datenbank aktiviert ist. Die globale Datenbank kann nicht gelöscht werden, wenn der Löschschutz aktiviert ist.

- EngineVersion (in der CLI: `--engine-version`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Versionsnummer der Datenbank-Engine, auf die ein Upgrade durchgeführt werden soll. Das Ändern dieses Parameters wird zu einem Nutzungsausfall führen. Die Änderung wird im nächsten Wartungsfenster übernommen, es sei denn, `ApplyImmediately` ist aktiviert.

  Zur Auflistung aller verfügbaren Neptune Engine-Versionen, verwenden Sie den folgenden Befehl:

  Example

  ```
  aws neptune describe-db-engine-versions \
            --engine neptune \
            --query '*[]|[?SupportsGlobalDatabases == 'true'].[EngineVersion]'
  ```

- GlobalClusterIdentifier(in der CLI:`--global-cluster-identifier`) — Erforderlich: a GlobalClusterIdentifier, vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge), nicht weniger als 1 oder mehr als 255? st? s, entspricht diesem regulären Ausdruck:`[A-Za-z][0-9A-Za-z-:._]*`.

  Die DB-Cluster-Kennung des zu ändernden globalen Clusters. Bei diesem Parameter wird nicht zwischen Groß- und Kleinschreibung unterschieden.

  Einschränkungen: Muss mit der Kennung eines vorhandenen globalen Datenbank-Clusters übereinstimmen.

- NewGlobalClusterIdentifier(in der CLI:`--new-global-cluster-identifier`) — a GlobalClusterIdentifier, vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge), nicht weniger als 1 oder mehr als 255? st? s, entspricht diesem regulären Ausdruck:`[A-Za-z][0-9A-Za-z-:._]*`.

  Eine neue Cluster-ID, die der globalen Datenbank zugewiesen werden soll. Dieser Wert wird als Zeichenfolge in Kleinbuchstaben gespeichert.

  Einschränkungen:

  - Muss zwischen 1 und 63 Buchstaben, Ziffern oder Bindestriche enthalten.
  - Das erste Zeichen muss ein Buchstabe sein.
  - Darf nicht mit einem Bindestrich enden oder zwei aufeinanderfolgende Bindestriche enthalten

  Beispiel: `my-cluster2`

Antwort

Enthält die Details einer globalen Amazon Neptune Neptune-Datenbank.

Dieser Datentyp wird als Antwortelement in den Aktionen the section called “CreateGlobalCluster”, the section called “DescribeGlobalClusters”, the section called “ModifyGlobalCluster”, the section called “DeleteGlobalCluster”, the section called “FailoverGlobalCluster” und the section called “RemoveFromGlobalCluster” verwendet.

- DeletionProtection— a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Die Löschschutz-Einstellung für die globale Datenbank.
- Engine – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Neptune-Datenbank-Engine, die von der globalen Datenbank verwendet wird (`"neptune"`).
- EngineVersion – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Neptune-Engine-Version, die von der globalen Datenbank verwendet wird.
- GlobalClusterArn – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Amazon-Ressourcenname (ARN) für die globale Datenbank.

- GlobalClusterIdentifier— a GlobalClusterIdentifier, vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge), nicht weniger als 1 oder mehr als 255? st? s, entspricht diesem regulären Ausdruck:`[A-Za-z][0-9A-Za-z-:._]*`.

  Enthält eine vom Benutzer bereitgestellte gelobale Datenbank-Cluster-ID. Diese Kennung ist der eindeutige Schlüssel zur Identifizierung einer globalen Datenbank.

- GlobalClusterMembers – Ein Array mit GlobalClusterMember-Objekten.

  Eine Liste von Clustern ARNs und Instanzen ARNs für alle DB-Cluster, die Teil der globalen Datenbank sind.

- GlobalClusterResourceId – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine unveränderliche ID für die globale Datenbank, der in allen Regionen einzigartig ist. Diese Kennung befindet CloudTrail sich in Protokolleinträgen, wenn auf den KMS-Schlüssel für den DB-Cluster zugegriffen wird.

- Status – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den aktuellen Status dieser globalen Datenbank an.

- StorageEncrypted— a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Die Speicherverschlüsselung für die globale Datenbank.

Fehler

- GlobalClusterNotFoundFault
- InvalidGlobalClusterStateFault

DescribeGlobalClusters (Aktion)

Der AWS CLI-Name für diese API lautet:`describe-global-clusters`.

Gibt Informationen zu globalen Neptune-Datenbank-Clustern zurück Diese API unterstützt Paginierung.

Anforderung

- GlobalClusterIdentifier(in der CLI:`--global-cluster-identifier`) — a GlobalClusterIdentifier, vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge), nicht weniger als 1 oder mehr als 255? st? s, entspricht diesem regulären Ausdruck:`[A-Za-z][0-9A-Za-z-:._]*`.

  Die vom Benutzer angegebene Kennung für den DB-Cluster. Wenn dieser Parameter angegeben wird, werden nur Informationen über den angegebenen DB-Cluster zurückgegeben. Bei diesem Parameter wird nicht zwischen Groß- und Kleinschreibung unterschieden.

  Beschränkungen: Falls angegeben, muss er mit einer bestehenden DB-Cluster-Kennung übereinstimmen.

- Marker (in der CLI: `--marker`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  (Optional) Ein Paginierungs-Token, der von einem vorherigen Aufruf von `DescribeGlobalClusters` ausgegeben wurde. Wenn Sie diesen Parameter angeben, enthält die Antwort nur die Datensätze zwischen der Markierung und dem durch `MaxRecords` angegebenen Wert.

- MaxRecords(in der CLI:`--max-records`) — ein IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Die maximale Anzahl der in der Antwort zurückgegebenen Datensätze. Wenn mehr Datensätze vorhanden sind als der angegebene `MaxRecords`-Wert, wird ein Paginierungsmarkierungstoken in die Antwort aufgenommen, mit dem Sie die verbleibenden Ergebnisse abrufen können.

  Standard: `100`

  Einschränkungen: Mindestwert 20, Höchstwert 100.

Antwort

- GlobalClusters – Ein Array mit GlobalCluster-Objekten.

  Die Liste der globalen Cluster und Instances, die von dieser Anfrage zurückgegeben wurden.

- Marker – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Ein Paginierungstoken. Wenn dieser Parameter in der Antwort ausgegeben wird, sind mehr Datensätze verfügbar, die durch einen oder mehrere zusätzliche Aufrufe von `DescribeGlobalClusters` abgerufen werden können.

Fehler

- GlobalClusterNotFoundFault

FailoverGlobalCluster (Aktion)

Der AWS CLI-Name für diese API lautet:`failover-global-cluster`.

Startet den Failover-Prozess für eine globale Neptune-Datenbank.

Ein Failover für eine globale Neptune-Datenbank stuft einen der sekundären schreibgeschützten DB-Cluster zum primären DB-Cluster herauf und degradiert den primären DB-Cluster zu einem sekundären (schreibgeschützten) DB-Cluster. Mit anderen Worten, die Rolle des aktuellen primären DB-Clusters und des ausgewählten sekundären Ziel-DB-Clusters werden vertauscht. Der ausgewählte sekundäre DB-Cluster geht von allen read/write Funktionen für die globale Neptune-Datenbank aus.

> **Note**
>
> Diese Aktion gilt nur für globale Neptune-Datenbanken. Diese Aktion ist nur für gesunde globale Neptune-Datenbanken mit intakten Neptune-DB-Clustern und ohne regionale Ausfälle, zum Testen von Notfallwiederherstellungsszenarien oder zur Neukonfiguration der globalen Datenbanktopologie vorgesehen.

Anforderung

- GlobalClusterIdentifier(in der CLI:`--global-cluster-identifier`) — Erforderlich: a GlobalClusterIdentifier, vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge), nicht weniger als 1 oder mehr als 255? st? s, entspricht diesem regulären Ausdruck:`[A-Za-z][0-9A-Za-z-:._]*`.

  Bezeichner der globalen Neptune-Datenbank, für die ein Failover durchgeführt werden soll. Die ID ist der eindeutige Schlüssel, der vom Benutzer bei der Erstellung der globalen Neptune-Datenbank zugewiesen wurde. Mit anderen Worten, es ist der Name der globalen Datenbank, für die Sie ein Failover ausführen möchten.

  Einschränkungen: Muss mit der ID eines vorhandenen globalen Neptune Datenbank-Clusters übereinstimmen.

- TargetDbClusterIdentifier (in der CLI: `--target-db-cluster-identifier`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Amazon-Ressourcennamen (ARN) des sekundären Neptune-DB-Clusters, das Sie als primäres Cluster für die globale -Datenbank hochstufen möchten.

Antwort

Enthält die Details einer globalen Amazon Neptune Neptune-Datenbank.

Dieser Datentyp wird als Antwortelement in den Aktionen the section called “CreateGlobalCluster”, the section called “DescribeGlobalClusters”, the section called “ModifyGlobalCluster”, the section called “DeleteGlobalCluster”, the section called “FailoverGlobalCluster” und the section called “RemoveFromGlobalCluster” verwendet.

- DeletionProtection— a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Die Löschschutz-Einstellung für die globale Datenbank.
- Engine – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Neptune-Datenbank-Engine, die von der globalen Datenbank verwendet wird (`"neptune"`).
- EngineVersion – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Neptune-Engine-Version, die von der globalen Datenbank verwendet wird.
- GlobalClusterArn – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Amazon-Ressourcenname (ARN) für die globale Datenbank.
- GlobalClusterIdentifier— a GlobalClusterIdentifier, vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge), nicht weniger als 1 oder mehr als 255? st? s, entspricht diesem regulären Ausdruck:`[A-Za-z][0-9A-Za-z-:._]*`.

  Enthält eine vom Benutzer bereitgestellte gelobale Datenbank-Cluster-ID. Diese Kennung ist der eindeutige Schlüssel zur Identifizierung einer globalen Datenbank.
- GlobalClusterMembers – Ein Array mit GlobalClusterMember-Objekten.

  Eine Liste von Clustern ARNs und Instanzen ARNs für alle DB-Cluster, die Teil der globalen Datenbank sind.

- GlobalClusterResourceId – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine unveränderliche ID für die globale Datenbank, der in allen Regionen einzigartig ist. Diese Kennung befindet CloudTrail sich in Protokolleinträgen, wenn auf den KMS-Schlüssel für den DB-Cluster zugegriffen wird.
- Status – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den aktuellen Status dieser globalen Datenbank an.
- StorageEncrypted— a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Die Speicherverschlüsselung für die globale Datenbank.

Fehler

- GlobalClusterNotFoundFault
- InvalidGlobalClusterStateFault
- Ungültig DBCluster StateFault
- DBClusterNotFoundFault

RemoveFromGlobalCluster (Aktion)

Der AWS CLI-Name für diese API lautet:`remove-from-global-cluster`.

Trennt einen Neptun-DB-Cluster von einer globalen Neptune-Datenbank. Ein sekundärer Cluster wird zu einem normalen eigenständigen Cluster mit Lese- und Schreibfähigkeit, anstatt nur lesend, und empfängt keine Daten mehr vom primären Cluster.

Anforderung

- DbClusterIdentifier (in der CLI: `--db-cluster-identifier`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Amazon-Ressourcenname (ARN), der den Cluster identifiziert, der vom globalen Neptune-Datenbank-Cluster getrennt werden soll.
- GlobalClusterIdentifier(in der CLI:`--global-cluster-identifier`) — Erforderlich: a GlobalClusterIdentifier, vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge), nicht weniger als 1 oder mehr als 255? st? s, entspricht diesem regulären Ausdruck:`[A-Za-z][0-9A-Za-z-:._]*`.

Die ID der globalen Neptune-Datenbank, von der dem angegebenen Neptune-DB-Cluster getrennt werden soll.

Antwort

Enthält die Details einer globalen Amazon Neptune Neptune-Datenbank.

Dieser Datentyp wird als Antwortelement in den Aktionen the section called “CreateGlobalCluster”, the section called “DescribeGlobalClusters”, the section called “ModifyGlobalCluster”, the section called “DeleteGlobalCluster”, the section called “FailoverGlobalCluster” und the section called “RemoveFromGlobalCluster” verwendet.

- DeletionProtection— a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Die Löschschutz-Einstellung für die globale Datenbank.

- Engine – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Neptune-Datenbank-Engine, die von der globalen Datenbank verwendet wird (`"neptune"`).

- EngineVersion – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Neptune-Engine-Version, die von der globalen Datenbank verwendet wird.

- GlobalClusterArn – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Amazon-Ressourcenname (ARN) für die globale Datenbank.

- GlobalClusterIdentifier— a GlobalClusterIdentifier, vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge), nicht weniger als 1 oder mehr als 255? st? s, entspricht diesem regulären Ausdruck:`[A-Za-z][0-9A-Za-z-:._]*`.

  Enthält eine vom Benutzer bereitgestellte gelobale Datenbank-Cluster-ID. Diese Kennung ist der eindeutige Schlüssel zur Identifizierung einer globalen Datenbank.

- GlobalClusterMembers – Ein Array mit GlobalClusterMember-Objekten.

  Eine Liste von Clustern ARNs und Instanzen ARNs für alle DB-Cluster, die Teil der globalen Datenbank sind.

- GlobalClusterResourceId – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Eine unveränderliche ID für die globale Datenbank, der in allen Regionen einzigartig ist. Diese Kennung befindet CloudTrail sich in Protokolleinträgen, wenn auf den KMS-Schlüssel für den DB-Cluster zugegriffen wird.

- Status – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den aktuellen Status dieser globalen Datenbank an.

- StorageEncrypted— a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Die Speicherverschlüsselung für die globale Datenbank.

Fehler

- GlobalClusterNotFoundFault
- InvalidGlobalClusterStateFault
- DBClusterNotFoundFault

Strukturen:

GlobalCluster (Struktur)

Enthält die Details einer globalen Amazon Neptune Neptune-Datenbank.

Dieser Datentyp wird als Antwortelement in den Aktionen the section called “CreateGlobalCluster”, the section called “DescribeGlobalClusters”, the section called “ModifyGlobalCluster”, the section called “DeleteGlobalCluster”, the section called “FailoverGlobalCluster” und the section called “RemoveFromGlobalCluster” verwendet.

Felder

- DeletionProtection— Dies ist ein BooleanOptional Wert vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Die Löschschutz-Einstellung für die globale Datenbank.

- Engine – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Neptune-Datenbank-Engine, die von der globalen Datenbank verwendet wird (`"neptune"`).

- EngineVersion – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Neptune-Engine-Version, die von der globalen Datenbank verwendet wird.
- GlobalClusterArn – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Amazon-Ressourcenname (ARN) für die globale Datenbank.
- GlobalClusterIdentifier— Dies ist ein GlobalClusterIdentifier Wert vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge), nicht weniger als 1 oder mehr als 255? st? s, entspricht diesem regulären Ausdruck:`[A-Za-z][0-9A-Za-z-:._]*`.

  Enthält eine vom Benutzer bereitgestellte gelobale Datenbank-Cluster-ID. Diese Kennung ist der eindeutige Schlüssel zur Identifizierung einer globalen Datenbank.
- GlobalClusterMembers – Dies ist ein Array von GlobalClusterMember-Objekten.

  Eine Liste von Clustern ARNs und Instanzen ARNs für alle DB-Cluster, die Teil der globalen Datenbank sind.
- GlobalClusterResourceId – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine unveränderliche ID für die globale Datenbank, der in allen Regionen einzigartig ist. Diese Kennung befindet CloudTrail sich in Protokolleinträgen, wenn auf den KMS-Schlüssel für den DB-Cluster zugegriffen wird.
- Status – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den aktuellen Status dieser globalen Datenbank an.
- StorageEncrypted— Dies ist ein BooleanOptional Wert vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Die Speicherverschlüsselung für die globale Datenbank.

`GlobalCluster` wird als Antwortelement verwendet für:

- CreateGlobalCluster
- ModifyGlobalCluster
- DeleteGlobalCluster
- RemoveFromGlobalCluster
- FailoverGlobalCluster

GlobalClusterMember (Struktur)

Eine Datenstruktur mit Informationen über alle primären und sekundären Cluster, die mit einer globalen Neptun-Datenbank verknüpft sind.

Felder

- DBClusterArn – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Amazon-Ressourcenname (ARN) für jeden Neptune-Cluster.
- IsWriter – Dies ist ein boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Gibt an, ob der Neptune-Cluster der primäre Cluster (d. h. Lese- und Schreibfähigkeit) für die globale Neptune-Datenbank ist, mit der er verknüpft ist.
- Readers – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Amazon-Ressourcenname (ARN) für jeden schreibgeschützten sekundären Cluster, der der globalen Neptune-Datenbank zugeordnet ist.

Neptune-Instances-API

Aktionen:

- Erstellen DBInstance (Aktion)
- Löschen DBInstance (Aktion)
- Ändern DBInstance (Aktion)
- Neustart DBInstance (Aktion)
- Beschreiben Sie DBInstances (Aktion)
- DescribeOrderableDBInstanceOptionen (Aktion)
- DescribeValidDBInstanceÄnderungen (Aktion)

Strukturen:

- DBInstance (Struktur)
- DBInstanceStatusInfo (Struktur)
- Bestellbare DBInstance Option (Struktur)

- PendingModifiedValues (Struktur)
- ValidStorageOptions (Struktur)
- Gültig DBInstance ModificationsMessage (Struktur)

Erstellen DBInstance (Aktion)

Der AWS CLI-Name für diese API lautet:`create-db-instance`.

Erstellt eine neue DB-Instance.

Anforderung

- AutoMinorVersionUpgrade(in der CLI:`--auto-minor-version-upgrade`) — a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Gibt an, dass kleinere Engine-Aktualisierungen innerhalb des Wartungsfensters automatisch auf die DB-Instance angewendet werden.

  Standard: `true`
- AvailabilityZone (in der CLI: `--availability-zone`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die EC2 Availability Zone, in der die DB-Instance erstellt wurde

  Standard: Eine zufällig vom System gewählte Availability Zone in der Amazon-Region des Endpunkts.

  Beispiel: `us-east-1d`

  Einschränkung: Der AvailabilityZone Parameter kann nicht angegeben werden, wenn der MultiAZ-Parameter auf `true` gesetzt ist. Die angegebene Availability Zone muss in derselben Amazon-Region sein wie der aktuelle Endpunkt.
- BackupRetentionPeriod(in der CLI:`--backup-retention-period`) — ein IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Die Anzahl von Tagen, über die hinweg automatische Sicherungen aufbewahrt werden.

  Nicht zutreffend. Der Aufbewahrungszeitraum von automatischen Backups wird vom DB-Cluster verwaltet. Weitere Informationen finden Sie unter the section called “Erstellen DBCluster”.

Standard: 1

Einschränkungen:

- Muss ein Wert zwischen 0 und 35 sein
- Kann nicht auf 0 gesetzt werden, wenn die DB-Instance eine Quelle für Read Replicas ist

- CopyTagsToSnapshot(in der CLI:`--copy-tags-to-snapshot`) — a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  "True" zum Kopieren aller Tags aus der DB-Instance in die Snapshots der DB-Instance und andernfalls "false". Der Standardwert lautet „false“.

- DBClusterIdentifier (in der CLI: `--db-cluster-identifier`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die ID des DB-Clusters, zu dem die Instance gehört.

  Weitere Informationen zum Erstellen eines DB-Clusters finden Sie unter the section called “Erstellen DBCluster”.

  Typ: Zeichenfolge

- DBInstanceClass (in der CLI: `--db-instance-class`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Rechen- und Speicherkapazität der DB-Instance, beispielsweise `db.m4.large`. Nicht alle DB-Instance-Klassen stehen in allen Amazon-Regionen zur Verfügung.

- DBInstanceIdentifier (in der CLI: `--db-instance-identifier`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die DB-Instance-Kennung. Dieser Parameter wird als Zeichenfolge in Kleinbuchstaben gespeichert.

  Einschränkungen:

  - Muss zwischen 1 und 63 Buchstaben, Ziffern oder Bindestriche enthalten.
  - Muss mit einem Buchstaben beginnen.
  - Darf nicht mit einem Bindestrich enden oder zwei aufeinanderfolgende Bindestriche enthalten.

  Beispiel: `mydbinstance`

- DBName (in der CLI: `--db-name`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Nicht unterstützt

- DBParameterGroupName (in der CLI: `--db-parameter-group-name`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Name der DB-Parametergruppe, die mit dieser DB-Instance verknüpft werden soll. Wenn dieses Argument weggelassen wird, wird die DBParameter Standardgruppe für die angegebene Engine verwendet.

  Einschränkungen:

  - Muss zwischen 1 und 255 Buchstaben, Zahlen oder Bindestriche enthalten.
  - Muss mit einem Buchstaben beginnen
  - Darf nicht mit einem Bindestrich enden oder zwei aufeinanderfolgende Bindestriche enthalten

- DBSecurityGroups (in der CLI: `--db-security-groups`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Liste der DB-Sicherheitsgruppen, die mit dieser DB-Instance verknüpft werden sollen.

  Standard: Die Standard-DB-Sicherheitsgruppe für die Datenbank-Engine.

- DBSubnetGroupName (in der CLI: `--db-subnet-group-name`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine DB-Subnetzgruppe, welche dieser DB-Instance zugeordnet werden soll.

  Wenn keine DB-Subnetzgruppe vorhanden ist, dann handelt es sich um eine Nicht-VPC-DB-Instance.

- DeletionProtection(in der CLI:`--deletion-protection`) — a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Ein Wert, der angibt, ob der Löschschutz für die DB-Instance aktiviert ist. Die Datenbank kann nicht gelöscht werden, wenn der Löschschutz aktiviert ist. Der Löschschutz ist standardmäßig deaktiviert. Siehe Löschen einer DB-Instance.

  DB-Instances in einem DB-Cluster können auch dann gelöscht werden, wenn der Löschschutz in ihrem übergeordneten DB-Cluster aktiviert ist.

- Domain (in der CLI: `--domain`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Geben Sie die Active Directory-Domäne an, in der die Instance erstellt werden soll.

- DomainIAMRoleName (in der CLI: `--domain-iam-role-name`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Geben Sie den Namen der IAM-Rolle an, die verwendet werden soll, wenn API-Aufrufe im Directory Service ausgerichtet werden.

- EnableCloudwatchLogsExports (in der CLI: `--enable-cloudwatch-logs-exports`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Liste der Protokolltypen, die für den Export in Logs aktiviert werden müssen. CloudWatch

- EnableIAMDatabaseAuthentication(in der CLI:`--enable-iam-database-authentication`) — a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Wird von Neptune nicht unterstützt (ignoriert).

- Engine (in der CLI: `--engine`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Name der Datenbank-Engine, die für diese Instance verwendet werden soll.

  Zulässige Werte: `neptune`

- EngineVersion (in der CLI: `--engine-version`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Versionsnummer der zu verwendenden Datenbank-Engine. Derzeit hat das Festlegen dieses Parameters keine Auswirkungen.

- Iops(in der CLI:`--iops`) — ein IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Die Menge von bereitgestellten IOPS (Ein-/Ausgabeoperationen pro Sekunde), die der DB-Instance anfänglich zugewiesen werden soll.

- KmsKeyId (in der CLI: `--kms-key-id`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die ID des Amazon-KMS-Schlüssels für eine verschlüsselte DB-Instance.

Die Kennung für den KMS-Schlüssel ist der Amazon-Ressourcenname (ARN) für den KMS-Verschlüsselungsschlüssel. Wenn Sie eine DB-Instance mit demselben Amazon-Konto erstellen, das über den KMS-Verschlüsselungsschlüssel verfügt, der für die Verschlüsselung der neuen DB-Instance verwendet wurde, können Sie den KMS-Schlüsselalias anstelle des ARNs für den KM-Verschlüsselungsschlüssel verwenden.

Nicht zutreffend. Die KMS-Schlüssel-ID wird vom DB-Cluster verwaltet. Weitere Informationen finden Sie unter the section called “Erstellen DBCluster”.

Wenn der `StorageEncrypted`-Parameter "true" ist und Sie keinen Wert für den `KmsKeyId`-Parameter angeben, verwendet Amazon Neptune den Standardverschlüsselungsschlüssel. Amazon-KMS erstellt den Standardverschlüsselungsschlüssel für Ihr Amazon-Konto. Ihr Amazon-Konto verfügt für jede Amazon-Region über einen anderen Standardverschlüsselungsschlüssel.

- LicenseModel (in der CLI: `--license-model`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Lizenzmodellinformationen für diese DB-Instance.

  Zulässige Werte: `license-included` | `bring-your-own-license` | `general-public-license`

- MonitoringInterval(in der CLI:`--monitoring-interval`) — ein IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Das Intervall in Sekunden zwischen den Punkten, an denen erweiterte Überwachungsmetriken für die DB-Instance erfasst werden. Um die Erfassung von Metriken für erweiterte Überwachung zu deaktivieren, geben Sie 0 an. Der Standardwert ist 0.

  Wenn `MonitoringRoleArn` angegeben ist, müssen Sie auch `MonitoringInterval` auf einen anderen Wert als 0 festlegen.

  Zulässige Werte: `0, 1, 5, 10, 15, 30, 60`

- MonitoringRoleArn (in der CLI: `--monitoring-role-arn`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der ARN für die IAM-Rolle, der es Neptune ermöglicht, erweiterte Überwachungsmetriken an Amazon Logs zu senden. CloudWatch Beispiel, `arn:aws:iam:123456789012:role/emaccess`.

Wenn `MonitoringInterval` auf einen anderen Wert als 0 festgelegt ist, müssen Sie einen `MonitoringRoleArn`-Wert bereitstellen.

- MultiAZ(in der CLI:`--multi-az`) — a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Gibt an, ob die DB-Instance eine Multi-AZ-Bereitstellung ist. Sie können den AvailabilityZone Parameter nicht festlegen, wenn der MultiAZ-Parameter auf true gesetzt ist.

- Port(in der CLI:`--port`) — ein IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Die Portnummer, an der die Datenbank Verbindungen akzeptiert.

  Nicht zutreffend. Der Port wird vom DB-Cluster verwaltet. Weitere Informationen finden Sie unter the section called “Erstellen DBCluster”.

  Standard: `8182`

  Typ: Ganzzahl

- PreferredBackupWindow (in der CLI: `--preferred-backup-window`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der tägliche Zeitraum, in dem automatische Backups erstellt werden.

  Nicht zutreffend. Der tägliche Zeitraum für das Erstellen von automatischen Backups wird vom DB-Cluster verwaltet. Weitere Informationen finden Sie unter the section called “Erstellen DBCluster”.

- PreferredMaintenanceWindow (in der CLI: `--preferred-maintenance-window`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der wöchentliche Zeitraum, in dem Systemwartungen durchgeführt werden können, in UTC (Universal Coordinated Time).

  Format: `ddd:hh24:mi-ddd:hh24:mi`

  Der Standardwert ist ein 30-minütiges Fenster, das zufällig aus einem 8-Stunden-Zeitraum für jede Amazon-Region an einem zufälligen Wochentag ausgewählt wird.

  Gültige Tage: Mo, Di, Mi, Do, Fr, Sa, So.

  Einschränkungen: mindestens 30-Minuten-Zeitfenster.

- PromotionTier(in der CLI:`--promotion-tier`) — ein IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Ein Wert, der die Reihenfolge angibt, in der eine Read Replica zur primären Instance hochgestuft wird, wenn die vorhandene primäre Instance ausfällt.

  Standard: 1

  Gültige Werte: 0 bis 15
- PubliclyAccessible(in der CLI:`--publicly-accessible`) — a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Dieses Flag sollte nicht mehr verwendet werden.
- StorageEncrypted(in der CLI:`--storage-encrypted`) — a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Gibt an, ob die DB-Instance verschlüsselt wird.

  Nicht zutreffend. Die Verschlüsselung für DB-Instances wird vom DB-Cluster verwaltet. Weitere Informationen finden Sie unter the section called “Erstellen DBCluster”.

  Standard: false
- StorageType (in der CLI: `--storage-type`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Nicht zutreffend. In Neptune wird der Speichertyp auf DB-Cluster-Ebene verwaltet.
- Tags (in der CLI: `--tags`) – Ein Array von Markierung Objekten.

  Die Tags, die der neuen Instance zugeordnet werden sollen.
- TdeCredentialArn (in der CLI: `--tde-credential-arn`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der ARN aus dem Schlüsselspeicher, mit dem die Instance für die TDE-Verschlüsselung verknüpft werden soll.
- TdeCredentialPassword(in der CLI:`--tde-credential-password`) — a SensitiveString, vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

  Das Passwort für den angegebenen ARN aus dem Schlüsselspeicher, um auf das Gerät zuzugreifen.

- Timezone (in der CLI: `--timezone`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Zeitzone der DB-Instance.

- VpcSecurityGroupIds (in der CLI: `--vpc-security-group-ids`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Liste von EC2 VPC-Sicherheitsgruppen, die dieser DB-Instance zugeordnet werden sollen.

  Nicht zutreffend. Die zugehörige Liste der EC2 VPC-Sicherheitsgruppen wird vom DB-Cluster verwaltet. Weitere Informationen finden Sie unter the section called “Erstellen DBCluster”.

  Standard: Die EC2 Standard-VPC-Sicherheitsgruppe für die VPC der DB-Subnetzgruppe.

Antwort

Enthält die Details einer Amazon Neptune-DB-Instance.

Dieser Datentyp wird als Antwortelement in der Aktion the section called “Beschreiben DBInstances” verwendet.

- AutoMinorVersionUpgrade – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Gibt an, dass Patches von Unterversionen automatisch angewendet werden.

- AvailabilityZone – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Namen der Availability Zone an, in der sich die DB-Instance befindet.

- BackupRetentionPeriod – eine Ganzzahl vom Typ `integer` (32-Bit-Ganzzahl mit Vorzeichen).

  Gibt die Anzahl der Tage an, für die automatische DB-Snapshots aufbewahrt werden.

- CACertificateIdentifier – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die ID des Zertifizierungsstellenzertifikats für diese DB-Instance.

- CopyTagsToSnapshot – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Gibt an, ob Tags aus der DB-Instance in Snapshots der DB-Instance kopiert werden.

- DBClusterIdentifier – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Wenn die DB-Instance ein Mitglied eines DB-Clusters ist, enthält sie den Namen des DB-Clusters, von dem die DB-Instance ein Mitglied ist.

- DBInstanceArn – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Amazon-Ressourcenname (ARN) für die DB-Instance.

- DBInstanceClass – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Enthält den Namen der Rechenkapazitäts- und Speicherkapazitätsklasse der DB-Instance.

- DBInstanceIdentifier – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Enthält eine vom Benutzer bereitgestellte Datenbank-ID. Diese ID ist der eindeutige Schlüssel zur Identifizierung einer DB-Instance.

- DbInstancePort – eine Ganzzahl vom Typ `integer` (32-Bit-Ganzzahl mit Vorzeichen).

  Gibt den Port an, den die DB-Instance abfragt. Wenn die DB-Instance Teil eines DB-Clusters ist, kann es sich dabei um einen anderen Port als den DB-Cluster-Port handeln.

- DBInstanceStatus – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den aktuellen Status dieser Datenbank an.

- DbiResourceId – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die für die Amazon-Region eindeutige, unveränderliche ID für die DB-Instance. Diese Kennung wird in CloudTrail Amazon-Protokolleinträgen immer dann gefunden, wenn auf den Amazon KMS-Schlüssel für die DB-Instance zugegriffen wird.

- DBName – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Datenbankname.

- DBParameterGroups – Ein Array mit DBParameterGroupStatus-Objekten.

  Stellt die Liste der DB-Parametergruppen bereit, die auf diese DB-Instance angewendet wurden.

- DBSecurityGroups – Ein Array mit DBSecurityGroupMembership-Objekten.

  Stellt eine Liste der DB-Sicherheitsgruppenelemente bereit, die nur `DBSecurityGroup.Name`- und `DBSecurityGroup.Status`-Unterelemente enthalten.

- DBSubnetGroup – Ein DBSubnetGruppe-Objekt.

Gibt Informationen zu der Subnetzgruppe an, die der DB-Instance zugeordnet ist, einschließlich dem Namen, der Beschreibung und der Subnetze in der Subnetzgruppe.

- DeletionProtection— a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Gibt an, ob der Löschschutz der DB-Instance aktiviert ist. Die Instance kann bei aktiviertem Löschschutz nicht gelöscht werden. Siehe Löschen einer DB-Instance.

- DomainMemberships – Ein Array mit DomainMembership-Objekten.

  Nicht unterstützt

- EnabledCloudwatchLogsExports – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Liste der Protokolltypen, für deren Export diese DB-Instance in Logs konfiguriert ist. CloudWatch

- Endpoint – Ein Endpunkt-Objekt.

  Gibt den Verbindungsendpunkt an.

- Engine – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Stellt den Namen der Datenbank-Engine bereit, die für diese DB-Instance verwendet werden soll.

- EngineVersion – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt die Version der Datenbank-Engine an.

- EnhancedMonitoringResourceArn – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Amazon-Ressourcenname (ARN) des Amazon CloudWatch Logs-Protokollstreams, der die Enhanced Monitoring-Metrikdaten für die DB-Instance empfängt.

- IAMDatabaseAuthenticationEnabled – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  „True“, wenn Amazon Identity and Access Management (IAM)-Authentifizierung aktiviert ist und andernfalls „false“.

- InstanceCreateTime— a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Offset von Mitternacht am 01.01.1970).

Stellt das Datum und die Uhrzeit bereit, wann die DB-Instance erstellt wurde.

- Iops— an IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Gibt den bereitgestellten IOPS (E/A-Operationen pro Sekunde)-Wert an.

- KmsKeyId – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Nicht unterstützt: Die Verschlüsselung für DB-Instances wird vom DB-Cluster verwaltet.

- LatestRestorableTime— a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Offset von Mitternacht am 01.01.1970).

  Gibt den letzten Zeitpunkt an, zu dem eine Datenbank mit point-in-time restore wiederhergestellt werden kann.

- LicenseModel – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Lizenzmodellinformationen für diese DB-Instance.

- MonitoringInterval— ein IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Das Intervall in Sekunden zwischen den Punkten, an denen erweiterte Überwachungsmetriken für die DB-Instance erfasst werden.

- MonitoringRoleArn – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der ARN für die IAM-Rolle, der es Neptune ermöglicht, Enhanced Monitoring-Metriken an Amazon Logs zu senden. CloudWatch

- MultiAZ – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Gibt an, ob die DB-Instance eine Multi-AZ-Bereitstellung ist.

- PendingModifiedValues – Ein PendingModifiedValues-Objekt.

  Gibt an, dass Änderungen an der DB-Instance ausstehen. Dieses Element ist nur enthalten, wenn Änderungen ausstehen. Spezifische Änderungen werden von Unterelementen identifiziert.

- PreferredBackupWindow – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den täglichen Zeitraum in koordinierter Weltzeit (UTC) an, in dem automatische Sicherungen erstellt werden, wenn automatische Sicherungen aktiviert sind, gemäß `BackupRetentionPeriod`.

- PreferredMaintenanceWindow – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den wöchentlichen Zeitraum, in dem Systemwartungen durchgeführt werden können, in UTC (Universal Coordinated Time) an.
- PromotionTier— ein IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Ein Wert, der die Reihenfolge angibt, in der eine Read Replica zur primären Instance hochgestuft wird, wenn die vorhandene primäre Instance ausfällt.
- PubliclyAccessible – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Dieses Flag sollte nicht mehr verwendet werden.
- ReadReplicaDBClusterIdentifiers – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Enthält eine oder mehrere IDs der DB-Cluster, die Read Replicas dieser DB-Instance sind.
- ReadReplicaDBInstanceIdentifiers – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Enthält eine oder mehrere IDs der Read Replicas, die dieser DB-Instance zugeordnet sind.
- ReadReplicaSourceDBInstanceIdentifier – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Enthält die ID der Quell-DB-Instance, wenn diese DB-Instance eine Read Replica ist.
- SecondaryAvailabilityZone – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt, wenn vorhanden, den Namen der sekundären Availability Zone für eine DB-Instance mit Multi-AZ-Unterstützung an.
- StatusInfos – Ein Array mit DBInstanceStatusInfo-Objekten.

  Der Status einer Read Replica. Wenn die Instance keine Read Replica ist, ist dies leer.
- StorageEncrypted – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Nicht unterstützt: Die Verschlüsselung für DB-Instances wird vom DB-Cluster verwaltet.
- StorageType – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Speichertyp an, der der DB-Instance zugeordnet ist.
- TdeCredentialArn – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Der ARN aus dem Schlüsselspeicher, mit dem die Instance für die TDE-Verschlüsselung verknüpft ist.

- Timezone – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Nicht unterstützt

- VpcSecurityGroups – Ein Array mit VpcSecurityGroupMembership-Objekten.

  Stellt eine Liste der VPC-Sicherheitsgruppenelemente bereit, zu denen die DB-Instance gehört.

Fehler

- DBInstanceAlreadyExistsFault
- Unzureichend DBInstance CapacityFault
- DBParameterGroupNotFoundFault
- DBSecurityGroupNotFoundFault
- InstanceQuotaExceededFault
- StorageQuotaExceededFault
- DBSubnetGroupNotFoundFault
- DBSubnetGroupDoesNotCoverEnoughAZs
- Ungültig DBCluster StateFault
- InvalidSubnet
- Ungültig VPCNetwork StateFault
- ProvisionedIopsNotAvailableInAZFault
- OptionGroupNotFoundFault
- DBClusterNotFoundFault
- StorageTypeNotSupportedFault
- AuthorizationNotFoundFault
- KMSKeyNotAccessibleFault
- DomainNotFoundFault

Löschen DBInstance (Aktion)

Der AWS CLI-Name für diese API lautet:`delete-db-instance`.

Die DBInstance Aktion Löschen löscht eine zuvor bereitgestellte DB-Instance. Beim Löschen einer DB-Instance werden alle automatischen Backups für diese Instance ebenfalls gelöscht und sind nicht wiederherstellbar. Manuelle DB-Snapshots der DB-Instance, die von `DeleteDBInstance` gelöscht werden soll, werden nicht gelöscht.

Wenn Sie einen abschließenden DB-Snapshot anfordern, lautet der Status der Amazon Neptune-DB-Instance `deleting`, bis der DB-Snapshot erstellt wird. Die API-Aktion `DescribeDBInstance` wird verwendet, um den Status dieser Operation zu überwachen. Die Aktion kann nicht abgebrochen oder zurückgesetzt werden, sobald gesendet.

Beachten Sie, dass wenn eine DB-Instance einen Fehler aufweist und über einen Status `failed`, `incompatible-restore` oder `incompatible-network` verfügt, Sie sie nur löschen können, wenn der `SkipFinalSnapshot`-Parameter auf `true` festgelegt ist.

Sie können eine DB-Instance nicht löschen, wenn sie die einzige Instance im DB-Cluster ist oder für sie der Löschschutz aktiviert ist.

Anforderung

- DBInstanceIdentifier (in der CLI: `--db-instance-identifier`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die DB-Instance-ID für die zu löschende DB-Instance. Bei diesem Parameter wird nicht zwischen Groß- und Kleinschreibung unterschieden.

  Einschränkungen:

  - Muss dem Namen einer vorhandenen DB-Instance entsprechen.
- FinalDBSnapshotIdentifier (in der CLI: `--final-db-snapshot-identifier`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der DBSnapshot Identifier der neu DBSnapshot erstellten Datei SkipFinalSnapshot ist auf gesetzt. `false`

> **Note**
>
> Die Angabe dieses Parameters und das Setzen des SkipFinalShapshot Parameters auf true führt zu einem Fehler.

Einschränkungen:

- Muss 1 bis 255 Buchstaben oder Zahlen enthalten.
- Muss mit einem Buchstaben beginnen
- Darf nicht mit einem Bindestrich enden oder zwei aufeinanderfolgende Bindestriche enthalten
- Kann beim Löschen einer Read Replica nicht angegeben werden.

- SkipFinalSnapshot (in der CLI: `--skip-final-snapshot`) – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Legt fest, ob vor dem Löschen der DB-Instance ein finaler DB-Snapshot erstellt wird. Wenn `true` angegeben, DBSnapshot wird kein erstellt. Wenn `false` angegeben ist, wird ein DB-Snapshot erstellt, bevor die DB-Instance gelöscht wird.

  Beachten Sie, dass eine DB-Instance, wenn sie sich in einem Fehlerstatus befindet und den Status 'failed', 'incompatible-restore' oder 'incompatible-network' hat, sie nur gelöscht werden kann, wenn der SkipFinalSnapshot Parameter auf „true“ gesetzt ist.

  Geben Sie `true` beim Löschen einer Read Replica an.

  > **Note**
  >
  > Falls ja, muss der Final DBSnapshot Identifier-Parameter angegeben werden. SkipFinalSnapshot `false`

  Standard: `false`

Antwort

Enthält die Details einer Amazon Neptune-DB-Instance.

Dieser Datentyp wird als Antwortelement in der Aktion the section called “Beschreiben DBInstances” verwendet.

- AutoMinorVersionUpgrade – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Gibt an, dass Patches von Unterversionen automatisch angewendet werden.
- AvailabilityZone – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Namen der Availability Zone an, in der sich die DB-Instance befindet.
- BackupRetentionPeriod – eine Ganzzahl vom Typ `integer` (32-Bit-Ganzzahl mit Vorzeichen).

  Gibt die Anzahl der Tage an, für die automatische DB-Snapshots aufbewahrt werden.
- CACertificateIdentifier – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die ID des Zertifizierungsstellenzertifikats für diese DB-Instance.
- CopyTagsToSnapshot – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Gibt an, ob Tags aus der DB-Instance in Snapshots der DB-Instance kopiert werden.
- DBClusterIdentifier – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Wenn die DB-Instance ein Mitglied eines DB-Clusters ist, enthält sie den Namen des DB-Clusters, von dem die DB-Instance ein Mitglied ist.
- DBInstanceArn – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Amazon-Ressourcenname (ARN) für die DB-Instance.
- DBInstanceClass – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Enthält den Namen der Rechenkapazitäts- und Speicherkapazitätsklasse der DB-Instance.
- DBInstanceIdentifier – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Enthält eine vom Benutzer bereitgestellte Datenbank-ID. Diese ID ist der eindeutige Schlüssel zur Identifizierung einer DB-Instance.
- DbInstancePort – eine Ganzzahl vom Typ `integer` (32-Bit-Ganzzahl mit Vorzeichen).

  Gibt den Port an, den die DB-Instance abfragt. Wenn die DB-Instance Teil eines DB-Clusters ist, kann es sich dabei um einen anderen Port als den DB-Cluster-Port handeln.
- DBInstanceStatus – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Gibt den aktuellen Status dieser Datenbank an.

- DbiResourceId – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die für die Amazon-Region eindeutige, unveränderliche ID für die DB-Instance. Diese Kennung wird in CloudTrail Amazon-Protokolleinträgen immer dann gefunden, wenn auf den Amazon KMS-Schlüssel für die DB-Instance zugegriffen wird.
- DBName – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Datenbankname.
- DBParameterGroups – Ein Array mit DBParameterGroupStatus-Objekten.

  Stellt die Liste der DB-Parametergruppen bereit, die auf diese DB-Instance angewendet wurden.
- DBSecurityGroups – Ein Array mit DBSecurityGroupMembership-Objekten.

  Stellt eine Liste der DB-Sicherheitsgruppenelemente bereit, die nur `DBSecurityGroup.Name`- und `DBSecurityGroup.Status`-Unterelemente enthalten.
- DBSubnetGroup – Ein DBSubnetGruppe-Objekt.

  Gibt Informationen zu der Subnetzgruppe an, die der DB-Instance zugeordnet ist, einschließlich dem Namen, der Beschreibung und der Subnetze in der Subnetzgruppe.
- DeletionProtection— a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Gibt an, ob der Löschschutz der DB-Instance aktiviert ist. Die Instance kann bei aktiviertem Löschschutz nicht gelöscht werden. Siehe Löschen einer DB-Instance.
- DomainMemberships – Ein Array mit DomainMembership-Objekten.

  Nicht unterstützt
- EnabledCloudwatchLogsExports – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Liste der Protokolltypen, für deren Export diese DB-Instance in Logs konfiguriert ist. CloudWatch
- Endpoint – Ein Endpunkt-Objekt.

  Gibt den Verbindungsendpunkt an.
- Engine – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Stellt den Namen der Datenbank-Engine bereit, die für diese DB-Instance verwendet werden soll.

- EngineVersion – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt die Version der Datenbank-Engine an.

- EnhancedMonitoringResourceArn – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Amazon-Ressourcenname (ARN) des Amazon CloudWatch Logs-Protokollstreams, der die Enhanced Monitoring-Metrikdaten für die DB-Instance empfängt.

- IAMDatabaseAuthenticationEnabled – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  „True“, wenn Amazon Identity and Access Management (IAM)-Authentifizierung aktiviert ist und andernfalls „false“.

- InstanceCreateTime— a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Offset von Mitternacht am 01.01.1970).

  Stellt das Datum und die Uhrzeit bereit, wann die DB-Instance erstellt wurde.

- Iops— an IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Gibt den bereitgestellten IOPS (E/A-Operationen pro Sekunde)-Wert an.

- KmsKeyId – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Nicht unterstützt: Die Verschlüsselung für DB-Instances wird vom DB-Cluster verwaltet.

- LatestRestorableTime— a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Offset von Mitternacht am 01.01.1970).

  Gibt den letzten Zeitpunkt an, zu dem eine Datenbank mit point-in-time restore wiederhergestellt werden kann.

- LicenseModel – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Lizenzmodellinformationen für diese DB-Instance.

- MonitoringInterval— ein IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Das Intervall in Sekunden zwischen den Punkten, an denen erweiterte Überwachungsmetriken für die DB-Instance erfasst werden.

- MonitoringRoleArn – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der ARN für die IAM-Rolle, der es Neptune ermöglicht, Enhanced Monitoring-Metriken an Amazon Logs zu senden. CloudWatch
- MultiAZ – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Gibt an, ob die DB-Instance eine Multi-AZ-Bereitstellung ist.
- PendingModifiedValues – Ein PendingModifiedValues-Objekt.

  Gibt an, dass Änderungen an der DB-Instance ausstehen. Dieses Element ist nur enthalten, wenn Änderungen ausstehen. Spezifische Änderungen werden von Unterelementen identifiziert.
- PreferredBackupWindow – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den täglichen Zeitraum in koordinierter Weltzeit (UTC) an, in dem automatische Sicherungen erstellt werden, wenn automatische Sicherungen aktiviert sind, gemäß `BackupRetentionPeriod`.
- PreferredMaintenanceWindow – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den wöchentlichen Zeitraum, in dem Systemwartungen durchgeführt werden können, in UTC (Universal Coordinated Time) an.
- PromotionTier— ein IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Ein Wert, der die Reihenfolge angibt, in der eine Read Replica zur primären Instance hochgestuft wird, wenn die vorhandene primäre Instance ausfällt.
- PubliclyAccessible – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Dieses Flag sollte nicht mehr verwendet werden.
- ReadReplicaDBClusterIdentifiers – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Enthält eine oder mehrere IDs der DB-Cluster, die Read Replicas dieser DB-Instance sind.
- ReadReplicaDBInstanceIdentifiers – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Enthält eine oder mehrere IDs der Read Replicas, die dieser DB-Instance zugeordnet sind.
- ReadReplicaSourceDBInstanceIdentifier – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Enthält die ID der Quell-DB-Instance, wenn diese DB-Instance eine Read Replica ist.

- SecondaryAvailabilityZone – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt, wenn vorhanden, den Namen der sekundären Availability Zone für eine DB-Instance mit Multi-AZ-Unterstützung an.

- StatusInfos – Ein Array mit DBInstanceStatusInfo-Objekten.

  Der Status einer Read Replica. Wenn die Instance keine Read Replica ist, ist dies leer.

- StorageEncrypted – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Nicht unterstützt: Die Verschlüsselung für DB-Instances wird vom DB-Cluster verwaltet.

- StorageType – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Speichertyp an, der der DB-Instance zugeordnet ist.

- TdeCredentialArn – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der ARN aus dem Schlüsselspeicher, mit dem die Instance für die TDE-Verschlüsselung verknüpft ist.

- Timezone – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Nicht unterstützt

- VpcSecurityGroups – Ein Array mit VpcSecurityGroupMembership-Objekten.

  Stellt eine Liste der VPC-Sicherheitsgruppenelemente bereit, zu denen die DB-Instance gehört.

Fehler

- DBInstanceNotFoundFault
- Ungültig DBInstance StateFault
- DBSnapshotAlreadyExistsFault
- SnapshotQuotaExceededFault
- Ungültig DBCluster StateFault

Ändern DBInstance (Aktion)

Der AWS CLI-Name für diese API lautet:`modify-db-instance`.

Ändert Einstellungen für eine DB-Instance. Sie können einen oder mehrere Datenbank-Konfigurationsparameter ändern, indem Sie diese Parameter und die neuen Werte in der Anforderung angeben. Für weitere Informationen zu den Änderungen, die Sie an Ihrer DB-Instance vornehmen können, rufen Sie the section called “DescribeValidDBInstanceÄnderungen” auf, bevor Sie the section called “Modifizieren DBInstance” aufrufen.

Anforderung

- AllowMajorVersionUpgrade (in der CLI: `--allow-major-version-upgrade`) – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Gibt an, dass Hauptversions-Upgrades zulässig sind. Das Ändern dieses Parameters führt nicht zu einem Nutzungsausfall und die Änderung wird asynchron zum nächstmöglichen Zeitpunkt übernommen.

- ApplyImmediately (in der CLI: `--apply-immediately`) – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Gibt an, ob die Änderungen in dieser Anforderung und sämtliche ausstehenden Änderungen asynchron zum nächstmöglichen Zeitpunkt übernommen werden, unabhängig von der `PreferredMaintenanceWindow`-Einstellung für die DB-Instance.

  Wenn dieser Parameter auf `false` festgelegt ist, werden Änderungen an der DB-Instance im nächsten Wartungsfenster übernommen. Einige Parameteränderungen können zu einem Nutzungsausfall führen und werden beim nächsten Aufruf von the section called “Starten Sie neu DBInstance” oder dem nächsten Ausfallneustart übernommen.

  Standard: `false`

- AutoMinorVersionUpgrade(in der CLI:`--auto-minor-version-upgrade`) — a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Gibt an, dass Unterversion-Upgrades innerhalb des Wartungsfensters automatisch auf die DB-Instance angewendet werden. Das Ändern dieses Parameters führt nicht zu einem Nutzungsausfall, mit Ausnahme des folgenden Falls, und die Änderung wird asynchron zum nächstmöglichen Zeitpunkt übernommen. Ein Nutzungsausfall tritt auf, wenn dieser Parameter während des Wartungsfensters auf `true` gesetzt ist und eine neuere Unterversion verfügbar ist und in Neptune das automatische Patchen für die Engine-Version aktiviert ist.

- BackupRetentionPeriod(in der CLI:`--backup-retention-period`) — ein IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

Nicht zutreffend. Der Aufbewahrungszeitraum von automatischen Backups wird vom DB-Cluster verwaltet. Weitere Informationen finden Sie unter the section called “Modifizieren DBCluster”.

Standard: Verwendet vorhandene Einstellung

- CACertificateIdentifier (in der CLI: `--ca-certificate-identifier`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt das Zertifikat an, das mit der Instance verknüpft werden muss.

- CloudwatchLogsExportConfiguration (in der CLI: `--cloudwatch-logs-export-configuration`) – Ein CloudwatchLogsExportConfiguration-Objekt.

  Die Konfigurationseinstellung für die Protokolltypen, die für den Export in CloudWatch Logs für eine bestimmte DB-Instance oder einen bestimmten DB-Cluster aktiviert werden sollen.

- CopyTagsToSnapshot(in der CLI:`--copy-tags-to-snapshot`) — a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  "True" zum Kopieren aller Tags aus der DB-Instance in die Snapshots der DB-Instance und andernfalls "false". Der Standardwert lautet „false“.

- DBInstanceClass (in der CLI: `--db-instance-class`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die neue Rechen- und Speicherkapazität der DB-Instance, beispielsweise `db.m4.large`. Nicht alle DB-Instance-Klassen stehen in allen Amazon-Regionen zur Verfügung.

  Wenn Sie die DB-Instance-Klasse ändern, kommt es während der Änderung zu einem Nutzungsausfall. Die Änderung wird während des nächsten Wartungsfensters angewendet, es sei denn, `ApplyImmediately` wird für diese Anforderung als `true` angegeben.

  Standard: Verwendet vorhandene Einstellung

- DBInstanceIdentifier (in der CLI: `--db-instance-identifier`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die DB-Instance-Kennung. Dieser Wert wird als Zeichenfolge in Kleinbuchstaben gespeichert.

  Einschränkungen:

  - Muss mit dem Bezeichner eines vorhandenen Bezeichners übereinstimmen. DBInstance

- DBParameterGroupName (in der CLI: `--db-parameter-group-name`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Name der DB-Parametergruppe, die auf die DB-Instance angewendet werden soll. Das Ändern dieser Einstellung führt nicht zu einem Nutzungsausfall. Der Name der Parametergruppe wird zwar direkt geändert, aber die tatsächlichen Parameteränderungen werden erst angewendet, wenn Sie die Instance ohne Failover neu starten. Die DB-Instance wird NICHT automatisch neu gestartet und die Parameteränderungen werden während des nächsten Wartungsfensters NICHT übernommen.

  Standard: Verwendet vorhandene Einstellung

  Einschränkungen: Die DB-Parametergruppe muss sich in derselben DB-Parametergruppenfamilie wie diese DB-Instance befinden.

- DBPortNumber(in der CLI:`--db-port-number`) — ein IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Die Portnummer, an der die Datenbank Verbindungen akzeptiert.

  Der Wert des `DBPortNumber`-Parameters darf keinem der Port-Werte entsprechen, die für Optionen in der Optionsgruppe für die DB-Instance angegeben wurden.

  Ihre Datenbank wird neu gestartet, wenn Sie den `DBPortNumber`-Wert ändern, unabhängig von dem Wert des `ApplyImmediately`-Parameters.

  Standard: `8182`

- DBSecurityGroups (in der CLI: `--db-security-groups`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Liste der DB-Sicherheitsgruppen, die auf dieser DB-Instance autorisiert werden sollen. Das Ändern dieser Einstellung führt nicht zu einem Nutzungsausfall und die Änderung wird asynchron zum nächstmöglichen Zeitpunkt übernommen.

  Einschränkungen:

  - Falls angegeben, muss es mit vorhandenen DBSecurity Gruppen übereinstimmen.

- DBSubnetGroupName (in der CLI: `--db-subnet-group-name`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die neue DB-Subnetzgruppe für die DB-Instance. Sie können diesen Parameter verwenden, um Ihre DB-Instance in eine andere VPC zu verschieben.

Das Ändern der Subnetzgruppe verursacht während der Änderung einen Nutzungsausfall. Die Änderung wird während des nächsten Wartungsfensters angewendet, es sei denn, Sie geben `true` für den `ApplyImmediately`-Parameter an.

Einschränkungen: Falls angegeben, muss es mit dem Namen einer vorhandenen DBSubnet Gruppe übereinstimmen.

Beispiel: `mySubnetGroup`

- DeletionProtection(in der CLI:`--deletion-protection`) — a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Ein Wert, der angibt, ob der Löschschutz für die DB-Instance aktiviert ist. Die Datenbank kann nicht gelöscht werden, wenn der Löschschutz aktiviert ist. Der Löschschutz ist standardmäßig deaktiviert. Siehe Löschen einer DB-Instance.
- Domain (in der CLI: `--domain`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Nicht unterstützt
- DomainIAMRoleName (in der CLI: `--domain-iam-role-name`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Nicht unterstützt
- EnableIAMDatabaseAuthentication(in der CLI:`--enable-iam-database-authentication`) — a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  „True“, um die Zuweisung von Amazon Identity and Access Management (IAM)-Konten zu Datenbankkonten zu aktivieren; andernfalls „false“.

  Sie können die IAM-Datenbankauthentifizierung für die folgenden Datenbank-Engines aktivieren

  Nicht zutreffend. Die Zuordnung von Amazon-IAM-Konten zu Datenbankkonten wird vom DB-Cluster verwaltet. Weitere Informationen finden Sie unter the section called “Modifizieren DBCluster”.

  Standard: `false`
- EngineVersion (in der CLI: `--engine-version`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Die Versionsnummer der Datenbank-Engine, auf die das Upgrade durchgeführt wird Derzeit hat das Festlegen dieses Parameters keine Auswirkungen. Verwenden Sie die the section called “ApplyPendingMaintenanceAction”-API, um Ihr Datenbank-Engine auf die neueste Version zu aktualisieren.

- Iops(in der CLI:`--iops`) — ein IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Der neue bereitgestellte IOPS (E/A-Operationen pro Sekunde)-Wert für die Instance.

  Das Ändern dieser Einstellung führt nicht zu einem Nutzungsausfall und die Änderung wird während des nächsten Wartungsfensters übernommen, es sei denn, der `ApplyImmediately`-Parameter ist für diese Anforderung auf `true` festgelegt.

  Standard: Verwendet vorhandene Einstellung

- MonitoringInterval(in der CLI:`--monitoring-interval`) — ein IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Das Intervall in Sekunden zwischen den Punkten, an denen erweiterte Überwachungsmetriken für die DB-Instance erfasst werden. Um die Erfassung von Metriken für erweiterte Überwachung zu deaktivieren, geben Sie 0 an. Der Standardwert ist 0.

  Wenn `MonitoringRoleArn` angegeben ist, müssen Sie auch `MonitoringInterval` auf einen anderen Wert als 0 festlegen.

  Zulässige Werte: `0, 1, 5, 10, 15, 30, 60`

- MonitoringRoleArn (in der CLI: `--monitoring-role-arn`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der ARN für die IAM-Rolle, der es Neptune ermöglicht, erweiterte Überwachungsmetriken an Amazon Logs zu senden. CloudWatch Beispiel, `arn:aws:iam:123456789012:role/emaccess`.

  Wenn `MonitoringInterval` auf einen anderen Wert als 0 festgelegt ist, müssen Sie einen `MonitoringRoleArn`-Wert bereitstellen.

- MultiAZ(in der CLI:`--multi-az`) — a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

Gibt an, ob die DB-Instance eine Multi-AZ-Bereitstellung ist. Das Ändern dieses Parameters führt nicht zu einem Nutzungsausfall und die Änderung wird während des nächsten Wartungsfensters übernommen, es sei denn, der `ApplyImmediately`-Parameter ist für diese Anforderung auf `true` festgelegt.

- NewDBInstanceIdentifier (in der CLI: `--new-db-instance-identifier`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die neue DB-Instance-Kennung für die DB-Instance, wenn eine DB-Instance umbenannt wird. Bei einer Änderung der DB-Instance-Kennung wird der Instance-Neustart entweder sofort (wenn `Apply Immediately` den Wert "true" aufweist) oder während des nächsten Wartungsfensters (wenn `Apply Immediately` auf den Wert "false" gesetzt ist) durchgeführt. Dieser Wert wird als Zeichenfolge in Kleinbuchstaben gespeichert.

  Einschränkungen:

  - Muss zwischen 1 und 63 Buchstaben, Ziffern oder Bindestriche enthalten.
  - Das erste Zeichen muss ein Buchstabe sein.
  - Darf nicht mit einem Bindestrich enden oder zwei aufeinanderfolgende Bindestriche enthalten.

  Beispiel: `mydbinstance`

- PreferredBackupWindow (in der CLI: `--preferred-backup-window`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der tägliche Zeitraum, in dem automatische Backups erstellt werden, wenn diese aktiviert sind.

  Nicht zutreffend. Der tägliche Zeitraum für das Erstellen von automatischen Backups wird vom DB-Cluster verwaltet. Weitere Informationen finden Sie unter the section called “Modifizieren DBCluster”.

  Einschränkungen:

  - Muss im Format hh24:mi-hh24:mi vorliegen.
  - Muss in Coordinated Universal Time (UTC) vorliegen.
  - Darf nicht mit dem bevorzugten Wartungsfenster in Konflikt treten.
  - Muss mindestens 30 Minuten betragen.

- PreferredMaintenanceWindow (in der CLI: `--preferred-maintenance-window`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Der wöchentliche Zeitraum (in UTC), in dem Systemwartungen durchgeführt werden können, die möglicherweise zu einem Nutzungsausfall führen. Das Ändern dieses Parameters führt nicht zu einem Nutzungsausfall, mit Ausnahme der folgenden Situation, und die Änderung wird asynchron zum nächstmöglichen Zeitpunkt übernommen. Wenn ausstehende Aktionen vorhanden sind, die einen Neustart verursachen, und das Wartungsfenster geändert wird, um die aktuelle Zeit einzubeziehen, dann führt das Ändern dieses Parameters zu einem Neustart der DB-Instance. Falls Sie dieses Fenster auf die aktuelle Zeit umstellen, müssen mindestens 30 Minuten zwischen der aktuellen Zeit und der Endzeit des Fensters liegen, um sicherzustellen, dass ausstehende Änderungen übernommen werden.

Standard: Verwendet vorhandene Einstellung

Format: ddd:hh24:mi-ddd:hh24:mi

Gültige Tage: Mo | Di | Mi | Do | Fr | Sa | So

Einschränkungen: Muss mindestens 30 Minuten betragen.

- PromotionTier(in der CLI:`--promotion-tier`) — ein IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Ein Wert, der die Reihenfolge angibt, in der eine Read Replica zur primären Instance hochgestuft wird, wenn die vorhandene primäre Instance ausfällt.

  Standard: 1

  Gültige Werte: 0 bis 15

- PubliclyAccessible(in der CLI:`--publicly-accessible`) — a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Dieses Flag sollte nicht mehr verwendet werden.

- StorageType (in der CLI: `--storage-type`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Nicht zutreffend. In Neptune wird der Speichertyp auf DB-Cluster-Ebene verwaltet.

- TdeCredentialArn (in der CLI: `--tde-credential-arn`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Der ARN aus dem Schlüsselspeicher, mit dem die Instance für die TDE-Verschlüsselung verknüpft werden soll.

- TdeCredentialPassword(in der CLI:`--tde-credential-password`) — a SensitiveString, vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

  Das Passwort für den angegebenen ARN aus dem Schlüsselspeicher, um auf das Gerät zuzugreifen.

- VpcSecurityGroupIds (in der CLI: `--vpc-security-group-ids`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Liste von EC2 VPC-Sicherheitsgruppen, die auf dieser DB-Instance autorisiert werden sollen. Diese Änderung wird asynchron zum nächstmöglichen Zeitpunkt übernommen.

  Nicht zutreffend. Die zugehörige Liste der EC2 VPC-Sicherheitsgruppen wird vom DB-Cluster verwaltet. Weitere Informationen finden Sie unter the section called “Modifizieren DBCluster”.

  Einschränkungen:

  - Falls angegeben, muss sie mit den vorhandenen VpcSecurityGroupIds übereinstimmen.

Antwort

Enthält die Details einer Amazon Neptune-DB-Instance.

Dieser Datentyp wird als Antwortelement in der Aktion the section called “Beschreiben DBInstances” verwendet.

- AutoMinorVersionUpgrade – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Gibt an, dass Patches von Unterversionen automatisch angewendet werden.

- AvailabilityZone – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Namen der Availability Zone an, in der sich die DB-Instance befindet.

- BackupRetentionPeriod – eine Ganzzahl vom Typ `integer` (32-Bit-Ganzzahl mit Vorzeichen).

  Gibt die Anzahl der Tage an, für die automatische DB-Snapshots aufbewahrt werden.

- CACertificateIdentifier – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Die ID des Zertifizierungsstellenzertifikats für diese DB-Instance.

- CopyTagsToSnapshot – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Gibt an, ob Tags aus der DB-Instance in Snapshots der DB-Instance kopiert werden.
- DBClusterIdentifier – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Wenn die DB-Instance ein Mitglied eines DB-Clusters ist, enthält sie den Namen des DB-Clusters, von dem die DB-Instance ein Mitglied ist.
- DBInstanceArn – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Amazon-Ressourcenname (ARN) für die DB-Instance.
- DBInstanceClass – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Enthält den Namen der Rechenkapazitäts- und Speicherkapazitätsklasse der DB-Instance.
- DBInstanceIdentifier – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Enthält eine vom Benutzer bereitgestellte Datenbank-ID. Diese ID ist der eindeutige Schlüssel zur Identifizierung einer DB-Instance.
- DbInstancePort – eine Ganzzahl vom Typ `integer` (32-Bit-Ganzzahl mit Vorzeichen).

  Gibt den Port an, den die DB-Instance abfragt. Wenn die DB-Instance Teil eines DB-Clusters ist, kann es sich dabei um einen anderen Port als den DB-Cluster-Port handeln.
- DBInstanceStatus – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den aktuellen Status dieser Datenbank an.
- DbiResourceId – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die für die Amazon-Region eindeutige, unveränderliche ID für die DB-Instance. Diese Kennung wird in CloudTrail Amazon-Protokolleinträgen immer dann gefunden, wenn auf den Amazon KMS-Schlüssel für die DB-Instance zugegriffen wird.
- DBName – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Datenbankname.
- DBParameterGroups – Ein Array mit DBParameterGroupStatus-Objekten.

  Stellt die Liste der DB-Parametergruppen bereit, die auf diese DB-Instance angewendet wurden.
- DBSecurityGroups – Ein Array mit DBSecurityGroupMembership-Objekten.

Stellt eine Liste der DB-Sicherheitsgruppenelemente bereit, die nur `DBSecurityGroup.Name`- und `DBSecurityGroup.Status`-Unterelemente enthalten.

- DBSubnetGroup – Ein DBSubnetGruppe-Objekt.

  Gibt Informationen zu der Subnetzgruppe an, die der DB-Instance zugeordnet ist, einschließlich dem Namen, der Beschreibung und der Subnetze in der Subnetzgruppe.

- DeletionProtection— a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Gibt an, ob der Löschschutz der DB-Instance aktiviert ist. Die Instance kann bei aktiviertem Löschschutz nicht gelöscht werden. Siehe Löschen einer DB-Instance.

- DomainMemberships – Ein Array mit DomainMembership-Objekten.

  Nicht unterstützt

- EnabledCloudwatchLogsExports – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Liste der Protokolltypen, für deren Export diese DB-Instance in Logs konfiguriert ist. CloudWatch

- Endpoint – Ein Endpunkt-Objekt.

  Gibt den Verbindungsendpunkt an.

- Engine – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Stellt den Namen der Datenbank-Engine bereit, die für diese DB-Instance verwendet werden soll.

- EngineVersion – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt die Version der Datenbank-Engine an.

- EnhancedMonitoringResourceArn – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Amazon-Ressourcenname (ARN) des Amazon CloudWatch Logs-Protokollstreams, der die Enhanced Monitoring-Metrikdaten für die DB-Instance empfängt.

- IAMDatabaseAuthenticationEnabled – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

„True“, wenn Amazon Identity and Access Management (IAM)-Authentifizierung aktiviert ist und andernfalls „false“.

- InstanceCreateTime— a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Offset von Mitternacht am 01.01.1970).

  Stellt das Datum und die Uhrzeit bereit, wann die DB-Instance erstellt wurde.

- Iops— an IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Gibt den bereitgestellten IOPS (E/A-Operationen pro Sekunde)-Wert an.

- KmsKeyId – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Nicht unterstützt: Die Verschlüsselung für DB-Instances wird vom DB-Cluster verwaltet.

- LatestRestorableTime— a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Offset von Mitternacht am 01.01.1970).

  Gibt den letzten Zeitpunkt an, zu dem eine Datenbank mit point-in-time restore wiederhergestellt werden kann.

- LicenseModel – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Lizenzmodellinformationen für diese DB-Instance.

- MonitoringInterval— ein IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Das Intervall in Sekunden zwischen den Punkten, an denen erweiterte Überwachungsmetriken für die DB-Instance erfasst werden.

- MonitoringRoleArn – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der ARN für die IAM-Rolle, der es Neptune ermöglicht, Enhanced Monitoring-Metriken an Amazon Logs zu senden. CloudWatch

- MultiAZ – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Gibt an, ob die DB-Instance eine Multi-AZ-Bereitstellung ist.

- PendingModifiedValues – Ein PendingModifiedValues-Objekt.

  Gibt an, dass Änderungen an der DB-Instance ausstehen. Dieses Element ist nur enthalten, wenn Änderungen ausstehen. Spezifische Änderungen werden von Unterelementen identifiziert.

- PreferredBackupWindow – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Gibt den täglichen Zeitraum in koordinierter Weltzeit (UTC) an, in dem automatische Sicherungen erstellt werden, wenn automatische Sicherungen aktiviert sind, gemäß `BackupRetentionPeriod`.

- PreferredMaintenanceWindow – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den wöchentlichen Zeitraum, in dem Systemwartungen durchgeführt werden können, in UTC (Universal Coordinated Time) an.
- PromotionTier— ein IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Ein Wert, der die Reihenfolge angibt, in der eine Read Replica zur primären Instance hochgestuft wird, wenn die vorhandene primäre Instance ausfällt.
- PubliclyAccessible – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Dieses Flag sollte nicht mehr verwendet werden.
- ReadReplicaDBClusterIdentifiers – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Enthält eine oder mehrere IDs der DB-Cluster, die Read Replicas dieser DB-Instance sind.
- ReadReplicaDBInstanceIdentifiers – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Enthält eine oder mehrere IDs der Read Replicas, die dieser DB-Instance zugeordnet sind.
- ReadReplicaSourceDBInstanceIdentifier – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Enthält die ID der Quell-DB-Instance, wenn diese DB-Instance eine Read Replica ist.
- SecondaryAvailabilityZone – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt, wenn vorhanden, den Namen der sekundären Availability Zone für eine DB-Instance mit Multi-AZ-Unterstützung an.
- StatusInfos – Ein Array mit DBInstanceStatusInfo-Objekten.

  Der Status einer Read Replica. Wenn die Instance keine Read Replica ist, ist dies leer.
- StorageEncrypted – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Nicht unterstützt: Die Verschlüsselung für DB-Instances wird vom DB-Cluster verwaltet.

- StorageType – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Speichertyp an, der der DB-Instance zugeordnet ist.

- TdeCredentialArn – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der ARN aus dem Schlüsselspeicher, mit dem die Instance für die TDE-Verschlüsselung verknüpft ist.

- Timezone – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Nicht unterstützt

- VpcSecurityGroups – Ein Array mit VpcSecurityGroupMembership-Objekten.

  Stellt eine Liste der VPC-Sicherheitsgruppenelemente bereit, zu denen die DB-Instance gehört.

Fehler

- Ungültig DBInstance StateFault
- Ungültig DBSecurity GroupStateFault
- DBInstanceAlreadyExistsFault
- DBInstanceNotFoundFault
- DBSecurityGroupNotFoundFault
- DBParameterGroupNotFoundFault
- Unzureichend DBInstance CapacityFault
- StorageQuotaExceededFault
- Ungültig VPCNetwork StateFault
- ProvisionedIopsNotAvailableInAZFault
- OptionGroupNotFoundFault
- DBUpgradeDependencyFailureFault
- StorageTypeNotSupportedFault
- AuthorizationNotFoundFault
- CertificateNotFoundFault
- DomainNotFoundFault

Neustart DBInstance (Aktion)

Der AWS CLI-Name für diese API lautet:`reboot-db-instance`.

Möglicherweise müssen Sie Ihre DB-Instance neu starten, in der Regel aus Wartungsgründen. Wenn Sie z. B. bestimmte Modifikationen vornehmen oder wenn Sie die der DB-Instance zugeordnete DB-Parametergruppe ändern, müssen Sie die Instance neu starten, damit die Änderungen wirksam werden.

Durch das Neustarten einer DB-Instance, wird der Datenbank-Engine-Dienst neugestartet. Das Neustarten einer DB-Instance bewirkt einen vorübergehenden Nutzungsausfall, wobei der Status für diese Instance währenddessen auf Wird neu gestartet … gesetzt wird.

Anforderung

- DBInstanceIdentifier (in der CLI: `--db-instance-identifier`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die DB-Instance-Kennung. Dieser Parameter wird als Zeichenfolge in Kleinbuchstaben gespeichert.

  Einschränkungen:
  - Muss mit dem Bezeichner einer vorhandenen Datei übereinstimmen DBInstance.
- ForceFailover(in der CLI:`--force-failover`) — a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Wenn `true`, erfolgt der Neustart über ein Multi-AZ-Failover.

  Beschränkung: Es ist nicht möglich, `true` anzugeben, wenn die Instance nicht für Multi-AZ konfiguriert ist.

Antwort

Enthält die Details einer Amazon Neptune-DB-Instance.

Dieser Datentyp wird als Antwortelement in der Aktion the section called “Beschreiben DBInstances” verwendet.

- AutoMinorVersionUpgrade – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

Gibt an, dass Patches von Unterversionen automatisch angewendet werden.

- AvailabilityZone – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Namen der Availability Zone an, in der sich die DB-Instance befindet.

- BackupRetentionPeriod – eine Ganzzahl vom Typ `integer` (32-Bit-Ganzzahl mit Vorzeichen).

  Gibt die Anzahl der Tage an, für die automatische DB-Snapshots aufbewahrt werden.

- CACertificateIdentifier – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die ID des Zertifizierungsstellenzertifikats für diese DB-Instance.

- CopyTagsToSnapshot – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Gibt an, ob Tags aus der DB-Instance in Snapshots der DB-Instance kopiert werden.

- DBClusterIdentifier – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Wenn die DB-Instance ein Mitglied eines DB-Clusters ist, enthält sie den Namen des DB-Clusters, von dem die DB-Instance ein Mitglied ist.

- DBInstanceArn – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Amazon-Ressourcenname (ARN) für die DB-Instance.

- DBInstanceClass – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Enthält den Namen der Rechenkapazitäts- und Speicherkapazitätsklasse der DB-Instance.

- DBInstanceIdentifier – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Enthält eine vom Benutzer bereitgestellte Datenbank-ID. Diese ID ist der eindeutige Schlüssel zur Identifizierung einer DB-Instance.

- DbInstancePort – eine Ganzzahl vom Typ `integer` (32-Bit-Ganzzahl mit Vorzeichen).

  Gibt den Port an, den die DB-Instance abfragt. Wenn die DB-Instance Teil eines DB-Clusters ist, kann es sich dabei um einen anderen Port als den DB-Cluster-Port handeln.

- DBInstanceStatus – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den aktuellen Status dieser Datenbank an.

- DbiResourceId – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Die für die Amazon-Region eindeutige, unveränderliche ID für die DB-Instance. Diese Kennung wird in CloudTrail Amazon-Protokolleinträgen immer dann gefunden, wenn auf den Amazon KMS-Schlüssel für die DB-Instance zugegriffen wird.

- DBName – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Datenbankname.

- DBParameterGroups – Ein Array mit DBParameterGroupStatus-Objekten.

  Stellt die Liste der DB-Parametergruppen bereit, die auf diese DB-Instance angewendet wurden.

- DBSecurityGroups – Ein Array mit DBSecurityGroupMembership-Objekten.

  Stellt eine Liste der DB-Sicherheitsgruppenelemente bereit, die nur `DBSecurityGroup.Name`- und `DBSecurityGroup.Status`-Unterelemente enthalten.

- DBSubnetGroup – Ein DBSubnetGruppe-Objekt.

  Gibt Informationen zu der Subnetzgruppe an, die der DB-Instance zugeordnet ist, einschließlich dem Namen, der Beschreibung und der Subnetze in der Subnetzgruppe.

- DeletionProtection— a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Gibt an, ob der Löschschutz der DB-Instance aktiviert ist. Die Instance kann bei aktiviertem Löschschutz nicht gelöscht werden. Siehe Löschen einer DB-Instance.

- DomainMemberships – Ein Array mit DomainMembership-Objekten.

  Nicht unterstützt

- EnabledCloudwatchLogsExports – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Liste der Protokolltypen, für deren Export diese DB-Instance in Logs konfiguriert ist. CloudWatch

- Endpoint – Ein Endpunkt-Objekt.

  Gibt den Verbindungsendpunkt an.

- Engine – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Stellt den Namen der Datenbank-Engine bereit, die für diese DB-Instance verwendet werden soll.

- EngineVersion – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Gibt die Version der Datenbank-Engine an.

- EnhancedMonitoringResourceArn – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Amazon-Ressourcenname (ARN) des Amazon CloudWatch Logs-Protokollstreams, der die Enhanced Monitoring-Metrikdaten für die DB-Instance empfängt.

- IAMDatabaseAuthenticationEnabled – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  „True“, wenn Amazon Identity and Access Management (IAM)-Authentifizierung aktiviert ist und andernfalls „false“.

- InstanceCreateTime— a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Offset von Mitternacht am 01.01.1970).

  Stellt das Datum und die Uhrzeit bereit, wann die DB-Instance erstellt wurde.

- Iops— an IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Gibt den bereitgestellten IOPS (E/A-Operationen pro Sekunde)-Wert an.

- KmsKeyId – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Nicht unterstützt: Die Verschlüsselung für DB-Instances wird vom DB-Cluster verwaltet.

- LatestRestorableTime— a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Offset von Mitternacht am 01.01.1970).

  Gibt den letzten Zeitpunkt an, zu dem eine Datenbank mit point-in-time restore wiederhergestellt werden kann.

- LicenseModel – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Lizenzmodellinformationen für diese DB-Instance.

- MonitoringInterval— ein IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Das Intervall in Sekunden zwischen den Punkten, an denen erweiterte Überwachungsmetriken für die DB-Instance erfasst werden.

- MonitoringRoleArn – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Der ARN für die IAM-Rolle, der es Neptune ermöglicht, Enhanced Monitoring-Metriken an Amazon Logs zu senden. CloudWatch

- MultiAZ – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Gibt an, ob die DB-Instance eine Multi-AZ-Bereitstellung ist.

- PendingModifiedValues – Ein PendingModifiedValues-Objekt.

  Gibt an, dass Änderungen an der DB-Instance ausstehen. Dieses Element ist nur enthalten, wenn Änderungen ausstehen. Spezifische Änderungen werden von Unterelementen identifiziert.

- PreferredBackupWindow – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den täglichen Zeitraum in koordinierter Weltzeit (UTC) an, in dem automatische Sicherungen erstellt werden, wenn automatische Sicherungen aktiviert sind, gemäß `BackupRetentionPeriod`.

- PreferredMaintenanceWindow – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den wöchentlichen Zeitraum, in dem Systemwartungen durchgeführt werden können, in UTC (Universal Coordinated Time) an.

- PromotionTier— ein IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Ein Wert, der die Reihenfolge angibt, in der eine Read Replica zur primären Instance hochgestuft wird, wenn die vorhandene primäre Instance ausfällt.

- PubliclyAccessible – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Dieses Flag sollte nicht mehr verwendet werden.

- ReadReplicaDBClusterIdentifiers – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Enthält eine oder mehrere IDs der DB-Cluster, die Read Replicas dieser DB-Instance sind.

- ReadReplicaDBInstanceIdentifiers – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Enthält eine oder mehrere IDs der Read Replicas, die dieser DB-Instance zugeordnet sind.

- ReadReplicaSourceDBInstanceIdentifier – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Enthält die ID der Quell-DB-Instance, wenn diese DB-Instance eine Read Replica ist.

- SecondaryAvailabilityZone – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt, wenn vorhanden, den Namen der sekundären Availability Zone für eine DB-Instance mit Multi-AZ-Unterstützung an.

- StatusInfos – Ein Array mit DBInstanceStatusInfo-Objekten.

  Der Status einer Read Replica. Wenn die Instance keine Read Replica ist, ist dies leer.

- StorageEncrypted – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Nicht unterstützt: Die Verschlüsselung für DB-Instances wird vom DB-Cluster verwaltet.

- StorageType – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Speichertyp an, der der DB-Instance zugeordnet ist.

- TdeCredentialArn – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der ARN aus dem Schlüsselspeicher, mit dem die Instance für die TDE-Verschlüsselung verknüpft ist.

- Timezone – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Nicht unterstützt

- VpcSecurityGroups – Ein Array mit VpcSecurityGroupMembership-Objekten.

  Stellt eine Liste der VPC-Sicherheitsgruppenelemente bereit, zu denen die DB-Instance gehört.

Fehler

- Ungültig DBInstance StateFault
- DBInstanceNotFoundFault

Beschreiben Sie DBInstances (Aktion)

Der AWS CLI-Name für diese API lautet:`describe-db-instances`.

Gibt Informationen zu bereitgestellten Instances zurück und unterstützt die Paginierung.

> **Note**
>
> Über diese Operation können auch Informationen für Amazon-RDS-Instances und Amazon-DocDB-Instances zurückgeben werden.

Anforderung

- DBInstanceIdentifier (in der CLI: `--db-instance-identifier`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die vom Benutzer angegebene Instance-ID. Wenn dieser Parameter angegeben ist, werden die Informationen nur von der spezifischen DB-Instance zurückgegeben. Bei diesem Parameter wird nicht zwischen Groß- und Kleinschreibung unterschieden.

  Einschränkungen:

  - Falls angegeben, muss er mit dem Bezeichner einer vorhandenen Datei übereinstimmen DBInstance.
- Filters (in der CLI: `--filters`) – Ein Array von Filter Objekten.

  Ein Filter, der eine oder mehrere zu beschreibende DB-Instances angibt.

  Unterstützte Filter:

  - `db-cluster-id`- Akzeptiert DB-Cluster-Identifikatoren und DB-Cluster-Amazon-Ressourcennamen (ARNs). Die Ergebnisliste wird nur Informationen über die DB-Instances enthalten, die den durch diese ARNs identifizierten DB-Clustern zugeordnet sind.
  - `engine` – Akzeptiert einen Engine-Namen (z. B. `neptune`) und beschränkt die Ergebnisliste auf DB-Instances, die von dieser Engine erstellt wurden.

  Um diese API beispielsweise über die Amazon CLI aufzurufen und so zu filtern, dass nur Neptune-DB-Instances zurückgegeben werden, können Sie den folgenden Befehl verwenden:

  Example

  ```
  aws neptune describe-db-instances \
              --filters  Name=engine,Values=neptune
  ```
- Marker (in der CLI: `--marker`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Ein optionales Paginierungstoken, das von einer vorherigen `DescribeDBInstances`-Anforderung bereitgestellt wird. Wenn Sie diesen Parameter angeben, enthält die Antwort nur die Datensätze zwischen der Markierung und dem durch `MaxRecords` angegebenen Wert.

- MaxRecords(in der CLI:`--max-records`) — ein IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Die maximale Anzahl der in der Antwort zurückgegebenen Datensätze. Wenn mehrere Datensätze vorhanden sind, als der Wert `MaxRecords` angibt, ist ein Paginierungstoken mit dem Namen einer Markierung in der Antwort enthalten, sodass die verbleibenden Ergebnisse abgerufen werden können.

  Standard: 100

  Einschränkungen: Mindestwert 20, Höchstwert 100.

Antwort

- DBInstances – Ein Array mit DBInstance-Objekten.

  Eine Liste der the section called “DBInstance”-Instances.

- Marker – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Ein optionales Paginierungstoken, das von einer vorherigen Anforderung bereitgestellt wird. Wenn Sie diesen Parameter angeben, enthält die Antwort nur die Datensätze zwischen der Markierung und dem durch `MaxRecords` angegebenen Wert.

Fehler

- DBInstanceNotFoundFault

DescribeOrderableDBInstanceOptionen (Aktion)

Der AWS CLI-Name für diese API lautet:`describe-orderable-db-instance-options`.

Gibt eine Liste der bestellbaren DB-Instance-Optionen für die angegebene Engine zurück.

Anforderung

- DBInstanceClass (in der CLI: `--db-instance-class`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Filterwert der DB-Instance-Klasse. Geben Sie diesen Parameter an, um nur die verfügbaren Angebote anzuzeigen, die mit der angegebene DB-Instance-Klasse übereinstimmen.

- Engine (in der CLI: `--engine`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Name der Engine, für die DB-Instance-Optionen abgerufen werden sollen.

- EngineVersion (in der CLI: `--engine-version`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Filterwert der Engine-Version. Geben Sie diesen Parameter an, um nur die verfügbaren Angebote anzuzeigen, die mit der angegebene Engine-Version übereinstimmen.

- Filters (in der CLI: `--filters`) – Ein Array von Filter Objekten.

  Dieser Parameter wird derzeit nicht unterstützt.

- LicenseModel (in der CLI: `--license-model`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Filterwert des Lizenzmodells. Geben Sie diesen Parameter an, um nur die verfügbaren Angebote anzuzeigen, die mit dem angegebene Lizenzmodell übereinstimmen.

- Marker (in der CLI: `--marker`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Ein optionales Paginierungstoken, das in einer früheren DescribeOrderable DBInstance Options-Anfrage bereitgestellt wurde. Wenn Sie diesen Parameter angeben, enthält die Antwort nur die Datensätze zwischen der Markierung und dem durch `MaxRecords` angegebenen Wert.

- MaxRecords(in der CLI:`--max-records`) — ein IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Die maximale Anzahl der in der Antwort zurückgegebenen Datensätze. Wenn mehrere Datensätze vorhanden sind, als der Wert `MaxRecords` angibt, ist ein Paginierungstoken mit dem Namen einer Markierung in der Antwort enthalten, sodass die verbleibenden Ergebnisse abgerufen werden können.

  Standard: 100

Einschränkungen: Mindestwert 20, Höchstwert 100.

- Vpc(in der CLI:`--vpc`) — a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Der Filterwert der VPC. Geben Sie diesen Parameter an, um nur die verfügbaren VPC- oder Nicht-VPC-Angebote anzuzeigen.

Antwort

- Marker – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Ein optionales Paginierungstoken, das in einer früheren Anfrage für DBInstance bestellbare Optionen bereitgestellt wurde. Wenn Sie diesen Parameter angeben, enthält die Antwort nur die Datensätze zwischen der Markierung und dem durch `MaxRecords` angegebenen Wert.
- OrderableDBInstanceOptions – Ein Array mit Bestellbare Option DBInstance-Objekten.

  Eine the section called “Bestellbare Option DBInstance”-Struktur mit Informationen zu den bestellbaren Optionen für die DB-Instance.

DescribeValidDBInstanceÄnderungen (Aktion)

Der AWS CLI-Name für diese API lautet:`describe-valid-db-instance-modifications`.

Sie können the section called “DescribeValidDBInstanceÄnderungen” für weitere Informationen zu den Änderungen aufrufen, die Sie an Ihrer DB-Instance vornehmen können. Sie können diese Informationen verwenden, wenn Sie the section called “Modifizieren DBInstance” aufrufen.

Anforderung

- DBInstanceIdentifier (in der CLI: `--db-instance-identifier`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Kunden-ID oder der ARN Ihrer DB-Instance.

Antwort

Informationen zu gültigen Änderungen, die Sie an Ihrer DB-Instance vornehmen können. Enthält das Ergebnis eines erfolgreichen Aufrufs der the section called “DescribeValidDBInstanceÄnderungen”-

Aktion. Sie können diese Informationen verwenden, wenn Sie the section called “Modifizieren DBInstance” aufrufen.

- Storage – Ein Array mit ValidStorageOptions-Objekten.

  Gültige Speicheroptionen für Ihre DB-Instance.

Fehler

- DBInstanceNotFoundFault
- Ungültig DBInstance StateFault

Strukturen:

DBInstance (Struktur)

Enthält die Details einer Amazon Neptune-DB-Instance.

Dieser Datentyp wird als Antwortelement in der Aktion the section called “Beschreiben DBInstances” verwendet.

Felder

- AutoMinorVersionUpgrade – Dies ist ein boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Gibt an, dass Patches von Unterversionen automatisch angewendet werden.
- AvailabilityZone – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Namen der Availability Zone an, in der sich die DB-Instance befindet.
- BackupRetentionPeriod – Dies ist eine Ganzzahl vom Typ `integer` (32-Bit-Ganzzahl mit Vorzeichen).

  Gibt die Anzahl der Tage an, für die automatische DB-Snapshots aufbewahrt werden.
- CACertificateIdentifier – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die ID des Zertifizierungsstellenzertifikats für diese DB-Instance.

- CopyTagsToSnapshot – Dies ist ein boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Gibt an, ob Tags aus der DB-Instance in Snapshots der DB-Instance kopiert werden.

- DBClusterIdentifier – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Wenn die DB-Instance ein Mitglied eines DB-Clusters ist, enthält sie den Namen des DB-Clusters, von dem die DB-Instance ein Mitglied ist.

- DBInstanceArn – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Amazon-Ressourcenname (ARN) für die DB-Instance.

- DBInstanceClass – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Enthält den Namen der Rechenkapazitäts- und Speicherkapazitätsklasse der DB-Instance.

- DBInstanceIdentifier – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Enthält eine vom Benutzer bereitgestellte Datenbank-ID. Diese ID ist der eindeutige Schlüssel zur Identifizierung einer DB-Instance.

- DbInstancePort – Dies ist eine Ganzzahl vom Typ `integer` (32-Bit-Ganzzahl mit Vorzeichen).

  Gibt den Port an, den die DB-Instance abfragt. Wenn die DB-Instance Teil eines DB-Clusters ist, kann es sich dabei um einen anderen Port als den DB-Cluster-Port handeln.

- DBInstanceStatus – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den aktuellen Status dieser Datenbank an.

- DbiResourceId – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die für die Amazon-Region eindeutige, unveränderliche ID für die DB-Instance. Diese Kennung wird in CloudTrail Amazon-Protokolleinträgen immer dann gefunden, wenn auf den Amazon KMS-Schlüssel für die DB-Instance zugegriffen wird.

- DBName – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Datenbankname.

- DBParameterGroups – Dies ist ein Array von DBParameterGroupStatus-Objekten.

  Stellt die Liste der DB-Parametergruppen bereit, die auf diese DB-Instance angewendet wurden.

- DBSecurityGroups – Dies ist ein Array von DBSecurityGroupMembership-Objekten.

Stellt eine Liste der DB-Sicherheitsgruppenelemente bereit, die nur `DBSecurityGroup.Name`- und `DBSecurityGroup.Status`-Unterelemente enthalten.

- DBSubnetGroup – Dies ist ein DBSubnetGruppe-Objekt.

  Gibt Informationen zu der Subnetzgruppe an, die der DB-Instance zugeordnet ist, einschließlich dem Namen, der Beschreibung und der Subnetze in der Subnetzgruppe.

- DeletionProtection— Dies ist ein BooleanOptional Wert vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Gibt an, ob der Löschschutz der DB-Instance aktiviert ist. Die Instance kann bei aktiviertem Löschschutz nicht gelöscht werden. Siehe Löschen einer DB-Instance.

- DomainMemberships – Dies ist ein Array von DomainMembership-Objekten.

  Nicht unterstützt

- EnabledCloudwatchLogsExports – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Liste der Protokolltypen, für deren Export diese DB-Instance in Logs konfiguriert ist. CloudWatch

- Endpoint – Dies ist ein Endpunkt-Objekt.

  Gibt den Verbindungsendpunkt an.

- Engine – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Stellt den Namen der Datenbank-Engine bereit, die für diese DB-Instance verwendet werden soll.

- EngineVersion – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt die Version der Datenbank-Engine an.

- EnhancedMonitoringResourceArn – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Amazon-Ressourcenname (ARN) des Amazon CloudWatch Logs-Protokollstreams, der die Enhanced Monitoring-Metrikdaten für die DB-Instance empfängt.

- IAMDatabaseAuthenticationEnabled – Dies ist ein boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

„True“, wenn Amazon Identity and Access Management (IAM)-Authentifizierung aktiviert ist und andernfalls „false“.

- InstanceCreateTime— Dies ist ein TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Offset von Mitternacht 1970-01-01).

  Stellt das Datum und die Uhrzeit bereit, wann die DB-Instance erstellt wurde.

- Iops— Das ist ein IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Gibt den bereitgestellten IOPS (E/A-Operationen pro Sekunde)-Wert an.

- KmsKeyId – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Nicht unterstützt: Die Verschlüsselung für DB-Instances wird vom DB-Cluster verwaltet.

- LatestRestorableTime— Dies ist ein TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Offset von Mitternacht 1970-01-01).

  Gibt den letzten Zeitpunkt an, zu dem eine Datenbank mit point-in-time restore wiederhergestellt werden kann.

- LicenseModel – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Lizenzmodellinformationen für diese DB-Instance.

- MonitoringInterval— Dies ist ein IntegerOptional Wert vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Das Intervall in Sekunden zwischen den Punkten, an denen erweiterte Überwachungsmetriken für die DB-Instance erfasst werden.

- MonitoringRoleArn – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der ARN für die IAM-Rolle, der es Neptune ermöglicht, Enhanced Monitoring-Metriken an Amazon Logs zu senden. CloudWatch

- MultiAZ – Dies ist ein boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Gibt an, ob die DB-Instance eine Multi-AZ-Bereitstellung ist.

- PendingModifiedValues – Dies ist ein PendingModifiedValues-Objekt.

  Gibt an, dass Änderungen an der DB-Instance ausstehen. Dieses Element ist nur enthalten, wenn Änderungen ausstehen. Spezifische Änderungen werden von Unterelementen identifiziert.

- PreferredBackupWindow – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den täglichen Zeitraum in koordinierter Weltzeit (UTC) an, in dem automatische Sicherungen erstellt werden, wenn automatische Sicherungen aktiviert sind, gemäß `BackupRetentionPeriod`.
- PreferredMaintenanceWindow – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den wöchentlichen Zeitraum, in dem Systemwartungen durchgeführt werden können, in UTC (Universal Coordinated Time) an.
- PromotionTier— Dies ist eine IntegerOptional Ganzzahl vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Ein Wert, der die Reihenfolge angibt, in der eine Read Replica zur primären Instance hochgestuft wird, wenn die vorhandene primäre Instance ausfällt.
- PubliclyAccessible – Dies ist ein boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Dieses Flag sollte nicht mehr verwendet werden.
- ReadReplicaDBClusterIdentifiers – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Enthält eine oder mehrere IDs der DB-Cluster, die Read Replicas dieser DB-Instance sind.
- ReadReplicaDBInstanceIdentifiers – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Enthält eine oder mehrere IDs der Read Replicas, die dieser DB-Instance zugeordnet sind.
- ReadReplicaSourceDBInstanceIdentifier – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Enthält die ID der Quell-DB-Instance, wenn diese DB-Instance eine Read Replica ist.
- SecondaryAvailabilityZone – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt, wenn vorhanden, den Namen der sekundären Availability Zone für eine DB-Instance mit Multi-AZ-Unterstützung an.

- StatusInfos – Dies ist ein Array von DBInstanceStatusInfo-Objekten.

  Der Status einer Read Replica. Wenn die Instance keine Read Replica ist, ist dies leer.
- StorageEncrypted – Dies ist ein boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Nicht unterstützt: Die Verschlüsselung für DB-Instances wird vom DB-Cluster verwaltet.
- StorageType – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Speichertyp an, der der DB-Instance zugeordnet ist.
- TdeCredentialArn – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der ARN aus dem Schlüsselspeicher, mit dem die Instance für die TDE-Verschlüsselung verknüpft ist.
- Timezone – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Nicht unterstützt
- VpcSecurityGroups – Dies ist ein Array von VpcSecurityGroupMembership-Objekten.

  Stellt eine Liste der VPC-Sicherheitsgruppenelemente bereit, zu denen die DB-Instance gehört.

`DBInstance` wird als Antwortelement verwendet für:

- Erstellen DBInstance
- Löschen DBInstance
- Modifizieren DBInstance
- Starten Sie neu DBInstance

DBInstanceStatusInfo (Struktur)

Stellt eine Liste der Statusinformationen für eine DB-Instance bereit.

Felder

- Message – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Details des Fehlers, wenn ein Fehler bei der Instance auftritt. Wenn die Instance keinen Fehlerstatus aufweist, ist dieser Wert leer.

- Normal – Dies ist ein boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Boolescher Wert, der "true" ist, wenn die Instance ordnungsgemäß funktioniert, oder "false", wenn die Instance einen Fehlerstatus aufweist.
- Status – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Status der DB-Instance. Bei einem StatusType Read-Replikat können die Werte repliziert, fehlerhaft, gestoppt oder beendet sein.
- StatusType – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Dieser Wert ist derzeit "Read-Replikation".

Bestellbare DBInstance Option (Struktur)

Enthält eine Liste der verfügbaren Optionen für eine DB-Instance.

Dieser Datentyp wird als Antwortelement in der Aktion the section called “DescribeOrderableDBInstanceOptionen” verwendet.

Felder

- AvailabilityZones – Dies ist ein Array von AvailabilityZone-Objekten.

  Eine Liste der Availability Zones für eine DB-Instance.
- DBInstanceClass – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die DB-Instance-Klasse für eine DB-Instance
- Engine – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Engine-Typ einer DB-Instance.
- EngineVersion – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Engine-Version einer DB-Instance.
- LicenseModel – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Lizenzmodell für eine DB-Instance.
- MaxIopsPerDbInstance— Dies ist eine IntegerOptional 32-Bit-Ganzzahl vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

Maximale gesamte bereitgestellte IOPS für eine DB-Instance.

- MaxIopsPerGib— Dies ist eine DoubleOptional, vom Typ: `double` (eine IEEE 754-Gleitkommazahl mit doppelter Genauigkeit).

  Maximale bereitgestellte IOPS pro GiB für eine DB-Instance.

- MaxStorageSize— Dies ist eine Zahl vom Typ: (eine IntegerOptional 32-Bit-Ganzzahl mit Vorzeichen). `integer`

  Maximale Speichergröße für eine DB-Instance.

- MinIopsPerDbInstance— Dies ist eine IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Minimale gesamte bereitgestellte IOPS für eine DB-Instance.

- MinIopsPerGib— Dies ist eine DoubleOptional, vom Typ: `double` (eine IEEE 754-Gleitkommazahl mit doppelter Genauigkeit).

  Minimale bereitgestellte IOPS pro GiB für eine DB-Instance.

- MinStorageSize— Dies ist eine Zahl vom Typ: (eine IntegerOptional 32-Bit-Ganzzahl mit Vorzeichen). `integer`

  Minimale Speichergröße für eine DB-Instance.

- MultiAZCapable – Dies ist ein boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Gibt an, ob eine DB-Instance Multi-AZ-fähig ist.

- ReadReplicaCapable – Dies ist ein boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Gibt an, ob eine DB-Instance eine Read Replica haben kann.

- StorageType – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Nicht zutreffend. In Neptune wird der Speichertyp auf DB-Cluster-Ebene verwaltet.

- SupportsEnhancedMonitoring – Dies ist ein boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Gibt an, ob eine DB-Instance erweiterte Überwachung in Abständen von 1 bis 60 Sekunden unterstützt.

- SupportsGlobalDatabases – Dies ist ein boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Ein Wert, der angibt, ob Sie globale Neptune-Datenbanken mit einer bestimmten Kombination anderer DB-Engine-Attribute verwenden können.

- SupportsIAMDatabaseAuthentication – Dies ist ein boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Gibt an, ob eine DB-Instance IAM-Datenbankauthentifizierung unterstützt.

- SupportsIops – Dies ist ein boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Gibt an, ob eine DB-Instance bereitgestellte IOPS unterstützt.

- SupportsStorageEncryption – Dies ist ein boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Gibt an, ob eine DB-Instance verschlüsselte Speicherung unterstützt.

- Vpc – Dies ist ein boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Gibt an, ob sich die DB-Instance in einer VPC befindet.

PendingModifiedValues (Struktur)

Dieser Datentyp wird als Antwortelement in der Aktion the section called “Modifizieren DBInstance” verwendet.

Felder

- AllocatedStorage— Dies ist eine IntegerOptional 32-Bit-Ganzzahl vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Enthält die neue `AllocatedStorage`-Größe für die DB-Instance, die angewendet werden wird oder derzeit angewendet wird.

- BackupRetentionPeriod— Dies ist eine IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Gibt die ausstehende Anzahl von Tagen an, die automatische Sicherungen aufbewahrt werden.

- CACertificateIdentifier – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt die ID des CA-Zertifikats für die DB-Instance an.
- DBInstanceClass – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Enthält die neue `DBInstanceClass` für die DB-Instance, die angewendet werden wird oder derzeit angewendet wird.
- DBInstanceIdentifier – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Enthält die neue `DBInstanceIdentifier` für die DB-Instance, die angewendet werden wird oder derzeit angewendet wird.
- DBSubnetGroupName – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die neue DB-Subnetzgruppe für die DB-Instance.
- EngineVersion – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt die Version der Datenbank-Engine an.
- Iops— Dies ist eine IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Gibt den Wert der neu bereitgestellten IOPS für die DB-Instance an, die angewendet werden sollen oder derzeit angewendet werden.
- MultiAZ— Dies ist ein BooleanOptional Wert vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Gibt an, dass die Single-AZ-DB-Instance in eine Multi-AZ-Bereitstellung geändert werden soll.
- PendingCloudwatchLogsExports – Dies ist ein PendingCloudwatchLogsExports-Objekt.

  Diese `PendingCloudwatchLogsExports` Struktur gibt ausstehende Änderungen an, welche CloudWatch Protokolle aktiviert und welche deaktiviert sind.
- Port— Es handelt sich um eine IntegerOptional 32-Bit-Ganzzahl vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Gibt den ausstehenden Port für die DB-Instance an.
- StorageType – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Nicht zutreffend. In Neptune wird der Speichertyp auf DB-Cluster-Ebene verwaltet.

ValidStorageOptions (Struktur)

Nicht zutreffend. In Neptune wird der Speichertyp auf DB-Cluster-Ebene verwaltet.

Felder

- IopsToStorageRatio – Dies ist ein Array von DoubleRange-Objekten.

  Nicht zutreffend. In Neptune wird der Speichertyp auf DB-Cluster-Ebene verwaltet.
- ProvisionedIops – Dies ist ein Array von Bereich-Objekten.

  Nicht zutreffend. In Neptune wird der Speichertyp auf DB-Cluster-Ebene verwaltet.
- StorageSize – Dies ist ein Array von Bereich-Objekten.

  Nicht zutreffend. In Neptune wird der Speichertyp auf DB-Cluster-Ebene verwaltet.
- StorageType – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Nicht zutreffend. In Neptune wird der Speichertyp auf DB-Cluster-Ebene verwaltet.

Gültig DBInstance ModificationsMessage (Struktur)

Informationen zu gültigen Änderungen, die Sie an Ihrer DB-Instance vornehmen können. Enthält das Ergebnis eines erfolgreichen Aufrufs der the section called “DescribeValidDBInstanceÄnderungen”-Aktion. Sie können diese Informationen verwenden, wenn Sie the section called “Modifizieren DBInstance” aufrufen.

Felder

- Storage – Dies ist ein Array von ValidStorageOptions-Objekten.

  Gültige Speicheroptionen für Ihre DB-Instance.

`ValidDBInstanceModificationsMessage` wird als Antwortelement verwendet für:

- DescribeValidDBInstanceÄnderungen

Neptune-Parameter-API

Aktionen:

- DBParameterGruppe kopieren (Aktion)
- Kopieren DBCluster ParameterGroup (Aktion)
- DBParameterGruppe erstellen (Aktion)
- Erstellen DBCluster ParameterGroup (Aktion)
- DBParameterGruppe löschen (Aktion)
- Löschen DBCluster ParameterGroup (Aktion)
- DBParameterGruppe ändern (Aktion)
- Ändern DBCluster ParameterGroup (Aktion)
- DBParameterGruppe zurücksetzen (Aktion)
- Zurücksetzen DBCluster ParameterGroup (Aktion)
- Beschreiben Sie DBParameters (Aktion)
- DBParameterGruppen beschreiben (Aktion)
- Beschreiben Sie die DBCluster Parameter (Aktion)
- Beschreiben Sie DBCluster ParameterGroups (Aktion)
- DescribeEngineDefaultParameters (Aktion)
- DescribeEngineDefaultClusterParameters (Aktion)

Strukturen:

- Parameter (Struktur)
- DBParameterGruppe (Struktur)
- DBClusterParameterGroup (Struktur)
- DBParameterGroupStatus (Struktur)

DBParameterGruppe kopieren (Aktion)

> Der AWS CLI-Name für diese API lautet:`copy-db-parameter-group`.

Kopiert die angegebene DB-Parametergruppe.

Anforderung

- SourceDBParameterGroupIdentifier (in der CLI: `--source-db-parameter-group-identifier`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Die ID oder der ARN der Quell-DB-Parametergruppe. Weitere Informationen zum Erstellen eines ARN finden Sie unter Erstellen eines Amazon-Ressourcennamens (ARN).

Einschränkungen:

- Es ist eine gültige DB-Parametergruppe erforderlich.
- Es ist eine gültige ID für die DB-Parametergruppe erforderlich, z. B. `my-db-param-group`, oder ein gültiger ARN.

- Tags (in der CLI: `--tags`) – Ein Array von Markierung Objekten.

  Die Tags, die der kopierten DB-Parametergruppe zugewiesen werden.

- TargetDBParameterGroupDescription (in der CLI: `--target-db-parameter-group-description`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Beschreibung der kopierten DB-Parametergruppe.

- TargetDBParameterGroupIdentifier (in der CLI: `--target-db-parameter-group-identifier`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die ID der kopierten DB-Parametergruppe.

  Einschränkungen:

  - Kann nicht Null, leer oder negativ sein.
  - Muss zwischen 1 und 255 Buchstaben, Ziffern oder Bindestriche enthalten.
  - Muss mit einem Buchstaben beginnen.
  - Darf nicht mit einem Bindestrich enden oder zwei aufeinanderfolgende Bindestriche enthalten.

  Beispiel: `my-db-parameter-group`

Antwort

Enthält die Details einer Amazon Neptune-DB-Parametergruppe.

Dieser Datentyp wird als Antwortelement in der Aktion the section called “Beschreiben Sie DBParameter Gruppen” verwendet.

- DBParameterGroupArn – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Amazon-Ressourcenname (ARN) für die DB-Parametergruppe.

- DBParameterGroupFamily – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Namen der DB-Parametergruppenfamilie an, mit der diese DB-Parametergruppe kompatibel ist.
- DBParameterGroupName – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Namen der DB-Parametergruppe an.
- Description – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt die vom Kunden angegebene Beschreibung für diese DB-Parametergruppe an.

Fehler

- DBParameterGroupNotFoundFault
- DBParameterGroupAlreadyExistsFault
- DBParameterGroupQuotaExceededFault

Kopieren DBCluster ParameterGroup (Aktion)

Der AWS CLI-Name für diese API lautet:`copy-db-cluster-parameter-group`.

Kopiert die angegebene DB-Cluster-Parametergruppe.

Anforderung

- SourceDBClusterParameterGroupIdentifier (in der CLI: `--source-db-cluster-parameter-group-identifier`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die ID oder der Amazon-Ressourcenname (ARN) für die Quell-DB-Cluster-Parametergruppe. Weitere Informationen zum Erstellen eines ARN finden Sie unter Erstellen eines Amazon-Ressourcennamens (ARN).

  Einschränkungen:
  - Es ist eine gültige DB-Cluster-Parametergruppe erforderlich.
  - Wenn sich die Quell-DB-Cluster-Parametergruppe und die Kopie in derselben Amazon-Region befinden, geben Sie eine gültige DB-Parametergruppen-ID (z. B. `my-db-cluster-param-group`) oder einen gültigen ARN an.

  - Wenn sich die Quell-DB-Parametergruppe und die Kopie in unterschiedlichen Amazon-Regionen befinden, geben Sie einen gültigen DB-Cluster-Parametergruppen-ARN an, z. B. `arn:aws:rds:us-east-1:123456789012:cluster-pg:custom-cluster-group1`.
- Tags (in der CLI: `--tags`) – Ein Array von Markierung Objekten.

  Die Tags, die der kopierten DB-Cluster-Parametergruppe zugewiesen werden.
- TargetDBClusterParameterGroupDescription (in der CLI: `--target-db-cluster-parameter-group-description`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Beschreibung der kopierten DB-Cluster-Parametergruppe.
- TargetDBClusterParameterGroupIdentifier (in der CLI: `--target-db-cluster-parameter-group-identifier`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die ID der kopierten DB-Cluster-Parametergruppe.

  Einschränkungen:
  - Kann nicht Null, leer oder negativ sein.
  - Muss zwischen 1 und 255 Buchstaben, Ziffern oder Bindestriche enthalten.
  - Muss mit einem Buchstaben beginnen
  - Darf nicht mit einem Bindestrich enden oder zwei aufeinanderfolgende Bindestriche enthalten

  Beispiel: `my-cluster-param-group1`

Antwort

Enthält die Details einer Amazon Neptune-DB-Cluster-Parametergruppe.

Dieser Datentyp wird als Antwortelement in der Aktion the section called “Beschreiben DBCluster ParameterGroups” verwendet.

- DBClusterParameterGroupArn – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Amazon-Ressourcenname (ARN) für die DB-Cluster-Parametergruppe.
- DBClusterParameterGroupName – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Gibt den Namen der DB-Cluster-Parametergruppe an.

- DBParameterGroupFamily – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Namen der DB-Parametergruppenfamilie an, mit der diese DB-Cluster-Parametergruppe kompatibel ist.

- Description – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt die vom Kunden angegebene Beschreibung für diese DB-Cluster-Parametergruppe an.

Fehler

- DBParameterGroupNotFoundFault
- DBParameterGroupQuotaExceededFault
- DBParameterGroupAlreadyExistsFault

DBParameterGruppe erstellen (Aktion)

Der AWS CLI-Name für diese API lautet:`create-db-parameter-group`.

Erstellt eine neue DB-Parametergruppe.

Zu Beginn wird eine DB-Parametergruppe mit den Standardparametern für die Datenbank-Engine erstellt, die von der DB-Instance verwendet wird. Um benutzerdefinierte Werte für einen der Parameter bereitzustellen, müssen Sie die Gruppe ändern, nachdem Sie sie mit Modify DBParameter Group erstellt haben. Nachdem Sie eine DB-Parametergruppe erstellt haben, müssen Sie sie mithilfe von Modify mit Ihrer DB-Instance verknüpfenDBInstance. Wenn Sie eine neue DB-Parametergruppe einer DB-Instance zuordnen, die gerade ausgeführt wird, müssen Sie die DB-Instance ohne Failover für die neue DB-Parametergruppe und die zugehörigen Einstellungen neu starten, damit die Änderungen wirksam werden.

> **Important**
>
> Nach der Erstellung einer DB-Parametergruppe sollten Sie mindestens fünf Minuten warten, bevor Sie die erste DB-Instance erstellen, die diese DB-Parametergruppe als Standardparametergruppe verwendet. So kann der Erstellvorgang in Amazon Neptune zunächst vollständig abgeschlossen werden, bevor die Parametergruppe als Standard

für eine neue DB-Instance verwendet wird. Dies ist besonders wichtig für Parameter, die kritisch für das Erstellen der Standard-Datenbank für eine DB-Instance sind, solche wie der Zeichensatz für die Standard-Datenbank über den Parameter `character_set_database`. Sie können die Option Parameter Groups der Amazon Neptune Neptune-Konsole oder den DBParameters Befehl Describe verwenden, um zu überprüfen, ob Ihre DB-Parametergruppe erstellt oder geändert wurde.

Anforderung

- DBParameterGroupFamily (in der CLI: `--db-parameter-group-family`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Name der DB-Parametergruppenfamilie. Eine DB-Parametergruppe kann nur einer einzigen DB-Parametergruppenfamilie zugeordnet und nur auf eine DB-Instance angewendet werden, in der eine Datenbank-Engine ausgeführt wird; deren Version muss mit dieser DB-Parametergruppenfamilie kompatibel sein.
- DBParameterGroupName (in der CLI: `--db-parameter-group-name`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Name der DB-Parametergruppe.

  Einschränkungen:
  - Muss zwischen 1 und 255 Buchstaben, Zahlen oder Bindestriche enthalten.
  - Muss mit einem Buchstaben beginnen
  - Darf nicht mit einem Bindestrich enden oder zwei aufeinanderfolgende Bindestriche enthalten

  > **Note**
  >
  > Dieser Wert wird als Zeichenfolge in Kleinbuchstaben gespeichert.
- Description (in der CLI: `--description`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Beschreibung für die DB-Parametergruppe.
- Tags (in der CLI: `--tags`) – Ein Array von Markierung Objekten.

  Die Tags, die der neuen DB-Parametergruppe zugewiesen werden.

Antwort

Enthält die Details einer Amazon Neptune-DB-Parametergruppe.

Dieser Datentyp wird als Antwortelement in der Aktion the section called “Beschreiben Sie DBParameter Gruppen” verwendet.

- DBParameterGroupArn – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Amazon-Ressourcenname (ARN) für die DB-Parametergruppe.
- DBParameterGroupFamily – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Namen der DB-Parametergruppenfamilie an, mit der diese DB-Parametergruppe kompatibel ist.
- DBParameterGroupName – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Namen der DB-Parametergruppe an.
- Description – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt die vom Kunden angegebene Beschreibung für diese DB-Parametergruppe an.

Fehler

- DBParameterGroupQuotaExceededFault
- DBParameterGroupAlreadyExistsFault

Erstellen DBCluster ParameterGroup (Aktion)

> Der AWS CLI-Name für diese API lautet:`create-db-cluster-parameter-group`.

Erstellt eine neue DB-Cluster-Parametergruppe.

Parameter in einer DB-Cluster-Parametergruppe gelten für alle Instances in einem DB-Cluster.

Zu Beginn wird eine DB-Cluster-Parametergruppe mit den Standardparametern für die Datenbank-Engine erstellt, die von den Instances im DB-Cluster verwendet wird. Wenn Sie benutzerdefinierte Werte für die Parameter bereitstellen möchten, müssen Sie die Gruppe nach der Erstellung mithilfe von the section called “Modifizieren DBCluster ParameterGroup” ändern. Sobald Sie eine DB-Cluster-Parametergruppe erstellt haben, müssen Sie sie mithilfe von the section called “Modifizieren

DBCluster" Ihrem DB-Cluster zuordnen. Wenn Sie eine neue DB-Cluster-Parametergruppe zu einem DB-Cluster zuordnen, der gerade ausgeführt wird, müssen Sie die DB-Instances im DB-Cluster ohne Failover für die neue DB-Cluster-Parametergruppe und die zugehörigen Einstellungen neu starten, damit die Änderungen wirksam werden.

> **Important**
>
> Nachdem Sie eine DB-Cluster-Parametergruppe erstellt haben, sollten Sie mindestens fünf Minuten warten, bevor Sie das erste DB-Cluster erstellen, das diese DB-Cluster-Parametergruppe als Standardparametergruppe verwendet. So kann der Erstellvorgang in Amazon Neptune zunächst vollständig abgeschlossen werden, bevor die DB-Cluster-Parametergruppe als Standard für einen neuen DB-Cluster verwendet wird. Dies gilt insbesondere für Parameter, die beim Erstellen der Standarddatenbank für einen DB-Cluster wichtig sind (z. B. der Zeichensatz für die Standarddatenbank, der über den Parameter `character_set_database` definiert wird). Mit der Option Parameter Groups (Parametergruppen) der Amazon Neptune-Konsole oder dem Befehl the section called "Beschreiben Sie die DBCluster Parameter" überprüfen Sie, ob die DB-Cluster-Parametergruppe erstellt oder geändert wurde.

Anforderung

- DBClusterParameterGroupName (in der CLI: `--db-cluster-parameter-group-name`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Name der DB-Cluster-Parametergruppe.

  Einschränkungen:

  - Muss mit dem Namen einer vorhandenen Datei übereinstimmen DBClusterParameterGroup.

  > **Note**
  >
  > Dieser Wert wird als Zeichenfolge in Kleinbuchstaben gespeichert.

- DBParameterGroupFamily (in der CLI: `--db-parameter-group-family`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Name der DB-Cluster-Parametergruppenfamilie. Eine DB-Cluster-Parametergruppe kann nur einer einzigen DB-Cluster-Parametergruppenfamilie zugeordnet und nur auf einen DB-Cluster

angewendet werden, in dem eine Datenbank-Engine ausgeführt wird; deren Version muss mit dieser DB-Cluster-Parametergruppenfamilie kompatibel sein.

- Description (in der CLI: `--description`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Beschreibung der DB-Cluster-Parametergruppe.

- Tags (in der CLI: `--tags`) – Ein Array von Markierung Objekten.

  Die Tags, die der neuen DB-Cluster-Parametergruppe zugewiesen werden.

Antwort

Enthält die Details einer Amazon Neptune-DB-Cluster-Parametergruppe.

Dieser Datentyp wird als Antwortelement in der Aktion the section called “Beschreiben DBCluster ParameterGroups” verwendet.

- DBClusterParameterGroupArn – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Amazon-Ressourcenname (ARN) für die DB-Cluster-Parametergruppe.

- DBClusterParameterGroupName – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Namen der DB-Cluster-Parametergruppe an.

- DBParameterGroupFamily – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Namen der DB-Parametergruppenfamilie an, mit der diese DB-Cluster-Parametergruppe kompatibel ist.

- Description – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt die vom Kunden angegebene Beschreibung für diese DB-Cluster-Parametergruppe an.

Fehler

- DBParameterGroupQuotaExceededFault
- DBParameterGroupAlreadyExistsFault

DBParameterGruppe löschen (Aktion)

Der AWS CLI-Name für diese API lautet:`delete-db-parameter-group`.

Löscht eine angegebene DBParameter Gruppe. Die zu löschende DBParameter Gruppe kann keiner DB-Instance zugeordnet werden.

Anforderung

- DBParameterGroupName (in der CLI: `--db-parameter-group-name`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Name der DB-Parametergruppe.

  Einschränkungen:
  - Muss dem Namen einer vorhandenen DB-Parametergruppe entsprechen.
  - Eine Standard-DB-Parametergruppe kann nicht gelöscht werden.
  - Kann keinen DB-Instances zugeordnet werden.

Antwort

- Keine Antwortparameter.

Fehler

- Ungültig DBParameter GroupStateFault
- DBParameterGroupNotFoundFault

Löschen DBCluster ParameterGroup (Aktion)

Der AWS CLI-Name für diese API lautet:`delete-db-cluster-parameter-group`.

Löscht eine angegebene DB-Cluster-Parametergruppe. Die zu löschende DB-Cluster-Parametergruppe kann keinen DB-Clustern zugeordnet werden.

Anforderung

- DBClusterParameterGroupName (in der CLI: `--db-cluster-parameter-group-name`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Der Name der DB-Cluster-Parametergruppe.

Einschränkungen:

- Muss dem Namen einer vorhandenen DB-Cluster-Parametergruppe entsprechen.
- Eine Standard-DB-Cluster-Parametergruppe kann nicht gelöscht werden.
- Kann keinen DB-Clustern zugeordnet werden.

Antwort

- Keine Antwortparameter.

Fehler

- Ungültig DBParameter GroupStateFault
- DBParameterGroupNotFoundFault

DBParameterGruppe ändern (Aktion)

Der AWS CLI-Name für diese API lautet:`modify-db-parameter-group`.

Ändert die Parameter einer DB-Parametergruppe. Wenn Sie mehr als einen Parameter ändern möchten, übergeben Sie eine Liste mit den folgenden Optionen: `ParameterName`, `ParameterValue` und `ApplyMethod`. Pro Anforderung können maximal 20 Parameter geändert werden.

> **Note**
>
> Änderungen an dynamischen Parametern werden sofort angewendet. Änderungen an statischen Parametern erfordern einen Neustart ohne Failover auf die DB-Instance, die der Parametergruppe zugeordnet ist. Erst danach werden die Änderungen wirksam.

> **Important**
>
> Nachdem Sie eine DB-Parametergruppe geändert haben, sollten Sie mindestens fünf Minuten warten, bevor Sie die erste DB-Instance erstellen, die diese DB-Parametergruppe als

Standardparametergruppe verwendet. So kann der Änderungsvorgang in Amazon Neptune zunächst vollständig abgeschlossen werden, bevor die Parametergruppe als Standard für eine neue DB-Instance verwendet wird. Dies ist besonders wichtig für Parameter, die kritisch für das Erstellen der Standard-Datenbank für eine DB-Instance sind, solche wie der Zeichensatz für die Standard-Datenbank über den Parameter `character_set_database`. Sie können die Option Parameter Groups der Amazon Neptune Neptune-Konsole oder den DBParameters Befehl Describe verwenden, um zu überprüfen, ob Ihre DB-Parametergruppe erstellt oder geändert wurde.

Anforderung

- DBParameterGroupName (in der CLI: `--db-parameter-group-name`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Name der DB-Parametergruppe.

  Einschränkungen:

  - Falls angegeben, muss sie mit dem Namen einer vorhandenen DBParameter Gruppe übereinstimmen.
- Parameters (in der CLI: `--parameters`) – Erforderlich: Ein Array von Parameter-Objekten.

  Ein Array von Parameternamen, Werten und der Anwendungsmethode für die Parameteraktualisierung. Es muss mindestens ein Name, ein Wert und eine Anwendungsmethode für den Parameter angegeben werden; nachfolgende Argumente sind optional. Pro Anforderung können maximal 20 Parameter geändert werden.

  Gültige Werte (für die Anwendungsmethode): `immediate | pending-reboot`

  > **Note**
  >
  > Den Wert "immediate" können Sie nur mit dynamischen Parametern verwenden. Den Wert "pending-reboot" können Sie für dynamische und statische Parameter verwenden; die Änderungen werden beim Neustart der DB-Instance ohne Failover wirksam.

Antwort

- DBParameterGroupName – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Namen der DB-Parametergruppe an.

Fehler

- DBParameterGroupNotFoundFault
- Ungültig DBParameter GroupStateFault

Ändern DBCluster ParameterGroup (Aktion)

Der AWS CLI-Name für diese API lautet:`modify-db-cluster-parameter-group`.

Ändert die Parameter einer DB-Cluster-Parametergruppe. Wenn Sie mehr als einen Parameter ändern möchten, übergeben Sie eine Liste mit den folgenden Optionen: `ParameterName`, `ParameterValue` und `ApplyMethod`. Pro Anforderung können maximal 20 Parameter geändert werden.

> **Note**
>
> Änderungen an dynamischen Parametern werden sofort angewendet. Änderungen an statischen Parametern erfordern einen Neustart ohne Failover auf den DB-Cluster, der der Parametergruppe zugeordnet ist. Erst danach werden die Änderungen wirksam.

> **Important**
>
> Nachdem Sie eine DB-Cluster-Parametergruppe erstellt haben, sollten Sie mindestens fünf Minuten warten, bevor Sie das erste DB-Cluster erstellen, das diese DB-Cluster-Parametergruppe als Standardparametergruppe verwendet. So kann der Erstellvorgang in Amazon Neptune zunächst vollständig abgeschlossen werden, bevor die Parametergruppe als Standard für einen neuen DB-Cluster verwendet wird. Dies gilt insbesondere für Parameter, die beim Erstellen der Standarddatenbank für einen DB-Cluster wichtig sind (z. B. der Zeichensatz für die Standarddatenbank, der über den Parameter `character_set_database` definiert wird). Mit der Option Parameter Groups (Parametergruppen) der Amazon Neptune-Konsole oder dem Befehl the section

called “Beschreiben Sie die DBCluster Parameter” überprüfen Sie, ob die DB-Cluster-Parametergruppe erstellt oder geändert wurde.

Anforderung

- DBClusterParameterGroupName (in der CLI: `--db-cluster-parameter-group-name`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Name der zu ändernden DB-Cluster-Parametergruppe.
- Parameters (in der CLI: `--parameters`) – Erforderlich: Ein Array von Parameter-Objekten.

  Eine Liste der Parameter in der zu ändernden DB-Cluster-Parametergruppe.

Antwort

- DBClusterParameterGroupName – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Name der DB-Cluster-Parametergruppe.

  Einschränkungen:
  - Muss 1 bis 255 Buchstaben oder Zahlen enthalten.
  - Muss mit einem Buchstaben beginnen
  - Darf nicht mit einem Bindestrich enden oder zwei aufeinanderfolgende Bindestriche enthalten

  > **Note**
  >
  > Dieser Wert wird als Zeichenfolge in Kleinbuchstaben gespeichert.

Fehler

- DBParameterGroupNotFoundFault
- Ungültig DBParameter GroupStateFault

DBParameterGruppe zurücksetzen (Aktion)

Der AWS CLI-Name für diese API lautet:`reset-db-parameter-group`.

Ändert die Parameter einer DB-Parametergruppe auf den engine/system Standardwert. Wenn Sie bestimmte Parameter zurücksetzen möchten, geben Sie eine Liste mit folgenden Optionen an: `ParameterName` und `ApplyMethod`. Wenn Sie die gesamte DB-Parametergruppe zurücksetzen möchten, geben Sie den Namen der `DBParameterGroup` und die `ResetAllParameters`-Parameter an. Wenn Sie die gesamte Gruppe zurücksetzen, werden dynamische Parameter sofort aktualisiert und statische Parameter auf `pending-reboot` gesetzt, damit sie beim nächsten Neustart der DB-Instance oder bei der `RebootDBInstance`-Anforderung wirksam werden.

Anforderung

- DBParameterGroupName (in der CLI: `--db-parameter-group-name`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Name der DB-Parametergruppe.

  Einschränkungen:

  - Muss mit dem Namen einer vorhandenen DBParameter Gruppe übereinstimmen.
- Parameters (in der CLI: `--parameters`) – Ein Array von Parameter Objekten.

  Wenn Sie die gesamte DB-Parametergruppe zurücksetzen möchten, geben Sie den Namen der `DBParameterGroup` und die `ResetAllParameters`-Parameter an. Wenn Sie bestimmte Parameter zurücksetzen möchten, geben Sie eine Liste mit folgenden Optionen an: `ParameterName` und `ApplyMethod`. Pro Anforderung können maximal 20 Parameter geändert werden.

  Gültige Werte (Anwendungsmethode): `pending-reboot`
- ResetAllParameters (in der CLI: `--reset-all-parameters`) – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Gibt an, ob alle Parameter in der DB-Parametergruppe auf ihre Standardwerte zurückgesetzt werden sollen oder nicht (`true` oder `false`).

  Standard: `true`

Antwort

- DBParameterGroupName – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Namen der DB-Parametergruppe an.

Fehler

- Ungültig DBParameter GroupStateFault
- DBParameterGroupNotFoundFault

Zurücksetzen DBCluster ParameterGroup (Aktion)

Der AWS CLI-Name für diese API lautet:`reset-db-cluster-parameter-group`.

Setzt die Parameter einer DB-Cluster-Parametergruppe auf die Standardwerte zurück. Wenn Sie bestimmte Parameter zurücksetzen möchten, übergeben Sie eine Liste mit folgenden Optionen: `ParameterName` und `ApplyMethod`. Wenn Sie die gesamte DB-Cluster-Parametergruppe zurücksetzen möchten, geben Sie den Namen der `DBClusterParameterGroupName` und die `ResetAllParameters`-Parameter an.

Wenn Sie die gesamte Gruppe zurücksetzen, werden dynamische Parameter sofort aktualisiert und statische Parameter auf `pending-reboot` gesetzt, damit sie beim nächsten Neustart der DB-Instance oder bei der the section called “Starten Sie neu DBInstance”-Anforderung wirksam werden. Sie müssen the section called “Starten Sie neu DBInstance” für alle DB-Instances in Ihrem DB-Cluster aufrufen, auf die der aktualisierte statische Parameter angewendet werden soll.

Anforderung

- DBClusterParameterGroupName (in der CLI: `--db-cluster-parameter-group-name`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Name der zurückzusetzenden DB-Cluster-Parametergruppe.
- Parameters (in der CLI: `--parameters`) – Ein Array von Parameter Objekten.

  Eine Liste von Parameternamen in der DB-Cluster-Parametergruppe, die auf ihre Standardwerte zurücksetzt werden soll. Sie können den Parameter nicht verwenden, wenn der Parameter `ResetAllParameters` auf `true` festgelegt ist.
- ResetAllParameters (in der CLI: `--reset-all-parameters`) – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

Dieser Wert ist auf `true` festgelegt, wenn alle Parameter in der DB-Cluster-Parametergruppe auf ihre Standardwerte zurückgesetzt werden sollen, und andernfalls auf `false`. Diesen Parameter können Sie nicht verwenden, wenn es eine Liste von Parameternamen für `Parameters` gibt.

Antwort

- DBClusterParameterGroupName – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Name der DB-Cluster-Parametergruppe.

  Einschränkungen:

  - Muss 1 bis 255 Buchstaben oder Zahlen enthalten.
  - Muss mit einem Buchstaben beginnen
  - Darf nicht mit einem Bindestrich enden oder zwei aufeinanderfolgende Bindestriche enthalten

  > **Note**
  >
  > Dieser Wert wird als Zeichenfolge in Kleinbuchstaben gespeichert.

Fehler

- Ungültig DBParameter GroupStateFault
- DBParameterGroupNotFoundFault

Beschreiben Sie DBParameters (Aktion)

Der AWS CLI-Name für diese API lautet:`describe-db-parameters`.

Gibt die detaillierte Parameterliste für eine bestimmte DB-Parametergruppe zurück.

Anforderung

- DBParameterGroupName (in der CLI: `--db-parameter-group-name`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Name einer bestimmten DB-Parametergruppe, deren Details zurückgegeben werden sollen.

Einschränkungen:

- Falls angegeben, muss er mit dem Namen einer vorhandenen DBParameter Gruppe übereinstimmen.

- Filters (in der CLI: `--filters`) – Ein Array von Filter Objekten.

  Dieser Parameter wird derzeit nicht unterstützt.

- Marker (in der CLI: `--marker`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Ein optionales Paginierungstoken, das von einer vorherigen `DescribeDBParameters`-Anforderung bereitgestellt wird. Wenn Sie diesen Parameter angeben, enthält die Antwort nur die Datensätze zwischen der Markierung und dem durch `MaxRecords` angegebenen Wert.

- MaxRecords(in der CLI:`--max-records`) — ein IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Die maximale Anzahl der in der Antwort zurückgegebenen Datensätze. Wenn mehrere Datensätze vorhanden sind, als der Wert `MaxRecords` angibt, ist ein Paginierungstoken mit dem Namen einer Markierung in der Antwort enthalten, sodass die verbleibenden Ergebnisse abgerufen werden können.

  Standard: 100

  Einschränkungen: Mindestwert 20, Höchstwert 100.

- Source (in der CLI: `--source`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Parametertypen, die zurückgegeben werden sollen.

  Standardmäßig werden alle Parametertypen zurückgegeben.

  Zulässige Werte: `user | system | engine-default`

Antwort

- Marker – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Ein optionales Paginierungstoken, das von einer vorherigen Anforderung bereitgestellt wird. Wenn Sie diesen Parameter angeben, enthält die Antwort nur die Datensätze zwischen der Markierung und dem durch `MaxRecords` angegebenen Wert.

- Parameters – Ein Array mit Parameter-Objekten.

  Eine Liste der the section called “Parameter”-Werte-

Fehler

- DBParameterGroupNotFoundFault

DBParameterGruppen beschreiben (Aktion)

Der AWS CLI-Name für diese API lautet:`describe-db-parameter-groups`.

Gibt eine Liste von `DBParameterGroup`-Beschreibungen zurück. Wenn ein `DBParameterGroupName` angegeben wurde, enthält die Liste nur die Beschreibung der angegebenen DB-Parametergruppe.

Anforderung

- DBParameterGroupName (in der CLI: `--db-parameter-group-name`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Name einer bestimmten DB-Parametergruppe, deren Details zurückgegeben werden sollen.

  Einschränkungen:

  - Falls angegeben, muss er mit dem Namen einer vorhandenen Datei übereinstimmen DBClusterParameterGroup.
- Filters (in der CLI: `--filters`) – Ein Array von Filter Objekten.

  Dieser Parameter wird derzeit nicht unterstützt.
- Marker (in der CLI: `--marker`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Ein optionales Paginierungstoken, das von einer vorherigen `DescribeDBParameterGroups`-Anforderung bereitgestellt wird. Wenn Sie diesen Parameter angeben, enthält die Antwort nur die Datensätze zwischen der Markierung und dem durch `MaxRecords` angegebenen Wert.

- MaxRecords(in der CLI:`--max-records`) — ein IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Die maximale Anzahl der in der Antwort zurückgegebenen Datensätze. Wenn mehrere Datensätze vorhanden sind, als der Wert `MaxRecords` angibt, ist ein Paginierungstoken mit dem Namen einer Markierung in der Antwort enthalten, sodass die verbleibenden Ergebnisse abgerufen werden können.

  Standard: 100

  Einschränkungen: Mindestwert 20, Höchstwert 100.

Antwort

- DBParameterGroups – Ein Array mit DBParameterGruppe-Objekten.

  Eine Liste der the section called “DBParameterGruppe”-Instances.

- Marker – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Ein optionales Paginierungstoken, das von einer vorherigen Anforderung bereitgestellt wird. Wenn Sie diesen Parameter angeben, enthält die Antwort nur die Datensätze zwischen der Markierung und dem durch `MaxRecords` angegebenen Wert.

Fehler

- DBParameterGroupNotFoundFault

Beschreiben Sie die DBCluster Parameter (Aktion)

> Der AWS CLI-Name für diese API lautet:`describe-db-cluster-parameters`.

Gibt die detaillierte Parameterliste für eine bestimmte DB-Cluster-Parametergruppe zurück.

Anforderung

- DBClusterParameterGroupName (in der CLI: `--db-cluster-parameter-group-name`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Der Name einer bestimmten DB-Cluster-Parametergruppe, deren Parameterdetails zurückgegeben werden sollen.

Einschränkungen:

- Falls angegeben, muss er mit dem Namen einer vorhandenen Datei übereinstimmen DBClusterParameterGroup.

- Filters (in der CLI: `--filters`) – Ein Array von Filter Objekten.

  Dieser Parameter wird derzeit nicht unterstützt.

- Marker (in der CLI: `--marker`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Ein optionales Paginierungstoken, das von einer vorherigen `DescribeDBClusterParameters`-Anforderung bereitgestellt wird. Wenn Sie diesen Parameter angeben, enthält die Antwort nur die Datensätze zwischen der Markierung und dem durch `MaxRecords` angegebenen Wert.

- MaxRecords(in der CLI:`--max-records`) — ein IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Die maximale Anzahl der in der Antwort zurückgegebenen Datensätze. Wenn mehrere Datensätze vorhanden sind, als der Wert `MaxRecords` angibt, ist ein Paginierungstoken mit dem Namen einer Markierung in der Antwort enthalten, sodass die verbleibenden Ergebnisse abgerufen werden können.

  Standard: 100

  Einschränkungen: Mindestwert 20, Höchstwert 100.

- Source (in der CLI: `--source`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Dieser Wert gibt an, dass nur die Parameter für eine bestimmte Quelle zurückgegeben werden sollen. Parameterquellen können `engine`, `service` oder `customer` sein.

Antwort

- Marker – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Ein optionales Paginierungstoken, das in einer früheren Anfrage zur Beschreibung der DBCluster Parameter bereitgestellt wurde. Wenn Sie diesen Parameter angeben, enthält die Antwort nur die Datensätze zwischen der Markierung und dem durch `MaxRecords` angegebenen Wert.

- Parameters – Ein Array mit Parameter-Objekten.

  Gibt eine Liste der Parameter für die DB-Cluster-Parametergruppe zurück.

Fehler

- DBParameterGroupNotFoundFault

Beschreiben Sie DBCluster ParameterGroups (Aktion)

Der AWS CLI-Name für diese API lautet:`describe-db-cluster-parameter-groups`.

Gibt eine Liste von `DBClusterParameterGroup`-Beschreibungen zurück. Wenn ein `DBClusterParameterGroupName`-Parameter angegeben wurde, enthält die Liste nur die Beschreibung der angegebenen DB-Cluster-Parametergruppe.

Anforderung

- DBClusterParameterGroupName (in der CLI: `--db-cluster-parameter-group-name`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Name einer bestimmten DB-Cluster-Parametergruppe, deren Details zurückgegeben werden sollen.

  Einschränkungen:

  - Falls angegeben, muss er mit dem Namen einer vorhandenen Datei übereinstimmen DBClusterParameterGroup.
- Filters (in der CLI: `--filters`) – Ein Array von Filter Objekten.

  Dieser Parameter wird derzeit nicht unterstützt.
- Marker (in der CLI: `--marker`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Ein optionales Paginierungstoken, das von einer vorherigen `DescribeDBClusterParameterGroups`-Anforderung bereitgestellt wird. Wenn Sie diesen

Parameter angeben, enthält die Antwort nur die Datensätze zwischen der Markierung und dem durch `MaxRecords` angegebenen Wert.

- MaxRecords(in der CLI:`--max-records`) — ein IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Die maximale Anzahl der in der Antwort zurückgegebenen Datensätze. Wenn mehrere Datensätze vorhanden sind, als der Wert `MaxRecords` angibt, ist ein Paginierungstoken mit dem Namen einer Markierung in der Antwort enthalten, sodass die verbleibenden Ergebnisse abgerufen werden können.

  Standard: 100

  Einschränkungen: Mindestwert 20, Höchstwert 100.

Antwort

- DBClusterParameterGroups – Ein Array mit DBClusterParameterGroup-Objekten.

  Eine Liste der DB-Cluster-Parametergruppen.

- Marker – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Ein optionales Paginierungstoken, das von einer vorherigen `DescribeDBClusterParameterGroups`-Anforderung bereitgestellt wird. Wenn Sie diesen Parameter angeben, enthält die Antwort nur die Datensätze zwischen der Markierung und dem durch `MaxRecords` angegebenen Wert.

Fehler

- DBParameterGroupNotFoundFault

DescribeEngineDefaultParameters (Aktion)

Der AWS CLI-Name für diese API lautet:`describe-engine-default-parameters`.

Gibt die Standard-Engine- und System-Parameterinformationen für die angegebene Datenbank-Engine zurück.

Anforderung

- DBParameterGroupFamily (in der CLI: `--db-parameter-group-family`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Name der DB-Cluster-Parametergruppenfamilie.

- Filters (in der CLI: `--filters`) – Ein Array von Filter Objekten.

  Wird derzeit nicht unterstützt.

- Marker (in der CLI: `--marker`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Ein optionales Paginierungstoken, das von einer vorherigen `DescribeEngineDefaultParameters`-Anforderung bereitgestellt wird. Wenn Sie diesen Parameter angeben, enthält die Antwort nur die Datensätze zwischen der Markierung und dem durch `MaxRecords` angegebenen Wert.

- MaxRecords(in der CLI:`--max-records`) — ein IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Die maximale Anzahl der in der Antwort zurückgegebenen Datensätze. Wenn mehrere Datensätze vorhanden sind, als der Wert `MaxRecords` angibt, ist ein Paginierungstoken mit dem Namen einer Markierung in der Antwort enthalten, sodass die verbleibenden Ergebnisse abgerufen werden können.

  Standard: 100

  Einschränkungen: Mindestwert 20, Höchstwert 100.

Antwort

Enthält das Ergebnis eines erfolgreichen Aufrufs der the section called “DescribeEngineDefaultParameters”-Aktion.

- DBParameterGroupFamily – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Namen der DB-Parametergruppenfamilie an, auf die die Standard-Parameter der Engine angewendet werden.

- Marker – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Ein optionales Paginierungstoken, das in einer früheren EngineDefaults Anfrage bereitgestellt wurde. Wenn Sie diesen Parameter angeben, enthält die Antwort nur die Datensätze zwischen der Markierung und dem durch `MaxRecords` angegebenen Wert.

- Parameters – Ein Array mit Parameter-Objekten.

  Enthält eine Liste der Standardparameter der Engine.

DescribeEngineDefaultClusterParameters (Aktion)

Der AWS CLI-Name für diese API lautet:`describe-engine-default-cluster-parameters`.

Gibt die Standard-Engine- und System-Parameterinformationen für die Cluster-Datenbank-Engine zurück.

Anforderung

- DBParameterGroupFamily (in der CLI: `--db-parameter-group-family`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Name der DB-Cluster-Parametergruppenfamilie, für die Parameterinformationen der Engine zurückgegeben werden sollen.
- Filters (in der CLI: `--filters`) – Ein Array von Filter Objekten.

  Dieser Parameter wird derzeit nicht unterstützt.
- Marker (in der CLI: `--marker`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Ein optionales Paginierungstoken, das von einer vorherigen `DescribeEngineDefaultClusterParameters`-Anforderung bereitgestellt wird. Wenn Sie diesen Parameter angeben, enthält die Antwort nur die Datensätze zwischen der Markierung und dem durch `MaxRecords` angegebenen Wert.
- MaxRecords(in der CLI:`--max-records`) — ein IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Die maximale Anzahl der in der Antwort zurückgegebenen Datensätze. Wenn mehrere Datensätze vorhanden sind, als der Wert `MaxRecords` angibt, ist ein Paginierungstoken mit dem Namen einer

Markierung in der Antwort enthalten, sodass die verbleibenden Ergebnisse abgerufen werden können.

Standard: 100

Einschränkungen: Mindestwert 20, Höchstwert 100.

Antwort

Enthält das Ergebnis eines erfolgreichen Aufrufs der the section called "DescribeEngineDefaultParameters"-Aktion.

- DBParameterGroupFamily – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Namen der DB-Parametergruppenfamilie an, auf die die Standard-Parameter der Engine angewendet werden.
- Marker – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Ein optionales Paginierungstoken, das in einer früheren EngineDefaults Anfrage bereitgestellt wurde. Wenn Sie diesen Parameter angeben, enthält die Antwort nur die Datensätze zwischen der Markierung und dem durch `MaxRecords` angegebenen Wert.
- Parameters – Ein Array mit Parameter-Objekten.

  Enthält eine Liste der Standardparameter der Engine.

Strukturen:

Parameter (Struktur)

Gibt einen Parameter an.

Felder

- AllowedValues – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den gültigen Wertebereich für den Parameter an.
- ApplyMethod— Dies ist eine Zeichenfolge ApplyMethod vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

Gibt an, wann Parameteraktualisierungen angewendet werden können.

- ApplyType – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Engine-spezifischen Parametertyp an.

- DataType – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den gültigen Datentyp für den Parameter an.

- Description – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Stellt eine Beschreibung des Parameters bereit.

- IsModifiable – Dies ist ein boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Gibt an, ob der Parameter geändert werden kann oder nicht (`true` oder `false`). Einige Parameter wirken sich auf die Sicherheit oder die betrieblichen Abläufe aus und können nicht geändert werden.

- MinimumEngineVersion – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die älteste Engine-Version, auf die der Parameter angewendet werden kann.

- ParameterName – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Namen des Parameters an.

- ParameterValue – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Wert des Parameters an.

- Source – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt die Quelle des Parameterwerts an.

DBParameterGruppe (Struktur)

Enthält die Details einer Amazon Neptune-DB-Parametergruppe.

Dieser Datentyp wird als Antwortelement in der Aktion the section called “Beschreiben Sie DBParameter Gruppen” verwendet.

Felder

- DBParameterGroupArn – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Amazon-Ressourcenname (ARN) für die DB-Parametergruppe.
- DBParameterGroupFamily – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Namen der DB-Parametergruppenfamilie an, mit der diese DB-Parametergruppe kompatibel ist.
- DBParameterGroupName – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Namen der DB-Parametergruppe an.
- Description – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt die vom Kunden angegebene Beschreibung für diese DB-Parametergruppe an.

`DBParameterGroup` wird als Antwortelement verwendet für:

- DBParameterGruppe kopieren
- DBParameterGruppe erstellen

DBClusterParameterGroup (Struktur)

Enthält die Details einer Amazon Neptune-DB-Cluster-Parametergruppe.

Dieser Datentyp wird als Antwortelement in der Aktion the section called “Beschreiben DBCluster ParameterGroups” verwendet.

Felder

- DBClusterParameterGroupArn – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Amazon-Ressourcenname (ARN) für die DB-Cluster-Parametergruppe.

- DBClusterParameterGroupName – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Namen der DB-Cluster-Parametergruppe an.
- DBParameterGroupFamily – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Namen der DB-Parametergruppenfamilie an, mit der diese DB-Cluster-Parametergruppe kompatibel ist.
- Description – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt die vom Kunden angegebene Beschreibung für diese DB-Cluster-Parametergruppe an.

`DBClusterParameterGroup` wird als Antwortelement verwendet für:

- Kopieren DBCluster ParameterGroup
- Erstellen DBCluster ParameterGroup

DBParameterGroupStatus (Struktur)

Der Status der DB-Parametergruppe.

Dieser Datentyp wird als Antwortelement in folgenden Aktionen verwendet:

- the section called “Erstellen DBInstance”
- the section called “Löschen DBInstance”
- the section called “Modifizieren DBInstance”
- the section called “Starten Sie neu DBInstance”

Felder

- DBParameterGroupName – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Name der DB-Parametergruppe.
- ParameterApplyStatus – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Der Status von Parameteraktualisierungen.

Neptune-Subnetz-API

Aktionen:

- DBSubnetGruppe erstellen (Aktion)
- DBSubnetGruppe löschen (Aktion)
- DBSubnetGruppe ändern (Aktion)
- DBSubnetGruppen beschreiben (Aktion)

Strukturen:

- Subnetz (Struktur)
- DBSubnetGruppe (Struktur)

DBSubnetGruppe erstellen (Aktion)

Der AWS CLI-Name für diese API lautet:`create-db-subnet-group`.

Erstellt eine neue DB-Subnetzgruppe. DB-Subnetzgruppen müssen mindestens ein Subnetz von mindestens zwei Subnetzen AZs in der Amazon-Region enthalten.

Anforderung

- DBSubnetGroupDescription (in der CLI: `--db-subnet-group-description`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Beschreibung für die Datenbank-Subnetzgruppe.
- DBSubnetGroupName (in der CLI: `--db-subnet-group-name`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Name für die DB-Subnetzgruppe. Dieser Wert wird als Zeichenfolge in Kleinbuchstaben gespeichert.

  Einschränkungen: Darf nicht mehr als 255 Buchstaben, Ziffern, Punkte, Unterstriche, Leerzeichen und Bindestriche enthalten. Der Name darf nicht default sein.

Beispiel: `mySubnetgroup`

- SubnetIds (in der CLI: `--subnet-ids`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Das EC2 Subnetz IDs für die DB-Subnetzgruppe.

- Tags (in der CLI: `--tags`) – Ein Array von Markierung Objekten.

  Die Tags, die der neuen DB-Subnetzgruppe zugewiesen werden.

Antwort

Enthält die Details einer Amazon Neptune-DB-Subnetzgruppe.

Dieser Datentyp wird als Antwortelement in der Aktion the section called “Beschreiben Sie DBSubnet Gruppen” verwendet.

- DBSubnetGroupArn – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Amazon-Ressourcenname (ARN) für die DB-Subnetzgruppe.

- DBSubnetGroupDescription – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Stellt die Beschreibung der DB-Subnetzgruppe bereit.

- DBSubnetGroupName – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Name der DB-Subnetzgruppe.

- SubnetGroupStatus – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Stellt den Status der DB-Subnetzgruppe bereit.

- Subnets – Ein Array mit Subnetz-Objekten.

  Enthält eine Liste von the section called “Subnetz”-Elementen.

- VpcId – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Stellt das VpcId der DB-Subnetzgruppe bereit.

Fehler

- DBSubnetGroupAlreadyExistsFault

- DBSubnetGroupQuotaExceededFault
- DBSubnetQuotaExceededFault
- DBSubnetGroupDoesNotCoverEnoughAZs
- InvalidSubnet

DBSubnetGruppe löschen (Aktion)

Der AWS CLI-Name für diese API lautet:`delete-db-subnet-group`.

Löscht eine DB-Subnetzgruppe.

> **Note**
>
> Die angegebene Datenbanksubnetzgruppe muss nicht mit beliebigen DB-Instances verknüpft werden.

Anforderung

- DBSubnetGroupName (in der CLI: `--db-subnet-group-name`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Name der zu löschenden Datenbanksubnetzgruppe.

  > **Note**
  >
  > Sie können die Standardsubnetzgruppe nicht löschen.

  Einschränkungen:

  Einschränkungen: Muss mit dem Namen einer vorhandenen DBSubnet Gruppe übereinstimmen. Der Name darf nicht default sein.

  Beispiel: `mySubnetgroup`

Antwort

- Keine Antwortparameter.

Fehler

- Ungültig DBSubnet GroupStateFault
- Ungültig DBSubnet StateFault
- DBSubnetGroupNotFoundFault

DBSubnetGruppe ändern (Aktion)

Der AWS CLI-Name für diese API lautet:`modify-db-subnet-group`.

Ändert eine vorhandene DB-Subnetzgruppe. DB-Subnetzgruppen müssen mindestens ein Subnetz von mindestens zwei Subnetzen AZs in der Amazon-Region enthalten.

Anforderung

- DBSubnetGroupDescription (in der CLI: `--db-subnet-group-description`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Beschreibung für die Datenbank-Subnetzgruppe.
- DBSubnetGroupName (in der CLI: `--db-subnet-group-name`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Name für die DB-Subnetzgruppe. Dieser Wert wird als Zeichenfolge in Kleinbuchstaben gespeichert. Sie können die Standardsubnetzgruppe nicht ändern.

  Einschränkungen: Muss mit dem Namen einer vorhandenen DBSubnet Gruppe übereinstimmen. Der Name darf nicht default sein.

  Beispiel: `mySubnetgroup`
- SubnetIds (in der CLI: `--subnet-ids`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Das EC2 Subnetz IDs für die DB-Subnetzgruppe.

Antwort

Enthält die Details einer Amazon Neptune-DB-Subnetzgruppe.

Dieser Datentyp wird als Antwortelement in der Aktion the section called “Beschreiben Sie DBSubnet Gruppen” verwendet.

- DBSubnetGroupArn – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Amazon-Ressourcenname (ARN) für die DB-Subnetzgruppe.
- DBSubnetGroupDescription – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Stellt die Beschreibung der DB-Subnetzgruppe bereit.
- DBSubnetGroupName – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Name der DB-Subnetzgruppe.
- SubnetGroupStatus – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Stellt den Status der DB-Subnetzgruppe bereit.
- Subnets – Ein Array mit Subnetz-Objekten.

  Enthält eine Liste von the section called “Subnetz”-Elementen.
- VpcId – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Stellt das VpcId der DB-Subnetzgruppe bereit.

Fehler

- DBSubnetGroupNotFoundFault
- DBSubnetQuotaExceededFault
- SubnetAlreadyInUse
- DBSubnetGroupDoesNotCoverEnoughAZs
- InvalidSubnet

DBSubnetGruppen beschreiben (Aktion)

Der AWS CLI-Name für diese API lautet:`describe-db-subnet-groups`.

Gibt eine Liste von DBSubnet Gruppenbeschreibungen zurück. Wenn a angegeben DBSubnet GroupName ist, enthält die Liste nur die Beschreibungen der angegebenen DBSubnet Gruppe.

Eine Übersicht über CIDR-Bereiche finden Sie im Wikipedia-Tutorial.

Anforderung

- DBSubnetGroupName (in der CLI: `--db-subnet-group-name`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Name der DB-Subnetzgruppe, deren Details Sie anzeigen möchten.
- Filters (in der CLI: `--filters`) – Ein Array von Filter Objekten.

  Dieser Parameter wird derzeit nicht unterstützt.
- Marker (in der CLI: `--marker`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Ein optionales Paginierungstoken, das in einer früheren Anfrage zur Beschreibung von DBSubnet Gruppen bereitgestellt wurde. Wenn Sie diesen Parameter angeben, enthält die Antwort nur die Datensätze zwischen der Markierung und dem durch `MaxRecords` angegebenen Wert.
- MaxRecords(in der CLI:`--max-records`) — ein IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Die maximale Anzahl der in der Antwort zurückgegebenen Datensätze. Wenn mehrere Datensätze vorhanden sind, als der Wert `MaxRecords` angibt, ist ein Paginierungstoken mit dem Namen einer Markierung in der Antwort enthalten, sodass die verbleibenden Ergebnisse abgerufen werden können.

  Standard: 100

  Einschränkungen: Mindestwert 20, Höchstwert 100.

Antwort

- DBSubnetGroups – Ein Array mit DBSubnetGruppe-Objekten.

  Eine Liste der the section called “DBSubnetGruppe”-Instances.
- Marker – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Ein optionales Paginierungstoken, das von einer vorherigen Anforderung bereitgestellt wird. Wenn Sie diesen Parameter angeben, enthält die Antwort nur die Datensätze zwischen der Markierung und dem durch `MaxRecords` angegebenen Wert.

Fehler

- DBSubnetGroupNotFoundFault

Strukturen:

Subnetz (Struktur)

Gibt ein Subnetz an.

Dieser Datentyp wird als Antwortelement in der Aktion the section called “Beschreiben Sie DBSubnet Gruppen” verwendet.

Felder

- SubnetAvailabilityZone – Dies ist ein AvailabilityZone-Objekt.

  Gibt die EC2 Availability Zone an, in der sich das Subnetz befindet.
- SubnetIdentifier – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt die Kennung des Subnetzes an.
- SubnetStatus – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Status des Subnetzes an.

DBSubnetGruppe (Struktur)

Enthält die Details einer Amazon Neptune-DB-Subnetzgruppe.

Dieser Datentyp wird als Antwortelement in der Aktion the section called “Beschreiben Sie DBSubnet Gruppen” verwendet.

Felder

- DBSubnetGroupArn – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Der Amazon-Ressourcenname (ARN) für die DB-Subnetzgruppe.

- DBSubnetGroupDescription – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Stellt die Beschreibung der DB-Subnetzgruppe bereit.
- DBSubnetGroupName – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Name der DB-Subnetzgruppe.
- SubnetGroupStatus – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Stellt den Status der DB-Subnetzgruppe bereit.
- Subnets – Dies ist ein Array von Subnetz-Objekten.

  Enthält eine Liste von the section called “Subnetz”-Elementen.
- VpcId – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Stellt die Nummer VpcId der DB-Subnetzgruppe bereit.

`DBSubnetGroup` wird als Antwortelement verwendet für:

- DBSubnetGruppe erstellen
- DBSubnetGruppe ändern

Neptune Snapshots-API

Aktionen:

- DBClusterSnapshot erstellen (Aktion)
- DBClusterSnapshot löschen (Aktion)
- DBClusterSchnappschuss kopieren (Aktion)
- Ändern DBCluster SnapshotAttribute (Aktion)
- Wiederherstellen DBCluster FromSnapshot (Aktion)
- Wiederherstellen DBCluster ToPointInTime (Aktion)
- Beschreiben Sie DBCluster Schnappschüsse (Aktion)

- Beschreiben Sie DBCluster SnapshotAttributes (Aktion)

Strukturen:

- DBClusterSnapshot (Struktur)
- DBClusterSnapshotAttribute (Struktur)
- DBClusterSnapshotAttributesResult (Struktur)

DBClusterSnapshot erstellen (Aktion)

Der AWS CLI-Name für diese API lautet:`create-db-cluster-snapshot`.

Erstellt einen Snapshot eines DB-Clusters.

Anforderung

- DBClusterIdentifier (in der CLI: `--db-cluster-identifier`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Kennung des DB-Clusters, für den ein Snapshot erstellt werden soll. Bei diesem Parameter wird nicht zwischen Groß- und Kleinschreibung unterschieden.

  Einschränkungen:

  - Muss mit dem Bezeichner einer vorhandenen Datei übereinstimmen DBCluster.

  Beispiel: `my-cluster1`
- DBClusterSnapshotIdentifier (in der CLI: `--db-cluster-snapshot-identifier`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Kennung des DB-Cluster-Snapshots. Dieser Parameter wird als Zeichenfolge in Kleinbuchstaben gespeichert.

  Einschränkungen:

  - Muss zwischen 1 und 63 Buchstaben, Ziffern oder Bindestriche enthalten.
  - Muss mit einem Buchstaben beginnen.
  - Darf nicht mit einem Bindestrich enden oder zwei aufeinanderfolgende Bindestriche enthalten.

  Beispiel: `my-cluster1-snapshot1`

- Tags (in der CLI: `--tags`) – Ein Array von Markierung Objekten.

  Die Tags, die dem DB-Cluster-Snapshot zugewiesen werden.

Antwort

Enthält die Details für einen Amazon Neptune-DB-Cluster-Snapshot.

Dieser Datentyp wird als Antwortelement in der Aktion the section called “Beschreiben Sie Schnappschüsse DBCluster” verwendet.

- AllocatedStorage – eine Ganzzahl vom Typ `integer` (32-Bit-Ganzzahl mit Vorzeichen).

  Legt die Größe des zugewiesenen Speichers in Gibibyte (GiB) fest.
- AvailabilityZones – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Stellt die Liste der EC2 Availability Zones bereit, in denen Instances im DB-Cluster-Snapshot wiederhergestellt werden können.
- ClusterCreateTime— a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Offset von Mitternacht 1970-01-01).

  Gibt den Zeitpunkt an, an dem der DB-Cluster erstellt wurde, in UTC (Universal Coordinated Time).
- DBClusterIdentifier – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Legt die DB-Cluster-Kennung des DB-Clusters fest, anhand dessen dieser DB-Cluster-Snapshot erstellt wurde.
- DBClusterSnapshotArn – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Amazon-Ressourcenname (ARN) für den DB-Cluster-Snapshot.
- DBClusterSnapshotIdentifier – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Legt die Kennung für den DB-Cluster-Snapshot fest. Muss mit der Kennung eines vorhandenen Snapshots übereinstimmen.

  Nachdem Sie einen DB-Cluster mit einem `DBClusterSnapshotIdentifier` wiederhergestellt haben, müssen Sie denselben `DBClusterSnapshotIdentifier` für zukünftige Aktualisierungen des DB-Clusters angeben. Wenn Sie diese Eigenschaft für eine Aktualisierung angeben, wird der DB-Cluster nicht erneut aus dem Snapshot wiederhergestellt und die Daten in der Datenbank werden nicht geändert.

Sollten Sie die `DBClusterSnapshotIdentifier`-Eigenschaft jedoch nicht angeben, wird ein leerer DB-Cluster erstellt und der ursprüngliche DB-Cluster wird gelöscht. Wenn Sie eine Eigenschaft angeben, die sich von der vorherigen Snapshot-Eigenschaft zur Wiederherstellung unterscheidet, wird der DB-Cluster aus dem Snapshot wiederhergestellt, der durch den `DBClusterSnapshotIdentifier` angegeben wurde, und der ursprüngliche DB-Cluster gelöscht.

- Engine – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Legt den Namen der Datenbank-Engine fest.

- EngineVersion – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Stellt die Version der Datenbank-Engine für diesen DB-Cluster-Snapshot bereit.

- IAMDatabaseAuthenticationEnabled – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Dies ist „true“, wenn die Zuweisung von Amazon Identity and Access Management (IAM)-Konten zu Datenbank-Konten aktiviert ist. Andernfalls „false“.

- KmsKeyId – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Wenn `StorageEncrypted` „true“ ist, ist dies die Amazon-KMS-Schlüsselkennung für den verschlüsselten DB-Cluster-Snapshot.

- LicenseModel – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Stellt die Lizenzmodellinformationen für diesen DB-Cluster-Snapshot bereit.

- PercentProgress – eine Ganzzahl vom Typ `integer` (32-Bit-Ganzzahl mit Vorzeichen).

  Gibt einen Prozentsatz der Daten an, die laut Schätzung bereits übertragen wurden.

- Port – eine Ganzzahl vom Typ `integer` (32-Bit-Ganzzahl mit Vorzeichen).

  Gibt den Port an, den der DB-Cluster zum Zeitpunkt des Snapshots überwacht hat.

- SnapshotCreateTime— a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Versatz von Mitternacht am 01.01.1970).

  Gibt das Datum und die Uhrzeit der Erstellung des Snapshots in koordinierter Weltzeit (UTC) an.

- SnapshotType – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Stellt den Typ des DB-Cluster-Snapshots bereit.

- SourceDBClusterSnapshotArn – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Falls der DB-Cluster-Snapshot von einem Quell-DB-Cluster-Snapshot kopiert wurde, der Amazon-Ressourcenname (ARN) für den Quell-DB-Cluster-Snapshot, andernfalls ein Null-Wert.
- Status – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Legt den Status dieses DB-Cluster-Snapshots fest.
- StorageEncrypted – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Gibt an, ob der DB-Cluster-Snapshot verschlüsselt ist.
- StorageType – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Speichertyp, der dem DB-Cluster-Snapshot zugeordnet ist.
- VpcId – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Stellt die VPC-ID bereit, die dem DB-Cluster-Snapshot zugeordnet ist.

Fehler

- DBClusterSnapshotAlreadyExistsFault
- Ungültig DBCluster StateFault
- DBClusterNotFoundFault
- SnapshotQuotaExceededFault
- Ungültig DBCluster SnapshotStateFault

DBClusterSnapshot löschen (Aktion)

Der AWS CLI-Name für diese API lautet:`delete-db-cluster-snapshot`.

Löscht einen DB-Cluster-Snapshot. Wenn der Snapshot gerade kopiert wird, wird der Kopiervorgang beendet.

> **Note**
>
> Der DB-Cluster-Snapshot muss sich im Status `available` befinden, um gelöscht werden zu können.

Anforderung

- DBClusterSnapshotIdentifier (in der CLI: `--db-cluster-snapshot-identifier`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Kennung des DB-Cluster-Snapshots, der gelöscht werden soll.

  Einschränkungen: Es muss der Name eines bestehenden DB-Cluster-Snapshots im Status `available` sein.

Antwort

Enthält die Details für einen Amazon Neptune-DB-Cluster-Snapshot.

Dieser Datentyp wird als Antwortelement in der Aktion the section called “Beschreiben Sie Schnappschüsse DBCluster” verwendet.

- AllocatedStorage – eine Ganzzahl vom Typ `integer` (32-Bit-Ganzzahl mit Vorzeichen).

  Legt die Größe des zugewiesenen Speichers in Gibibyte (GiB) fest.
- AvailabilityZones – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Stellt die Liste der EC2 Availability Zones bereit, in denen Instances im DB-Cluster-Snapshot wiederhergestellt werden können.
- ClusterCreateTime— a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Offset von Mitternacht 1970-01-01).

  Gibt den Zeitpunkt an, an dem der DB-Cluster erstellt wurde, in UTC (Universal Coordinated Time).
- DBClusterIdentifier – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Legt die DB-Cluster-Kennung des DB-Clusters fest, anhand dessen dieser DB-Cluster-Snapshot erstellt wurde.
- DBClusterSnapshotArn – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Amazon-Ressourcenname (ARN) für den DB-Cluster-Snapshot.
- DBClusterSnapshotIdentifier – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Legt die Kennung für den DB-Cluster-Snapshot fest. Muss mit der Kennung eines vorhandenen Snapshots übereinstimmen.

Nachdem Sie einen DB-Cluster mit einem `DBClusterSnapshotIdentifier` wiederhergestellt haben, müssen Sie denselben `DBClusterSnapshotIdentifier` für zukünftige Aktualisierungen des DB-Clusters angeben. Wenn Sie diese Eigenschaft für eine Aktualisierung angeben, wird der DB-Cluster nicht erneut aus dem Snapshot wiederhergestellt und die Daten in der Datenbank werden nicht geändert.

Sollten Sie die `DBClusterSnapshotIdentifier`-Eigenschaft jedoch nicht angeben, wird ein leerer DB-Cluster erstellt und der ursprüngliche DB-Cluster wird gelöscht. Wenn Sie eine Eigenschaft angeben, die sich von der vorherigen Snapshot-Eigenschaft zur Wiederherstellung unterscheidet, wird der DB-Cluster aus dem Snapshot wiederhergestellt, der durch den `DBClusterSnapshotIdentifier` angegeben wurde, und der ursprüngliche DB-Cluster gelöscht.

- Engine – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Legt den Namen der Datenbank-Engine fest.

- EngineVersion – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Stellt die Version der Datenbank-Engine für diesen DB-Cluster-Snapshot bereit.

- IAMDatabaseAuthenticationEnabled – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Dies ist „true“, wenn die Zuweisung von Amazon Identity and Access Management (IAM)-Konten zu Datenbank-Konten aktiviert ist. Andernfalls „false“.

- KmsKeyId – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Wenn `StorageEncrypted` „true“ ist, ist dies die Amazon-KMS-Schlüsselkennung für den verschlüsselten DB-Cluster-Snapshot.

- LicenseModel – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Stellt die Lizenzmodellinformationen für diesen DB-Cluster-Snapshot bereit.

- PercentProgress – eine Ganzzahl vom Typ `integer` (32-Bit-Ganzzahl mit Vorzeichen).

  Gibt einen Prozentsatz der Daten an, die laut Schätzung bereits übertragen wurden.

- Port – eine Ganzzahl vom Typ `integer` (32-Bit-Ganzzahl mit Vorzeichen).

  Gibt den Port an, den der DB-Cluster zum Zeitpunkt des Snapshots überwacht hat.

- SnapshotCreateTime— a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Versatz von Mitternacht am 01.01.1970).

  Gibt das Datum und die Uhrzeit der Erstellung des Snapshots in koordinierter Weltzeit (UTC) an.
- SnapshotType – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Stellt den Typ des DB-Cluster-Snapshots bereit.
- SourceDBClusterSnapshotArn – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Falls der DB-Cluster-Snapshot von einem Quell-DB-Cluster-Snapshot kopiert wurde, der Amazon-Ressourcenname (ARN) für den Quell-DB-Cluster-Snapshot, andernfalls ein Null-Wert.
- Status – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Legt den Status dieses DB-Cluster-Snapshots fest.
- StorageEncrypted – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Gibt an, ob der DB-Cluster-Snapshot verschlüsselt ist.
- StorageType – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Speichertyp, der dem DB-Cluster-Snapshot zugeordnet ist.
- VpcId – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Stellt die VPC-ID bereit, die dem DB-Cluster-Snapshot zugeordnet ist.

Fehler

- Ungültig DBCluster SnapshotStateFault
- DBClusterSnapshotNotFoundFault

DBClusterSchnappschuss kopieren (Aktion)

> Der AWS CLI-Name für diese API lautet:`copy-db-cluster-snapshot`.

Kopiert einen Snapshot eines DB-Clusters.

Um einen DB-Cluster-Snapshot von einem freigegebenen manuellen DB-Cluster-Snapshot zu kopieren, muss `SourceDBClusterSnapshotIdentifier` der Amazon-Ressourcenname (ARN) des freigegebenen DB-Snapshots sein.

Anforderung

- CopyTags(in der CLI:`--copy-tags`) — a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  "True", um alle Tags aus dem Quell-DB-Cluster-Snapshot in den Ziel-DB-Cluster-Snapshot zu kopieren, andernfalls "false". Der Standardwert lautet „false“.
- KmsKeyId (in der CLI: `--kms-key-id`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Schlüssel-ID von Amazon KMS für einen verschlüsselten DB-Snapshot. Die KMS-Schlüssel-ID ist der Amazon-Ressourcenname (ARN), der KMS-Schlüsselbezeichner oder der KMS-Schlüsselalias für den KMS-Verschlüsselungsschlüssel.

  Wenn Sie einen verschlüsselten DB-Cluster-Snapshot aus Ihrem Amazon-Konto kopieren, können Sie für `KmsKeyId` einen Wert angeben, damit die Kopie mit einem neuen KMS-Verschlüsselungsschlüssel verschlüsselt wird. Wenn Sie keinen Wert für `KmsKeyId` angeben, wird die Kopie des DB-Cluster-Snapshots mit demselben KMS-Schlüssel wie der Quell-DB-Cluster-Snapshot verschlüsselt.

  Wenn Sie einen verschlüsselten DB-Cluster-Snapshot kopieren, der von einem anderen Amazon-Konto freigegeben wurde, müssen Sie einen Wert für `KmsKeyId` angeben.

  KMS-Schlüssel sind spezifisch für die Region, in der sie erstellt wurden, und Sie können keine Verschlüsselungsschlüssel aus einer Amazon-Region in einer anderen AmazonRegion verwenden.

  Sie können einen unverschlüsselten DB-Cluster-Snapshot nicht verschlüsseln, wenn Sie ihn kopieren. Wenn Sie versuchen, einen unverschlüsselten DB-Cluster-Snapshot zu kopieren und einen Wert für den KmsKeyId Parameter anzugeben, wird ein Fehler zurückgegeben.
- PreSignedUrl (in der CLI: `--pre-signed-url`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Wird derzeit nicht unterstützt.
- SourceDBClusterSnapshotIdentifier (in der CLI: `--source-db-cluster-snapshot-identifier`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Die Kennung des zu kopierenden DB-Cluster-Snapshots. Bei diesem Parameter wird nicht zwischen Groß- und Kleinschreibung unterschieden.

Einschränkungen:

- Es ist ein gültiger System-Snapshot im Status "available" erforderlich.
- Geben Sie eine gültige DB-Snapshot-Kennung an.

Beispiel: `my-cluster-snapshot1`

- Tags (in der CLI: `--tags`) – Ein Array von Markierung Objekten.

  Die Tags für die Zuweisung zur neuen DB-Cluster-Snapshot-Kopie.
- TargetDBClusterSnapshotIdentifier (in der CLI: `--target-db-cluster-snapshot-identifier`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Kennung des neuen DB-Cluster-Snapshots, der aus dem Quell-DB-Cluster-Snapshot erstellt werden soll. Bei diesem Parameter wird nicht zwischen Groß- und Kleinschreibung unterschieden.

  Einschränkungen:
  - Muss zwischen 1 und 63 Buchstaben, Ziffern oder Bindestriche enthalten.
  - Muss mit einem Buchstaben beginnen.
  - Darf nicht mit einem Bindestrich enden oder zwei aufeinanderfolgende Bindestriche enthalten.

  Beispiel: `my-cluster-snapshot2`

Antwort

Enthält die Details für einen Amazon Neptune-DB-Cluster-Snapshot.

Dieser Datentyp wird als Antwortelement in der Aktion the section called “Beschreiben Sie Schnappschüsse DBCluster” verwendet.

- AllocatedStorage – eine Ganzzahl vom Typ `integer` (32-Bit-Ganzzahl mit Vorzeichen).

  Legt die Größe des zugewiesenen Speichers in Gibibyte (GiB) fest.
- AvailabilityZones – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Stellt die Liste der EC2 Availability Zones bereit, in denen Instances im DB-Cluster-Snapshot wiederhergestellt werden können.

- ClusterCreateTime— a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Offset von Mitternacht 1970-01-01).

  Gibt den Zeitpunkt an, an dem der DB-Cluster erstellt wurde, in UTC (Universal Coordinated Time).

- DBClusterIdentifier – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Legt die DB-Cluster-Kennung des DB-Clusters fest, anhand dessen dieser DB-Cluster-Snapshot erstellt wurde.

- DBClusterSnapshotArn – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Amazon-Ressourcenname (ARN) für den DB-Cluster-Snapshot.

- DBClusterSnapshotIdentifier – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Legt die Kennung für den DB-Cluster-Snapshot fest. Muss mit der Kennung eines vorhandenen Snapshots übereinstimmen.

  Nachdem Sie einen DB-Cluster mit einem `DBClusterSnapshotIdentifier` wiederhergestellt haben, müssen Sie denselben `DBClusterSnapshotIdentifier` für zukünftige Aktualisierungen des DB-Clusters angeben. Wenn Sie diese Eigenschaft für eine Aktualisierung angeben, wird der DB-Cluster nicht erneut aus dem Snapshot wiederhergestellt und die Daten in der Datenbank werden nicht geändert.

  Sollten Sie die `DBClusterSnapshotIdentifier`-Eigenschaft jedoch nicht angeben, wird ein leerer DB-Cluster erstellt und der ursprüngliche DB-Cluster wird gelöscht. Wenn Sie eine Eigenschaft angeben, die sich von der vorherigen Snapshot-Eigenschaft zur Wiederherstellung unterscheidet, wird der DB-Cluster aus dem Snapshot wiederhergestellt, der durch den `DBClusterSnapshotIdentifier` angegeben wurde, und der ursprüngliche DB-Cluster gelöscht.

- Engine – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Legt den Namen der Datenbank-Engine fest.

- EngineVersion – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Stellt die Version der Datenbank-Engine für diesen DB-Cluster-Snapshot bereit.

- IAMDatabaseAuthenticationEnabled – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

Dies ist „true“, wenn die Zuweisung von Amazon Identity and Access Management (IAM)-Konten zu Datenbank-Konten aktiviert ist. Andernfalls „false“.

- KmsKeyId – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Wenn `StorageEncrypted` „true“ ist, ist dies die Amazon-KMS-Schlüsselkennung für den verschlüsselten DB-Cluster-Snapshot.
- LicenseModel – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Stellt die Lizenzmodellinformationen für diesen DB-Cluster-Snapshot bereit.
- PercentProgress – eine Ganzzahl vom Typ `integer` (32-Bit-Ganzzahl mit Vorzeichen).

  Gibt einen Prozentsatz der Daten an, die laut Schätzung bereits übertragen wurden.
- Port – eine Ganzzahl vom Typ `integer` (32-Bit-Ganzzahl mit Vorzeichen).

  Gibt den Port an, den der DB-Cluster zum Zeitpunkt des Snapshots überwacht hat.
- SnapshotCreateTime— a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Versatz von Mitternacht am 01.01.1970).

  Gibt das Datum und die Uhrzeit der Erstellung des Snapshots in koordinierter Weltzeit (UTC) an.
- SnapshotType – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Stellt den Typ des DB-Cluster-Snapshots bereit.
- SourceDBClusterSnapshotArn – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Falls der DB-Cluster-Snapshot von einem Quell-DB-Cluster-Snapshot kopiert wurde, der Amazon-Ressourcenname (ARN) für den Quell-DB-Cluster-Snapshot, andernfalls ein Null-Wert.
- Status – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Legt den Status dieses DB-Cluster-Snapshots fest.
- StorageEncrypted – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Gibt an, ob der DB-Cluster-Snapshot verschlüsselt ist.
- StorageType – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Speichertyp, der dem DB-Cluster-Snapshot zugeordnet ist.
- VpcId – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Stellt die VPC-ID bereit, die dem DB-Cluster-Snapshot zugeordnet ist.

Fehler

- DBClusterSnapshotAlreadyExistsFault
- DBClusterSnapshotNotFoundFault
- Ungültig DBCluster StateFault
- Ungültig DBCluster SnapshotStateFault
- SnapshotQuotaExceededFault
- KMSKeyNotAccessibleFault

Ändern DBCluster SnapshotAttribute (Aktion)

Der AWS CLI-Name für diese API lautet:`modify-db-cluster-snapshot-attribute`.

Fügt ein Attribut und Werte zu einem manuellen DB-Cluster-Snapshot hinzu oder entfernt ein Attribut und Werte daraus.

Um einen manuellen DB-Cluster-Snapshot mit anderen Amazon-Konten zu teilen, geben Sie `restore` als an `AttributeName` und verwenden Sie den `ValuesToAdd` Parameter, um eine Liste IDs der Amazon-Konten hinzuzufügen, die berechtigt sind, den manuellen DB-Cluster-Snapshot wiederherzustellen. Verwenden Sie den Wert `all`, um den manuellen DB-Cluster-Snapshot zu veröffentlichen. Dies bedeutet, dass er von allen Amazon-Konten kopiert oder wiederhergestellt werden kann. Fügen Sie den Wert `all` zu keinem manuellen DB-Cluster-Snapshot hinzu, der private Informationen enthält, die nicht für alle Amazon-Konten verfügbar sein sollen. Wenn ein manueller DB-Cluster-Snapshot verschlüsselt ist, kann er geteilt werden, aber nur, indem eine Liste autorisierter Amazon-Konten IDs für den `ValuesToAdd` Parameter angegeben wird. Sie können in diesem Fall `all` nicht als Wert für diesen Parameter verwenden.

Um anzuzeigen, welche Amazon-Konten Zugriff für das Kopieren oder Wiederherstellen eines manuellen DB-Cluster-Snapshots haben, oder ob ein manueller DB-Cluster-Snapshot öffentlich oder privat ist, verwenden Sie die API-Aktion the section called “Beschreiben DBCluster SnapshotAttributes”.

Anforderung

- AttributeName (in der CLI: `--attribute-name`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Name des zu ändernden DB-Cluster-Snapshot-Attributs.

  Um die Autorisierung für andere Amazon-Konten zu verwalten, um einen manuellen DB-Cluster-Snapshot zu kopieren oder wiederherzustellen, setzen Sie diesen Wert auf `restore`.
- DBClusterSnapshotIdentifier (in der CLI: `--db-cluster-snapshot-identifier`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Kennung für den DB-Cluster-Snapshot, für den die Attribute geändert werden sollen.
- ValuesToAdd (in der CLI: `--values-to-add`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Liste von DB-Cluster-Snapshot-Attributen zum Hinzufügen zu dem Attribut, das durch `AttributeName` festgelegt ist.

  Um andere Amazon-Konten zum Kopieren oder Wiederherstellen eines manuellen DB-Cluster-Snapshots zu autorisieren, legen Sie diese Liste so fest, dass sie ein oder mehrere Amazon-Konten enthält oder `all` dass der manuelle DB-Cluster-Snapshot von einem beliebigen Amazon-Konto IDs wiederhergestellt werden kann. Fügen Sie den Wert `all` zu keinem manuellen DB-Cluster-Snapshot hinzu, der private Informationen enthält, die nicht für alle Amazon-Konten verfügbar sein sollen.
- ValuesToRemove (in der CLI: `--values-to-remove`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Liste von DB-Cluster-Snapshot-Attributen zum Entfernen von dem Attribut, das durch `AttributeName` festgelegt ist.

  Um die Autorisierung anderer Amazon-Konten zum Kopieren oder Wiederherstellen eines manuellen DB-Cluster-Snapshots zu entfernen, legen Sie diese Liste so fest, dass sie eine oder mehrere Amazon-Konto-Kennungen enthält, oder wählen Sie `all`, um die Autorisierung zum Kopieren oder Wiederherstellen des manuellen DB-Cluster-Snapshots von jedem Amazon-Konto zu entfernen. Wenn Sie `all` festlegen, kann ein Amazon-Konto, dessen Konto-ID explizit zum Attribut `restore` hinzugefügt wird, noch immer einen manuellen DB-Cluster-Snapshot kopieren oder wiederherstellen.

Antwort

Enthält die Ergebnisse eines erfolgreichen Aufrufs an die API-Aktion the section called “Beschreiben DBCluster SnapshotAttributes”.

Manuelle DB-Cluster-Snapshot-Attribute werden verwendet, um andere Amazon-Konten zu autorisieren und einen manuellen DB-Cluster-Snapshot zu kopieren oder wiederherzustellen. Weitere Informationen finden Sie in der API-Aktion the section called “Modifizieren DBCluster SnapshotAttribute”.

- DBClusterSnapshotAttributes – Ein Array mit DBClusterSnapshotAttribute-Objekten.

  Die Liste der Attribute und Werte für den manuellen DB-Cluster-Snapshot.
- DBClusterSnapshotIdentifier – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Kennung des manuellen DB-Cluster-Snapshots, für den die Attribute gelten.

Fehler

- DBClusterSnapshotNotFoundFault
- Ungültig DBCluster SnapshotStateFault
- SharedSnapshotQuotaExceededFault

Wiederherstellen DBCluster FromSnapshot (Aktion)

Der AWS CLI-Name für diese API lautet:`restore-db-cluster-from-snapshot`.

Erstellt einen neuen DB-Cluster aus einem DB-Snapshot oder einem DB-Cluster-Snapshot.

Wenn ein DB-Snapshot angegeben ist, wird der Ziel-DB-Cluster mit einer Standard-Konfiguration und Standard-Sicherheitsgruppe aus dem Quell-DB-Snapshot erstellt.

Wenn ein DB-Cluster-Snapshot angegeben ist, wird der Ziel-DB-Cluster vom Quell-DB-Cluster-Wiederherstellungspunkt mit der gleichen Konfiguration wie der ursprüngliche Quell-DB-Cluster erstellt, mit der Ausnahme, dass der neue DB-Cluster mit der Standard-Sicherheitsgruppe erstellt wird.

Anforderung

- AvailabilityZones (in der CLI: `--availability-zones`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Stellt die Liste der EC2 Availability Zones bereit, in denen Instances im wiederhergestellten DB-Cluster erstellt werden können.
- CopyTagsToSnapshot(in der CLI:`--copy-tags-to-snapshot`) — a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Wenn auf *`true`* gesetzt, werden Tags in jeden Snapshot des wiederhergestellten DB-Clusters kopiert, der erstellt wird.
- DatabaseName (in der CLI: `--database-name`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Nicht unterstützt
- DBClusterIdentifier (in der CLI: `--db-cluster-identifier`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Name des DB-Clusters, der aus dem DB-Snapshot oder dem DB-Cluster-Snapshot erstellt werden soll. Bei diesem Parameter wird nicht zwischen Groß- und Kleinschreibung unterschieden.

  Einschränkungen:
  - Muss zwischen 1 und 63 Buchstaben, Ziffern oder Bindestriche enthalten.
  - Muss mit einem Buchstaben beginnen
  - Darf nicht mit einem Bindestrich enden oder zwei aufeinanderfolgende Bindestriche enthalten

  Beispiel: `my-snapshot-id`
- DBClusterParameterGroupName (in der CLI: `--db-cluster-parameter-group-name`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Name der DB-Cluster-Parametergruppe, die mit dem neuen DB-Cluster verknüpft werden soll.

  Einschränkungen:
  - Falls angegeben, muss er mit dem Namen eines vorhandenen übereinstimmen. DBCluster ParameterGroup
- DBSubnetGroupName (in der CLI: `--db-subnet-group-name`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Name der DB-Subnetz-Gruppe, die für den neuen DB-Cluster verwendet werden soll.

Einschränkungen: Falls angegeben, muss es mit dem Namen einer vorhandenen DBSubnet Gruppe übereinstimmen.

Beispiel: `mySubnetgroup`

- DeletionProtection(in der CLI:`--deletion-protection`) — a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Ein Wert, der angibt, ob der Löschschutz des DB-Clusters aktiviert ist. Die Datenbank kann nicht gelöscht werden, wenn der Löschschutz aktiviert ist. Der Löschschutz ist standardmäßig deaktiviert.

- EnableCloudwatchLogsExports (in der CLI: `--enable-cloudwatch-logs-exports`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Liste der Protokolle, die der wiederhergestellte DB-Cluster nach Amazon CloudWatch Logs exportieren soll.

- EnableIAMDatabaseAuthentication(in der CLI:`--enable-iam-database-authentication`) — a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  „True“, um die Zuweisung von Amazon Identity and Access Management (IAM)-Konten zu Datenbankkonten zu aktivieren; andernfalls „false“.

  Standard: `false`

- Engine (in der CLI: `--engine`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Datenbank-Engine, die für den neuen DB-Cluster verwendet werden soll.

  Standardmäßig: Gleich wie Quelle

  Einschränkung: Muss mit der Engine der Quelle kompatibel sein

- EngineVersion (in der CLI: `--engine-version`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Version der Datenbank-Engine, die für den neuen DB-Cluster verwendet werden soll.

- KmsKeyId (in der CLI: `--kms-key-id`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Die Amazon KMS-Schlüsselkennung, die bei der Wiederherstellung eines verschlüsselten DB-Clusters aus einem DB-Snapshot oder DB-Cluster-Snapshot verwendet werden soll.

Die Kennung für den KMS-Schlüssel ist der Amazon-Ressourcenname (ARN) für den KMS-Verschlüsselungsschlüssel. Wenn Sie ein DB-Cluster mit demselben Amazon-Konto wiederherstellen, das über den KMS-Verschlüsselungsschlüssel verfügt, der für die Verschlüsselung des neuen DB-Clusters verwendet wurde, können Sie den KMS-Schlüssel-Alias anstelle des ARNs für den KMS-Verschlüsselungsschlüssel verwenden.

Wenn Sie keinen Wert für den Parameter `KmsKeyId` angeben, geschieht folgendes:

- Wenn der DB-Snapshot oder DB-Cluster-Snapshot in `SnapshotIdentifier` verschlüsselt ist, wird der wiederhergestellte DB-Cluster anhand des KMS-Schlüssels verschlüsselt, der zum Verschlüsseln des DB-Snapshots oder DB-Cluster-Snapshots verwendet wurde.
- Wenn der DB-Snapshot oder DB-Cluster Snapshot in `SnapshotIdentifier` nicht verschlüsselt ist, ist der wiederhergestellte DB-Cluster nicht verschlüsselt.

- Port(in der CLI:`--port`) — ein IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Die Nummer des Ports, an dem der neue DB-Cluster Verbindungen akzeptiert.

  Einschränkungen: Der Wert muss `1150-65535` lauten.

  Standard: Der gleiche Port wie der ursprüngliche DB-Cluster.

- ServerlessV2ScalingConfiguration (in der CLI: `--serverless-v2-scaling-configuration`) – Ein Serverless V2 ScalingConfiguration-Objekt.

  Enthält die Skalierungskonfiguration eines Neptune Serverless DB-Clusters.

  Weitere Informationen finden Sie unter Verwendung von Amazon Neptune Serverless im Amazon Neptune-Benutzerhandbuch.

- SnapshotIdentifier (in der CLI: `--snapshot-identifier`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Kennung für den DB-Snapshot oder den DB-Cluster-Snapshot, der zur Wiederherstellung verwendet werden soll.

Sie können entweder den Namen oder den Amazon-Ressourcennamen (ARN) verwenden, um einen DB-Cluster-Snapshot festzulegen. Sie können jedoch nur den ARN verwenden, um einen DB-Snapshot festzulegen.

Einschränkungen:

- Muss mit der Kennung eines vorhandenen Snapshots übereinstimmen.

- StorageType (in der CLI: `--storage-type`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Speichertyp an, der dem DB-Cluster zugeordnet werden soll.

  Zulässige Werte: `standard`, `iopt1`

  Standard: `standard`

- Tags (in der CLI: `--tags`) – Ein Array von Markierung Objekten.

  Die Tags, die dem wiederhergestellten DB-Cluster zugewiesen werden sollen.

- VpcSecurityGroupIds (in der CLI: `--vpc-security-group-ids`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Liste der VPC-Sicherheitsgruppen, zu denen der neue DB-Cluster gehören wird.

Antwort

Enthält die Details eines Amazon Neptune-DB-Clusters.

Dieser Datentyp wird als Antwortelement in der the section called “Beschreiben DBClusters” verwendet.

- AllocatedStorage— ein IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  `AllocatedStorage` gibt immer 1 zurück, da die Neptune-DB-Cluster-Speichergröße nicht fest ist, sondern sich bei Bedarf automatisch anpasst.

- AssociatedRoles – Ein Array mit DBClusterRolle-Objekten.

  Bietet eine Liste der Amazon Identity and Access Management (IAM)-Rollen, die dem DB-Cluster zugeordnet sind. IAM-Rollen, die einem DB-Cluster zugeordnet sind, erteilen dem DB-Cluster die Berechtigung, auf andere Amazon-Services in Ihrem Namen zuzugreifen.

- AutomaticRestartTime— a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Offset von Mitternacht am 01.01.1970).

  Zeitpunkt, zu dem der DB-Cluster automatisch neu gestartet wird.
- AvailabilityZones – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Stellt die Liste der EC2 Availability Zones bereit, in denen Instances im DB-Cluster erstellt werden können.
- BacktrackConsumedChangeRecords— a LongOptional, vom Typ: `long` (eine 64-Bit-Ganzzahl mit Vorzeichen).

  Wird von Neptune nicht unterstützt.
- BacktrackWindow— a LongOptional, vom Typ: `long` (eine 64-Bit-Ganzzahl mit Vorzeichen).

  Wird von Neptune nicht unterstützt.
- BackupRetentionPeriod— an IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Gibt die Anzahl der Tage an, für die automatische DB-Snapshots aufbewahrt werden.
- Capacity— an IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Wird von Neptune nicht unterstützt.
- CloneGroupId – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Identifiziert die Clone-Gruppe, mit der der DB-Cluster verknüpft ist.
- ClusterCreateTime— a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Offset von Mitternacht am 01.01.1970).

  Gibt den Zeitpunkt an, an dem der DB-Cluster erstellt wurde, in UTC (Universal Coordinated Time).
- CopyTagsToSnapshot— a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Wenn auf *`true`* gesetzt, werden Tags in jeden Snapshot des DB-Clusters kopiert, der erstellt wird.
- CrossAccountClone— a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Wenn auf `true` gesetzt, kann der DB-Cluster kontenübergreifend geklont werden.
- DatabaseName – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Enthält den Namen der initialen Datenbank dieses DB-Clusters, die zum Erstellungszeitpunkt bereitgestellt wurde, sofern eine angegeben wurde, als der DB-Cluster erstellt wurde. Derselbe Name wird über die Lebensdauer des DB-Clusters zurückgegeben.

- DBClusterArn – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Amazon-Ressourcenname (ARN) für den DB-Cluster.

- DBClusterIdentifier – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Enthält eine vom Benutzer bereitgestellte DB-Cluster-Kennung. Diese Kennung ist der eindeutige Schlüssel zur Identifizierung eines DB-Clusters.

- DBClusterMembers – Ein Array mit DBClusterMitglied-Objekten.

  Enthält eine Liste der Instances, aus denen der DB-Cluster besteht.

- DBClusterParameterGroup – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Namen der DB-Cluster-Parametergruppe für den DB-Cluster an.

- DbClusterResourceId – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die für die Amazon-Region eindeutige, unveränderliche Kennung für den DB-Cluster. Diese Kennung wird in CloudTrail Amazon-Protokolleinträgen immer dann gefunden, wenn auf den Amazon KMS-Schlüssel für den DB-Cluster zugegriffen wird.

- DBSubnetGroup – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt Informationen zu der Subnetzgruppe an, die dem DB-Cluster zugeordnet ist, einschließlich dem Namen, der Beschreibung und der Subnetze in der Subnetzgruppe.

- DeletionProtection— a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Gibt an, ob der Löschschutz des DB-Clusters aktiviert ist. Die Datenbank kann nicht gelöscht werden, wenn der Löschschutz aktiviert ist.

- EarliestBacktrackTime— a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Offset von Mitternacht 1970-01-01).

  Wird von Neptune nicht unterstützt.

- EarliestRestorableTime— a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Versatz von Mitternacht am 01.01.1970).

Gibt den frühesten Zeitpunkt an, zu dem eine Datenbank mit point-in-time restore wiederhergestellt werden kann.

- EnabledCloudwatchLogsExports – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Liste der Protokolltypen, für deren Export dieser DB-Cluster in CloudWatch Logs konfiguriert ist. Gültige Protokolltypen sind: `audit` (zum Veröffentlichen von Audit-Logs CloudWatch) und slowquery (zum Veröffentlichen von Slow-Query-Logs). CloudWatch Weitere Informationen finden Sie unter Neptune-Logs in CloudWatch Amazon-Protokollen veröffentlichen.

- Endpoint – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Verbindungsendpunkt für die primäre Instance des DB-Clusters an.

- Engine – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Namen der Datenbank-Engine an, die für diesen DB-Cluster verwendet werden soll.

- EngineVersion – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt die Version der Datenbank-Engine an.

- GlobalClusterIdentifier— a GlobalClusterIdentifier, vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge), nicht weniger als 1 oder mehr als 255? st? s, entspricht diesem regulären Ausdruck:`[A-Za-z][0-9A-Za-z-:._]*`.

  Enthält eine vom Benutzer bereitgestellte gelobale Datenbank-Cluster-ID. Diese Kennung ist der eindeutige Schlüssel zur Identifizierung einer globalen Datenbank.

- HostedZoneId – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt die ID an, die Amazon Route 53 zuweist, wenn Sie eine gehostete Zone erstellen.

- IAMDatabaseAuthenticationEnabled – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Dies ist „true“, wenn die Zuweisung von Amazon Identity and Access Management (IAM)-Konten zu Datenbank-Konten aktiviert ist. Andernfalls „false“.

- IOOptimizedNextAllowedModificationTime— a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Offset von Mitternacht 1970-01-01).

  Beim nächsten Mal können Sie den DB-Cluster so ändern, dass der Speichertyp `iopt1` verwendet wird.

- KmsKeyId – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Wenn `StorageEncrypted` „true“ ist, ist dies die Amazon-KMS-Schlüsselkennung für das verschlüsselte DB-Cluster.
- LatestRestorableTime— a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Versatz von Mitternacht am 01.01.1970).

  Gibt den letzten Zeitpunkt an, zu dem eine Datenbank mit point-in-time restore wiederhergestellt werden kann.
- MultiAZ – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Gibt an, ob der DB-Cluster über Instances in mehreren Availability Zones verfügt.
- PendingModifiedValues – Ein ClusterPendingModifiedValues-Objekt.

  Dieser Datentyp wird als Antwortelement in der `ModifyDBCluster` Operation verwendet und enthält Änderungen, die während des nächsten Wartungsfensters angewendet werden.
- PercentProgress – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Fortschritt der Operation als Prozentsatz an.
- Port— ein IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Gibt die Portnummer an, die von der Datenbank-Engine überwacht wird.
- PreferredBackupWindow – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den täglichen Zeitraum in koordinierter Weltzeit (UTC) an, in dem automatische Sicherungen erstellt werden, wenn automatische Sicherungen aktiviert sind, gemäß `BackupRetentionPeriod`.
- PreferredMaintenanceWindow – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den wöchentlichen Zeitraum, in dem Systemwartungen durchgeführt werden können, in UTC (Universal Coordinated Time) an.
- ReaderEndpoint – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Leser-Endpunkt für den DB-Cluster. Der Reader-Endpunkt für ein DB-Cluster gleicht die Verbindungslasten zwischen den Read Replicas aus, die in einem DB-Cluster verfügbar sind. Während Clients neue Verbindungsanfragen an den Reader-Endpunkt tätigen, verteilt Neptune die Verbindungsanfragen zwischen den Read Replicas im DB-Cluster. Diese Funktionalität sorgt dafür,

dass die Workload-Auslastung für Lesevorgänge auf mehrere Read Replicas im DB-Cluster verteilt wird.

Wenn ein Failover auftritt und das Read Replica, mit dem Sie verbunden sind, als primäre Instance hochgestuft wird, wird Ihre Verbindung unterbrochen. Verbinden Sie sich erneut mit dem Reader-Endpunkt, um mit dem Senden Ihres Workloads für Lesevorgänge an andere Read Replicas im Cluster fortzufahren.

- ReadReplicaIdentifiers – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Enthält eine oder mehrere Kennungen der mit diesem DB-Cluster verbundenen Read Replicas.
- ReplicationSourceIdentifier – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Wird von Neptune nicht unterstützt.
- ReplicationType – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Wird von Neptune nicht unterstützt.
- ServerlessV2ScalingConfiguration – Ein Serverless V2 ScalingConfigurationInfo-Objekt.

  Zeigt die Skalierungskonfiguration für einen Neptune Serverless DB-Cluster.

  Weitere Informationen finden Sie unter Verwendung von Amazon Neptune Serverless im Amazon Neptune-Benutzerhandbuch.
- Status – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den aktuellen Status dieses DB-Clusters an.
- StorageEncrypted – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Gibt an, ob der DB-Cluster verschlüsselt ist.
- StorageType – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Speichertyp, der vom DB-Cluster verwendet wird.

  Zulässige Werte:

  - **`standard`**— (Standard) Bietet kostengünstigen Datenbankspeicher für Anwendungen mit mäßiger bis geringer I/O Nutzung.
  - **`iopt1`**— Ermöglicht I/O-optimierten Speicher, der auf die Durchsatzanforderungen zugeschnitten ist. I/O-intensive graph workloads that require predictable pricing with low I/O latency and consistent I/O

E/A-optimierter Neptune-Speicher ist erst ab Engine-Version 1.3.0.0 verfügbar.

- VpcSecurityGroups – Ein Array mit VpcSecurityGroupMembership-Objekten.

  Stellt eine Liste der VPC-Sicherheitsgruppen zur Verfügung, zu denen das DB-Cluster gehört.

Fehler

- DBClusterAlreadyExistsFault
- DBClusterQuotaExceededFault
- StorageQuotaExceededFault
- DBSubnetGroupNotFoundFault
- DBSnapshotNotFoundFault
- DBClusterSnapshotNotFoundFault
- Unzureichend DBCluster CapacityFault
- InsufficientStorageClusterCapacityFault
- Ungültig DBSnapshot StateFault
- Ungültig DBCluster SnapshotStateFault
- StorageQuotaExceededFault
- Ungültig VPCNetwork StateFault
- InvalidRestoreFault
- DBSubnetGroupNotFoundFault
- InvalidSubnet
- OptionGroupNotFoundFault
- KMSKeyNotAccessibleFault
- DBClusterParameterGroupNotFoundFault

Wiederherstellen DBCluster ToPointInTime (Aktion)

Der AWS CLI-Name für diese API lautet:`restore-db-cluster-to-point-in-time`.

Stellt einen DB-Cluster im Zustand eines beliebigen Zeitpunkts wieder her. Benutzer können den Zustand jedes beliebigen Zeitpunkts vor `LatestRestorableTime` bis zu `BackupRetentionPeriod` Tagen wiederherstellen. Der Ziel-DB-Cluster wird vom Quell-DB-Cluster

mit der gleichen Konfiguration wie der ursprüngliche DB-Cluster erstellt, mit der Ausnahme, dass der neue DB-Cluster mit der Standard-DB-Sicherheitsgruppe erstellt wird.

> **Note**
>
> Diese Aktion stellt nur den DB-Cluster wieder her, nicht die DB-Instances für diesen DB-Cluster. Sie müssen die Aktion the section called “Erstellen DBInstance” aufrufen, um DB-Instances für den wiederhergestellten DB-Cluster zu erstellen, wobei Sie in `DBClusterIdentifier` die Kennung des wiederhergestellten DB-Clusters angeben. Sie können DB-Instances erst erstellen, nachdem die Aktion `RestoreDBClusterToPointInTime` abgeschlossen ist und der DB-Cluster verfügbar ist.

Anforderung

- DBClusterIdentifier (in der CLI: `--db-cluster-identifier`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Name des neuen DB-Clusters, der erstellt werden soll.

  Einschränkungen:

  - Muss zwischen 1 und 63 Buchstaben, Ziffern oder Bindestriche enthalten.
  - Muss mit einem Buchstaben beginnen
  - Darf nicht mit einem Bindestrich enden oder zwei aufeinanderfolgende Bindestriche enthalten
- DBClusterParameterGroupName (in der CLI: `--db-cluster-parameter-group-name`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Name der DB-Cluster-Parametergruppe, die mit dem neuen DB-Cluster verknüpft werden soll.

  Einschränkungen:

  - Falls angegeben, muss er mit dem Namen einer vorhandenen Datei übereinstimmen DBClusterParameterGroup.
- DBSubnetGroupName (in der CLI: `--db-subnet-group-name`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Name der DB-Subnetzgruppe, die für den neuen DB-Cluster verwendet werden soll.

Einschränkungen: Falls angegeben, muss es mit dem Namen einer vorhandenen DBSubnet Gruppe übereinstimmen.

Beispiel: `mySubnetgroup`

- DeletionProtection(in der CLI:`--deletion-protection`) — a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Ein Wert, der angibt, ob der Löschschutz des DB-Clusters aktiviert ist. Die Datenbank kann nicht gelöscht werden, wenn der Löschschutz aktiviert ist. Der Löschschutz ist standardmäßig deaktiviert.
- EnableCloudwatchLogsExports (in der CLI: `--enable-cloudwatch-logs-exports`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Liste der Protokolle, die der wiederhergestellte DB-Cluster in Logs exportieren soll CloudWatch.
- EnableIAMDatabaseAuthentication(in der CLI:`--enable-iam-database-authentication`) — a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  „True“, um die Zuweisung von Amazon Identity and Access Management (IAM)-Konten zu Datenbankkonten zu aktivieren; andernfalls „false“.

  Standard: `false`
- KmsKeyId (in der CLI: `--kms-key-id`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Amazon KMS-Schlüsselkennung, die bei der Wiederherstellung eines verschlüsselten DB-Clusters aus einem verschlüsselten DB-Cluster verwendet werden soll.

  Die Kennung für den KMS-Schlüssel ist der Amazon-Ressourcenname (ARN) für den KMS-Verschlüsselungsschlüssel. Wenn Sie ein DB-Cluster mit demselben Amazon-Konto wiederherstellen, das über den KMS-Verschlüsselungsschlüssel verfügt, der für die Verschlüsselung des neuen DB-Clusters verwendet wurde, können Sie den KMS-Schlüssel-Alias anstelle des ARNs für den KMS-Verschlüsselungsschlüssel verwenden.

  Sie können auf einen neuen DB-Cluster wiederherstellen und den neuen DB-Cluster mit einem KMS-Schlüssel verschlüsseln, der sich von dem KMS-Schlüssel unterscheidet, der zum Verschlüsseln des Quell-DB-Clusters verwendet wurde. Der neue DB-Cluster wird mit dem KMS-Schlüssel verschlüsselt, der durch den Parameter `KmsKeyId` gekennzeichnet ist.

Wenn Sie keinen Wert für den Parameter `KmsKeyId` angeben, geschieht folgendes:

- Wenn der DB-Cluster verschlüsselt ist, dann wird der wiederhergestellte DB-Cluster mit dem KMS-Schlüssel verschlüsselt, mit dem auch der Quell-DB-Cluster verschlüsselt wurde.
- Wenn der DB-Cluster nicht verschlüsselt ist, dann ist der wiederhergestellte DB-Cluster nicht verschlüsselt.

Wenn sich `DBClusterIdentifier` auf einen DB-Cluster bezieht, der nicht verschlüsselt ist, wird die Wiederherstellungsanforderung abgelehnt.

- Port(in der CLI:`--port`) — ein IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Die Nummer des Ports, an dem der neue DB-Cluster Verbindungen akzeptiert.

  Einschränkungen: Der Wert muss `1150-65535` lauten.

  Standard: Der gleiche Port wie der ursprüngliche DB-Cluster.

- RestoreToTime(in der CLI:`--restore-to-time`) — a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Offset von Mitternacht 1970-01-01).

  Das Datum und die Uhrzeit für die Wiederherstellung des DB-Clusters.

  Gültige Werte: Der Wert muss eine Uhrzeit im UTC-Format (Universal Coordinated Time) sein

  Einschränkungen:

  - Muss vor dem letzten wiederherstellbaren Zeitpunkt für die DB-Instance liegen
  - Muss angegeben werden, wenn der Parameter `UseLatestRestorableTime` nicht angegeben ist
  - Kann nicht angegeben werden, wenn der Parameter `UseLatestRestorableTime` auf "true" festgelegt ist
  - Kann nicht angegeben werden, wenn der Parameter `RestoreType` auf `copy-on-write` festgelegt ist

  Beispiel: `2015-03-07T23:45:00Z`

- RestoreType (in der CLI: `--restore-type`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Typ der auszuführenden Wiederherstellung. Sie können einen der folgenden Werte angeben:

- `full-copy` – Der neue DB-Cluster wird als vollständige Kopie des Quell-DB-Clusters wiederhergestellt.
- `copy-on-write` – Der neue DB-Cluster wird als Klon des Quell-DB-Clusters wiederhergestellt.

Wenn Sie keinen Wert für `RestoreType` angeben, wird der neue DB-Cluster als vollständige Kopie des Quell-DB-Clusters wiederhergestellt.

- ServerlessV2ScalingConfiguration (in der CLI: `--serverless-v2-scaling-configuration`) – Ein Serverless V2 ScalingConfiguration-Objekt.

  Enthält die Skalierungskonfiguration eines Neptune Serverless DB-Clusters.

  Weitere Informationen finden Sie unter Verwendung von Amazon Neptune Serverless im Amazon Neptune-Benutzerhandbuch.
- SourceDBClusterIdentifier (in der CLI: `--source-db-cluster-identifier`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Kennung des Quell-DB-Clusters, von dem wiederhergestellt werden soll.

  Einschränkungen:

  - Muss mit dem Bezeichner eines vorhandenen Bezeichners übereinstimmen. DBCluster
- StorageType (in der CLI: `--storage-type`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Speichertyp an, der dem DB-Cluster zugeordnet werden soll.

  Zulässige Werte: `standard`, `iopt1`

  Standard: `standard`
- Tags (in der CLI: `--tags`) – Ein Array von Markierung Objekten.

  Die Tags, die auf den wiederhergestellten DB-Cluster angewendet werden sollen.
- UseLatestRestorableTime (in der CLI: `--use-latest-restorable-time`) – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Ein Wert, der auf `true` festgelegt ist, um den DB-Cluster auf den letzten wiederherstellbaren Sicherungszeitpunkt wiederherzustellen, und sonst `false` anzeigt.

  Standard: `false`

Einschränkungen: Darf nicht angegeben werden, wenn der Parameter `RestoreToTime` angegeben ist.

- VpcSecurityGroupIds (in der CLI: `--vpc-security-group-ids`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Liste der VPC-Sicherheitsgruppen, zu denen der neue DB-Cluster gehört.

Antwort

Enthält die Details eines Amazon Neptune-DB-Clusters.

Dieser Datentyp wird als Antwortelement in der the section called “Beschreiben DBClusters” verwendet.

- AllocatedStorage— ein IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  `AllocatedStorage` gibt immer 1 zurück, da die Neptune-DB-Cluster-Speichergröße nicht fest ist, sondern sich bei Bedarf automatisch anpasst.
- AssociatedRoles – Ein Array mit DBClusterRolle-Objekten.

  Bietet eine Liste der Amazon Identity and Access Management (IAM)-Rollen, die dem DB-Cluster zugeordnet sind. IAM-Rollen, die einem DB-Cluster zugeordnet sind, erteilen dem DB-Cluster die Berechtigung, auf andere Amazon-Services in Ihrem Namen zuzugreifen.
- AutomaticRestartTime— a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Offset von Mitternacht am 01.01.1970).

  Zeitpunkt, zu dem der DB-Cluster automatisch neu gestartet wird.
- AvailabilityZones – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Stellt die Liste der EC2 Availability Zones bereit, in denen Instances im DB-Cluster erstellt werden können.
- BacktrackConsumedChangeRecords— a LongOptional, vom Typ: `long` (eine 64-Bit-Ganzzahl mit Vorzeichen).

  Wird von Neptune nicht unterstützt.
- BacktrackWindow— a LongOptional, vom Typ: `long` (eine 64-Bit-Ganzzahl mit Vorzeichen).

  Wird von Neptune nicht unterstützt.

- BackupRetentionPeriod— an IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Gibt die Anzahl der Tage an, für die automatische DB-Snapshots aufbewahrt werden.
- Capacity— an IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Wird von Neptune nicht unterstützt.
- CloneGroupId – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Identifiziert die Clone-Gruppe, mit der der DB-Cluster verknüpft ist.
- ClusterCreateTime— a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Offset von Mitternacht am 01.01.1970).

  Gibt den Zeitpunkt an, an dem der DB-Cluster erstellt wurde, in UTC (Universal Coordinated Time).
- CopyTagsToSnapshot— a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Wenn auf *`true`* gesetzt, werden Tags in jeden Snapshot des DB-Clusters kopiert, der erstellt wird.
- CrossAccountClone— a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Wenn auf `true` gesetzt, kann der DB-Cluster kontenübergreifend geklont werden.
- DatabaseName – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Enthält den Namen der initialen Datenbank dieses DB-Clusters, die zum Erstellungszeitpunkt bereitgestellt wurde, sofern eine angegeben wurde, als der DB-Cluster erstellt wurde. Derselbe Name wird über die Lebensdauer des DB-Clusters zurückgegeben.
- DBClusterArn – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Amazon-Ressourcenname (ARN) für den DB-Cluster.
- DBClusterIdentifier – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Enthält eine vom Benutzer bereitgestellte DB-Cluster-Kennung. Diese Kennung ist der eindeutige Schlüssel zur Identifizierung eines DB-Clusters.
- DBClusterMembers – Ein Array mit DBClusterMitglied-Objekten.

  Enthält eine Liste der Instances, aus denen der DB-Cluster besteht.
- DBClusterParameterGroup – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Gibt den Namen der DB-Cluster-Parametergruppe für den DB-Cluster an.

- DbClusterResourceId – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die für die Amazon-Region eindeutige, unveränderliche Kennung für den DB-Cluster. Diese Kennung wird in CloudTrail Amazon-Protokolleinträgen immer dann gefunden, wenn auf den Amazon KMS-Schlüssel für den DB-Cluster zugegriffen wird.
- DBSubnetGroup – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt Informationen zu der Subnetzgruppe an, die dem DB-Cluster zugeordnet ist, einschließlich dem Namen, der Beschreibung und der Subnetze in der Subnetzgruppe.
- DeletionProtection— a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Gibt an, ob der Löschschutz des DB-Clusters aktiviert ist. Die Datenbank kann nicht gelöscht werden, wenn der Löschschutz aktiviert ist.
- EarliestBacktrackTime— a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Offset von Mitternacht 1970-01-01).

  Wird von Neptune nicht unterstützt.
- EarliestRestorableTime— a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Versatz von Mitternacht am 01.01.1970).

  Gibt den frühesten Zeitpunkt an, zu dem eine Datenbank mit point-in-time restore wiederhergestellt werden kann.
- EnabledCloudwatchLogsExports – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Liste der Protokolltypen, für deren Export dieser DB-Cluster in CloudWatch Logs konfiguriert ist. Gültige Protokolltypen sind: `audit` (zum Veröffentlichen von Audit-Logs CloudWatch) und slowquery (zum Veröffentlichen von Slow-Query-Logs). CloudWatch Weitere Informationen finden Sie unter Neptune-Logs in CloudWatch Amazon-Protokollen veröffentlichen.
- Endpoint – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Verbindungsendpunkt für die primäre Instance des DB-Clusters an.
- Engine – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Namen der Datenbank-Engine an, die für diesen DB-Cluster verwendet werden soll.

- EngineVersion – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt die Version der Datenbank-Engine an.

- GlobalClusterIdentifier— a GlobalClusterIdentifier, vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge), nicht weniger als 1 oder mehr als 255? st? s, entspricht diesem regulären Ausdruck:`[A-Za-z][0-9A-Za-z-:._]*`.

  Enthält eine vom Benutzer bereitgestellte gelobale Datenbank-Cluster-ID. Diese Kennung ist der eindeutige Schlüssel zur Identifizierung einer globalen Datenbank.

- HostedZoneId – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt die ID an, die Amazon Route 53 zuweist, wenn Sie eine gehostete Zone erstellen.

- IAMDatabaseAuthenticationEnabled – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Dies ist „true“, wenn die Zuweisung von Amazon Identity and Access Management (IAM)-Konten zu Datenbank-Konten aktiviert ist. Andernfalls „false“.

- IOOptimizedNextAllowedModificationTime— a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Offset von Mitternacht 1970-01-01).

  Beim nächsten Mal können Sie den DB-Cluster so ändern, dass der Speichertyp `iopt1` verwendet wird.

- KmsKeyId – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Wenn `StorageEncrypted` „true“ ist, ist dies die Amazon-KMS-Schlüsselkennung für das verschlüsselte DB-Cluster.

- LatestRestorableTime— a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Versatz von Mitternacht am 01.01.1970).

  Gibt den letzten Zeitpunkt an, zu dem eine Datenbank mit point-in-time restore wiederhergestellt werden kann.

- MultiAZ – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Gibt an, ob der DB-Cluster über Instances in mehreren Availability Zones verfügt.

- PendingModifiedValues – Ein ClusterPendingModifiedValues-Objekt.

  Dieser Datentyp wird als Antwortelement in der `ModifyDBCluster` Operation verwendet und enthält Änderungen, die während des nächsten Wartungsfensters angewendet werden.

- PercentProgress – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Fortschritt der Operation als Prozentsatz an.

- Port— ein IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Gibt die Portnummer an, die von der Datenbank-Engine überwacht wird.

- PreferredBackupWindow – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den täglichen Zeitraum in koordinierter Weltzeit (UTC) an, in dem automatische Sicherungen erstellt werden, wenn automatische Sicherungen aktiviert sind, gemäß `BackupRetentionPeriod`.

- PreferredMaintenanceWindow – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den wöchentlichen Zeitraum, in dem Systemwartungen durchgeführt werden können, in UTC (Universal Coordinated Time) an.

- ReaderEndpoint – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Leser-Endpunkt für den DB-Cluster. Der Reader-Endpunkt für ein DB-Cluster gleicht die Verbindungslasten zwischen den Read Replicas aus, die in einem DB-Cluster verfügbar sind. Während Clients neue Verbindungsanfragen an den Reader-Endpunkt tätigen, verteilt Neptune die Verbindungsanfragen zwischen den Read Replicas im DB-Cluster. Diese Funktionalität sorgt dafür, dass die Workload-Auslastung für Lesevorgänge auf mehrere Read Replicas im DB-Cluster verteilt wird.

  Wenn ein Failover auftritt und das Read Replica, mit dem Sie verbunden sind, als primäre Instance hochgestuft wird, wird Ihre Verbindung unterbrochen. Verbinden Sie sich erneut mit dem Reader-Endpunkt, um mit dem Senden Ihres Workloads für Lesevorgänge an andere Read Replicas im Cluster fortzufahren.

- ReadReplicaIdentifiers – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Enthält eine oder mehrere Kennungen der mit diesem DB-Cluster verbundenen Read Replicas.

- ReplicationSourceIdentifier – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Wird von Neptune nicht unterstützt.

- ReplicationType – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Wird von Neptune nicht unterstützt.

- ServerlessV2ScalingConfiguration – Ein Serverless V2 ScalingConfigurationInfo-Objekt.

  Zeigt die Skalierungskonfiguration für einen Neptune Serverless DB-Cluster.

  Weitere Informationen finden Sie unter Verwendung von Amazon Neptune Serverless im Amazon Neptune-Benutzerhandbuch.
- Status – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den aktuellen Status dieses DB-Clusters an.
- StorageEncrypted – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Gibt an, ob der DB-Cluster verschlüsselt ist.
- StorageType – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Speichertyp, der vom DB-Cluster verwendet wird.

  Zulässige Werte:
  - **`standard`**— (Standard) Bietet kostengünstigen Datenbankspeicher für Anwendungen mit mäßiger bis geringer I/O Nutzung.
  - **`iopt1`**— Ermöglicht I/O-optimierten Speicher, der auf die Durchsatzanforderungen zugeschnitten ist. I/O-intensive graph workloads that require predictable pricing with low I/O latency and consistent I/O

    E/A-optimierter Neptune-Speicher ist erst ab Engine-Version 1.3.0.0 verfügbar.
- VpcSecurityGroups – Ein Array mit VpcSecurityGroupMembership-Objekten.

  Stellt eine Liste der VPC-Sicherheitsgruppen zur Verfügung, zu denen das DB-Cluster gehört.

Fehler

- DBClusterAlreadyExistsFault
- DBClusterNotFoundFault
- DBClusterQuotaExceededFault
- DBClusterSnapshotNotFoundFault
- DBSubnetGroupNotFoundFault
- Unzureichend DBCluster CapacityFault
- InsufficientStorageClusterCapacityFault

- Ungültig DBCluster SnapshotStateFault
- Ungültig DBCluster StateFault
- Ungültig DBSnapshot StateFault
- InvalidRestoreFault
- InvalidSubnet
- Ungültig VPCNetwork StateFault
- KMSKeyNotAccessibleFault
- OptionGroupNotFoundFault
- StorageQuotaExceededFault
- DBClusterParameterGroupNotFoundFault

Beschreiben Sie DBCluster Schnappschüsse (Aktion)

Der AWS CLI-Name für diese API lautet:`describe-db-cluster-snapshots`.

Gibt Informationen zu DB-Cluster-Snapshots zurück. Diese API-Aktion unterstützt Paginierung.

Anforderung

- DBClusterIdentifier (in der CLI: `--db-cluster-identifier`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die ID des DB-Clusters, für den die Liste von DB-Cluster-Snapshots abgerufen werden soll. Dieser Parameter kann nicht in Verbindung mit dem Parameter `DBClusterSnapshotIdentifier` verwendet werden. Bei diesem Parameter wird nicht zwischen Groß- und Kleinschreibung unterschieden.

  Einschränkungen:

  - Falls angegeben, muss er mit dem Bezeichner einer vorhandenen Datei übereinstimmen DBCluster.
- DBClusterSnapshotIdentifier (in der CLI: `--db-cluster-snapshot-identifier`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine bestimmte zu beschreibende DB-Cluster-Snapshot-Kennung. Dieser Parameter kann nicht in Verbindung mit dem Parameter `DBClusterIdentifier` verwendet werden. Dieser Wert wird als Zeichenfolge in Kleinbuchstaben gespeichert.

Einschränkungen:

- Falls angegeben, muss sie mit der ID eines vorhandenen DBCluster Snapshots übereinstimmen.
- Wenn diese Kennung für einen automatisierten Snapshot ist, muss auch der Parameter `SnapshotType` angegeben werden.

- Filters (in der CLI: `--filters`) – Ein Array von Filter Objekten.

  Dieser Parameter wird derzeit nicht unterstützt.

- IncludePublic (in der CLI: `--include-public`) – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  „True“, um manuelle DB-Cluster-Snapshots einzuschließen, die öffentlich sind und von jedem beliebigen Amazon-Konto kopiert oder wiederhergestellt werden können, und andernfalls „false“. Der Standardwert ist `false`. Der Standardwert lautet „false“.

  Sie können mithilfe der API-Aktion the section called “Modifizieren DBCluster SnapshotAttribute” einen manuellen DB-Cluster-Snapshot als öffentlich freigeben.

- IncludeShared (in der CLI: `--include-shared`) – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  „True“, um freigegebene manuelle DB-Cluster-Snapshots von anderen Amazon-Konten einzuschließen, für die diesem Amazon-Konto die Berechtigung zum Kopieren oder Wiederherstellen erteilt wurde, und andernfalls „false“. Der Standardwert ist `false`.

  Sie können mit der API-Aktion the section called “Modifizieren DBCluster SnapshotAttribute” einem Amazon-Konto die Berechtigung zum Wiederherstellen eines manuellen DB-Cluster-Snapshots aus einem anderen Amazon-Konto erteilen.

- Marker (in der CLI: `--marker`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Ein optionales Paginierungstoken, das von einer vorherigen `DescribeDBClusterSnapshots`-Anforderung bereitgestellt wird. Wenn Sie diesen Parameter angeben, enthält die Antwort nur die Datensätze zwischen der Markierung und dem durch `MaxRecords` angegebenen Wert.

- MaxRecords(in der CLI:`--max-records`) — ein IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Die maximale Anzahl der in der Antwort zurückgegebenen Datensätze. Wenn mehrere Datensätze vorhanden sind, als der Wert `MaxRecords` angibt, ist ein Paginierungstoken mit dem Namen einer

Markierung in der Antwort enthalten, sodass die verbleibenden Ergebnisse abgerufen werden können.

Standard: 100

Einschränkungen: Mindestwert 20, Höchstwert 100.

- SnapshotType (in der CLI: `--snapshot-type`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Typ der DB-Cluster-Snapshots, der zurückgegeben werden soll. Sie können einen der folgenden Werte angeben:

  - `automated` – Rückgabe aller DB-Cluster-Snapshots, die von Amazon Neptune automatisch für mein Amazon-Konto verwendet wurden.
  - `manual` – Rückgabe aller DB-Cluster-Snapshots, die von meinem Amazon-Konto verwendet wurden.
  - `shared` – Rückgabe aller manuellen DB-Cluster-Snapshots, die mit meinem Amazon-Konto geteilt wurden.
  - `public` – Rückgabe aller DB-Cluster-Snapshots, die als öffentlich markiert wurden.

  Wenn Sie keinen Wert für `SnapshotType` angeben, werden sowohl automatisierte als auch manuelle DB-Cluster-Snapshots zurückgegeben. Sie können freigegebene DB-Cluster-Snapshots zu diesen Ergebnissen hinzufügen, indem Sie den Parameter `IncludeShared` auf `true` festlegen. Sie können öffentliche DB-Cluster-Snapshots zu diesen Ergebnissen hinzufügen, indem Sie den Parameter `IncludePublic` auf `true` festlegen.

  Die Parameter `IncludePublic` und `IncludeShared` gelten nicht für `SnapshotType`-Werte von `manual` oder `automated`. Der Parameter `IncludePublic` gilt nicht, wenn `SnapshotType` auf `shared` festgelegt ist. Der Parameter `IncludeShared` gilt nicht, wenn `SnapshotType` auf `public` festgelegt ist.

Antwort

- DBClusterSnapshots – Ein Array mit DBClusterSchnappschuss-Objekten.

  Stellt eine Liste von DB-Cluster-Snapshots für den Benutzer bereit.

- Marker – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Ein optionales Paginierungstoken, das von einer vorherigen the section called “Beschreiben Sie Schnappschüsse DBCluster”-Anforderung bereitgestellt wird. Wenn Sie diesen Parameter angeben, enthält die Antwort nur die Datensätze zwischen der Markierung und dem durch `MaxRecords` angegebenen Wert.

Fehler

- DBClusterSnapshotNotFoundFault

Beschreiben Sie DBCluster SnapshotAttributes (Aktion)

Der AWS CLI-Name für diese API lautet:`describe-db-cluster-snapshot-attributes`.

Gibt eine Liste der Namen und Werte von DB-Cluster-Attributen eines manuellen DB-Cluster-Snapshots zurück.

Wenn Snapshots mit anderen Amazon-Konten geteilt werden, werden das `restore` Attribut und eine Liste der IDs Amazon-Konten `DescribeDBClusterSnapshotAttributes` zurückgegeben, die berechtigt sind, den manuellen DB-Cluster-Snapshot zu kopieren oder wiederherzustellen. Wenn `all` in der Liste der Werte für das Attribut `restore` enthalten ist, dann ist der manuelle DB-Cluster-Snapshot öffentlich und kann von allen Amazon-Konten kopiert oder wiederhergestellt werden.

Um einem Amazon-Konto den Zugriff für das Kopieren oder Wiederherstellen eines manuellen DB-Cluster-Snapshots zu erteilen oder ihn von ihm zu entfernen, oder um einen manuellen DB-Cluster-Snapshot öffentlich oder privat zu machen, verwenden Sie die API-Aktion the section called “Modifizieren DBCluster SnapshotAttribute”.

Anforderung

- DBClusterSnapshotIdentifier (in der CLI: `--db-cluster-snapshot-identifier`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Kennung für den DB-Cluster-Snapshot, für den die Attribute beschrieben werden sollen.

Antwort

Enthält die Ergebnisse eines erfolgreichen Aufrufs an die API-Aktion the section called “Beschreiben DBCluster SnapshotAttributes”.

Manuelle DB-Cluster-Snapshot-Attribute werden verwendet, um andere Amazon-Konten zu autorisieren und einen manuellen DB-Cluster-Snapshot zu kopieren oder wiederherzustellen. Weitere Informationen finden Sie in der API-Aktion the section called “Modifizieren DBCluster SnapshotAttribute”.

- DBClusterSnapshotAttributes – Ein Array mit DBClusterSnapshotAttribute-Objekten.

  Die Liste der Attribute und Werte für den manuellen DB-Cluster-Snapshot.
- DBClusterSnapshotIdentifier – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Kennung des manuellen DB-Cluster-Snapshots, für den die Attribute gelten.

Fehler

- DBClusterSnapshotNotFoundFault

Strukturen:

DBClusterSnapshot (Struktur)

Enthält die Details für einen Amazon Neptune-DB-Cluster-Snapshot.

Dieser Datentyp wird als Antwortelement in der Aktion the section called “Beschreiben Sie Schnappschüsse DBCluster” verwendet.

Felder

- AllocatedStorage – Dies ist eine Ganzzahl vom Typ `integer` (32-Bit-Ganzzahl mit Vorzeichen).

  Legt die Größe des zugewiesenen Speichers in Gibibyte (GiB) fest.
- AvailabilityZones – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Stellt die Liste der EC2 Availability Zones bereit, in denen Instances im DB-Cluster-Snapshot wiederhergestellt werden können.
- ClusterCreateTime— Dies ist ein TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als ein Offset von Mitternacht 1970-01-01).

  Gibt den Zeitpunkt an, an dem der DB-Cluster erstellt wurde, in UTC (Universal Coordinated Time).

- DBClusterIdentifier – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Legt die DB-Cluster-Kennung des DB-Clusters fest, anhand dessen dieser DB-Cluster-Snapshot erstellt wurde.
- DBClusterSnapshotArn – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Amazon-Ressourcenname (ARN) für den DB-Cluster-Snapshot.
- DBClusterSnapshotIdentifier – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Legt die Kennung für den DB-Cluster-Snapshot fest. Muss mit der Kennung eines vorhandenen Snapshots übereinstimmen.

  Nachdem Sie einen DB-Cluster mit einem `DBClusterSnapshotIdentifier` wiederhergestellt haben, müssen Sie denselben `DBClusterSnapshotIdentifier` für zukünftige Aktualisierungen des DB-Clusters angeben. Wenn Sie diese Eigenschaft für eine Aktualisierung angeben, wird der DB-Cluster nicht erneut aus dem Snapshot wiederhergestellt und die Daten in der Datenbank werden nicht geändert.

  Sollten Sie die `DBClusterSnapshotIdentifier`-Eigenschaft jedoch nicht angeben, wird ein leerer DB-Cluster erstellt und der ursprüngliche DB-Cluster wird gelöscht. Wenn Sie eine Eigenschaft angeben, die sich von der vorherigen Snapshot-Eigenschaft zur Wiederherstellung unterscheidet, wird der DB-Cluster aus dem Snapshot wiederhergestellt, der durch den `DBClusterSnapshotIdentifier` angegeben wurde, und der ursprüngliche DB-Cluster gelöscht.
- Engine – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Legt den Namen der Datenbank-Engine fest.
- EngineVersion – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Stellt die Version der Datenbank-Engine für diesen DB-Cluster-Snapshot bereit.
- IAMDatabaseAuthenticationEnabled – Dies ist ein boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Dies ist „true“, wenn die Zuweisung von Amazon Identity and Access Management (IAM)-Konten zu Datenbank-Konten aktiviert ist. Andernfalls „false“.
- KmsKeyId – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Wenn `StorageEncrypted` „true“ ist, ist dies die Amazon-KMS-Schlüsselkennung für den verschlüsselten DB-Cluster-Snapshot.

- LicenseModel – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Stellt die Lizenzmodellinformationen für diesen DB-Cluster-Snapshot bereit.

- PercentProgress – Dies ist eine Ganzzahl vom Typ `integer` (32-Bit-Ganzzahl mit Vorzeichen).

  Gibt einen Prozentsatz der Daten an, die laut Schätzung bereits übertragen wurden.

- Port – Dies ist eine Ganzzahl vom Typ `integer` (32-Bit-Ganzzahl mit Vorzeichen).

  Gibt den Port an, den der DB-Cluster zum Zeitpunkt des Snapshots überwacht hat.

- SnapshotCreateTime— Dies ist ein TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als ein Offset von Mitternacht 1970-01-01).

  Gibt das Datum und die Uhrzeit der Erstellung des Snapshots in koordinierter Weltzeit (UTC) an.

- SnapshotType – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Stellt den Typ des DB-Cluster-Snapshots bereit.

- SourceDBClusterSnapshotArn – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Falls der DB-Cluster-Snapshot von einem Quell-DB-Cluster-Snapshot kopiert wurde, der Amazon-Ressourcenname (ARN) für den Quell-DB-Cluster-Snapshot, andernfalls ein Null-Wert.

- Status – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Legt den Status dieses DB-Cluster-Snapshots fest.

- StorageEncrypted – Dies ist ein boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Gibt an, ob der DB-Cluster-Snapshot verschlüsselt ist.

- StorageType – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Speichertyp, der dem DB-Cluster-Snapshot zugeordnet ist.

- VpcId – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Stellt die VPC-ID bereit, die dem DB-Cluster-Snapshot zugeordnet ist.

`DBClusterSnapshot` wird als Antwortelement verwendet für:

- DBClusterSnapshot erstellen
- DBClusterSnapshot kopieren
- DBClusterSchnappschuss löschen

DBClusterSnapshotAttribute (Struktur)

Enthält die Namen und die Werte eines manuellen DB-Cluster-Snapshot-Attributs.

Manuelle DB-Cluster-Snapshot-Attribute werden verwendet, um andere Amazon-Konten zu autorisieren und einen manuellen DB-Cluster-Snapshot wiederherzustellen. Weitere Informationen finden Sie in der API-Aktion the section called “Modifizieren DBCluster SnapshotAttribute”.

Felder

- AttributeName – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Name des manuellen DB-Cluster-Snapshot-Attributs.

  Das Attribut mit dem Namen `restore` bezieht sich auf die Liste der Amazon-Konten, die über die Berechtigung zum Kopieren oder Wiederherstellen des manuellen DB-Cluster-Snapshots verfügen. Weitere Informationen finden Sie in der API-Aktion the section called “Modifizieren DBCluster SnapshotAttribute”.
- AttributeValues – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der bzw. die Werte für das manuelle DB-Cluster-Snapshot-Attribut.

  Wenn das `AttributeName` Feld auf gesetzt ist`restore`, gibt dieses Element eine Liste IDs der Amazon-Konten zurück, die berechtigt sind, den manuellen DB-Cluster-Snapshot zu kopieren oder wiederherzustellen. Wenn sich in der Liste der Wert `all` befindet, ist der manuelle DB-Cluster-Snapshot öffentlich und für jedes Amazon-Konto zum Kopieren oder Wiederherstellen verfügbar.

DBClusterSnapshotAttributesResult (Struktur)

Enthält die Ergebnisse eines erfolgreichen Aufrufs an die API-Aktion the section called “Beschreiben DBCluster SnapshotAttributes”.

Manuelle DB-Cluster-Snapshot-Attribute werden verwendet, um andere Amazon-Konten zu autorisieren und einen manuellen DB-Cluster-Snapshot zu kopieren oder wiederherzustellen. Weitere Informationen finden Sie in der API-Aktion the section called “Modifizieren DBCluster SnapshotAttribute”.

Felder

- DBClusterSnapshotAttributes – Dies ist ein Array von DBClusterSnapshotAttribute-Objekten.

  Die Liste der Attribute und Werte für den manuellen DB-Cluster-Snapshot.
- DBClusterSnapshotIdentifier – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Kennung des manuellen DB-Cluster-Snapshots, für den die Attribute gelten.

`DBClusterSnapshotAttributesResult` wird als Antwortelement verwendet für:

- Beschreiben DBCluster SnapshotAttributes
- Modifizieren DBCluster SnapshotAttribute

API von Neptune Ereignissen

Aktionen:

- CreateEventSubscription (Aktion)
- DeleteEventSubscription (Aktion)
- ModifyEventSubscription (Aktion)
- DescribeEventSubscriptions (Aktion)
- AddSourceIdentifierToSubscription (Aktion)
- RemoveSourceIdentifierFromSubscription (Aktion)
- DescribeEvents (Aktion)
- DescribeEventCategories (Aktion)

Strukturen:

- Ereignis (Struktur)

- EventCategoriesMap (Struktur)
- EventSubscription (Struktur)

CreateEventSubscription (Aktion)

Der AWS CLI-Name für diese API lautet:`create-event-subscription`.

Erstellt ein Abonnement für Ereignisbenachrichtigungen. Diese Aktion erfordert einen Thema-ARN (Amazon-Ressourcenname), der entweder von der Neptune Konsole, der SNS-Konsole oder der SNS-API erstellt wurde. Um einen ARN mit SNS zu erstellen, müssen Sie ein Thema in Amazon SNS erstellen und das Thema abonnieren. Der ARN wird in der SNS-Konsole angezeigt.

Sie können den Typ der Quelle (SourceType) angeben, über die Sie benachrichtigt werden möchten, eine Liste von Neptunquellen (SourceIds) bereitstellen, die die Ereignisse auslösen, und eine Liste von Ereigniskategorien (EventCategories) für Ereignisse bereitstellen, über die Sie benachrichtigt werden möchten. Sie können beispielsweise SourceType = db-instance, = mydbinstance1, mydbinstance2 und SourceIds = Availability, Backup angeben. EventCategories

Wenn Sie sowohl als auch angeben SourceIds, z. B. SourceType = db-instance SourceType und SourceIdentifier = my, werden Sie über alle DB-Instance-Ereignisse für die angegebene DBInstance1 Quelle informiert. Wenn Sie a angeben, SourceType aber keine angeben SourceIdentifier, erhalten Sie eine Benachrichtigung über die Ereignisse für diesen Quelltyp für alle Ihre Neptunquellen. Wenn Sie SourceType weder das noch das angeben, werden Sie über Ereignisse informiert SourceIdentifier, die aus allen Neptune-Quellen generiert wurden, die zu Ihrem Kundenkonto gehören.

Anforderung

- Enabled(in der CLI:`--enabled`) — a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Ein boolescher Wert. Legen Sie ihn auf true fest, um das Abonnement zu aktivieren, legen Sie ihn auf false fest, um das Abonnement zu erstellen, aber nicht zu aktivieren.
- EventCategories (in der CLI: `--event-categories`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Liste von Event-Kategorien für eine SourceType , die Sie abonnieren möchten. Mithilfe der DescribeEventCategoriesAktion können Sie sich eine Liste der Kategorien für ein bestimmtes SourceType Objekt anzeigen lassen.

- SnsTopicArn (in der CLI: `--sns-topic-arn`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Amazon-Ressourcenname (ARN) des SNS-Themas, das für die Ereignisbenachrichtigung erstellt wurde. Der ARN wird von Amazon SNS erstellt, wenn Sie ein Thema erstellen und es abonnieren.

- SourceIds (in der CLI: `--source-ids`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Liste der IDs der Ereignisquellen, für die Ereignisse zurückgegeben werden. Wenn nicht angegeben, werden alle Quellen zur Antwort hinzugefügt. Eine ID muss mit einem Buchstaben beginnen und darf nur ASCII-Buchstaben, Ziffern und Bindestriche enthalten; sie darf nicht mit einem Bindestrich oder zwei aufeinander folgenden Bindestrichen enden.

  Einschränkungen:

  - Wenn SourceIds angegeben, SourceType muss auch angegeben werden.
  - Wenn der Quelltyp eine DB-Instance ist, muss ein `DBInstanceIdentifier` angegeben werden.
  - Wenn der Quelltyp eine DB-Sicherheitsgruppe ist, muss ein `DBSecurityGroupName` angegeben werden.
  - Wenn der Quelltyp eine DB-Parametergruppe ist, muss ein `DBParameterGroupName` angegeben werden.
  - Wenn der Quelltyp ein DB-Snapshot ist, muss ein `DBSnapshotIdentifier` angegeben werden.

- SourceType (in der CLI: `--source-type`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Typ der Quelle, die die Ereignisse generiert. Wenn Sie beispielsweise über Ereignisse benachrichtigt werden möchten, die von einer DB-Instance generiert werden, legen Sie diesen Parameter auf "db-instance" fest. Wenn dieser Wert nicht festgelegt ist, werden alle Ereignisse zurückgegeben.

  Zulässige Werte: `db-instance` | `db-cluster` | `db-parameter-group` | `db-security-group` | `db-snapshot` | `db-cluster-snapshot`

- SubscriptionName (in der CLI: `--subscription-name`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Der Name des Abonnements.

Einschränkungen: Der Name muss weniger als 255 Zeichen lang sein.

- Tags (in der CLI: `--tags`) – Ein Array von Markierung Objekten.

  Die Tags, die auf das neue Ereignisabonnement angewendet werden sollen.

Antwort

Enthält die Ergebnisse eines erfolgreichen Aufrufs der the section called "DescribeEventSubscriptions"-Aktion.

- CustomerAwsId – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Das Amazon-Kundenkonto, das dem Abonnement für Ereignisbenachrichtigungen zugeordnet ist.
- CustSubscriptionId – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Abonnement-ID für die Ereignisbenachrichtigung.
- Enabled – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Ein boolescher Wert, der angibt, ob das Abonnement aktiviert ist. "True" zeigt an, dass das Abonnement aktiviert ist.
- EventCategoriesList – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Liste von Ereigniskategorien für das Abonnement für die Ereignisbenachrichtigung.
- EventSubscriptionArn – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Amazon-Ressourcenname (ARN) für das Ereignisabonnement.
- SnsTopicArn – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der ARN des Themas des Abonnements für die Ereignisbenachrichtigung.
- SourceIdsList – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Quellenliste IDs für das Abonnement für Ereignisbenachrichtigungen.
- SourceType – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Quelltyp für das Abonnement für die Ereignisbenachrichtigung.
- Status – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Der Status des Abonnements für die Ereignis-Benachrichtigung.

Einschränkungen:

Dabei kann es sich um eine der folgenden Optionen handeln: erstellen | ändern | löschen | aktiv | keine Erlaubnis | topic-not-exist

Der Status "Keine Berechtigung" gibt an, dass Neptune keine Berechtigung mehr hat, an das SNS-Thema zu posten. Der Status "topic-not-exist" gibt an, dass das Thema nach der Erstellung des Abonnements gelöscht wurde.

- SubscriptionCreationTime – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Uhrzeit, zu der das Abonnement für die Ereignisbenachrichtigung erstellt wurde.

Fehler

- EventSubscriptionQuotaExceededFault
- SubscriptionAlreadyExistFault
- SNSInvalidTopicFault
- SNSNoAuthorizationFault
- SNSTopicArnNotFoundFault
- SubscriptionCategoryNotFoundFault
- SourceNotFoundFault

DeleteEventSubscription (Aktion)

Der AWS CLI-Name für diese API lautet:`delete-event-subscription`.

Löscht ein Abonnement für Ereignisbenachrichtigungen.

Anforderung

- SubscriptionName (in der CLI: `--subscription-name`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Name des Abonnements für Ereignisbenachrichtigungen, das Sie löschen möchten.

Antwort

Enthält die Ergebnisse eines erfolgreichen Aufrufs der the section called “DescribeEventSubscriptions”-Aktion.

- CustomerAwsId – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Das Amazon-Kundenkonto, das dem Abonnement für Ereignisbenachrichtigungen zugeordnet ist.
- CustSubscriptionId – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Abonnement-ID für die Ereignisbenachrichtigung.
- Enabled – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Ein boolescher Wert, der angibt, ob das Abonnement aktiviert ist. "True" zeigt an, dass das Abonnement aktiviert ist.
- EventCategoriesList – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Liste von Ereigniskategorien für das Abonnement für die Ereignisbenachrichtigung.
- EventSubscriptionArn – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Amazon-Ressourcenname (ARN) für das Ereignisabonnement.
- SnsTopicArn – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der ARN des Themas des Abonnements für die Ereignisbenachrichtigung.
- SourceIdsList – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Quellenliste IDs für das Abonnement für Ereignisbenachrichtigungen.
- SourceType – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Quelltyp für das Abonnement für die Ereignisbenachrichtigung.
- Status – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Status des Abonnements für die Ereignis-Benachrichtigung.

  Einschränkungen:

  Dabei kann es sich um eine der folgenden Optionen handeln: erstellen | ändern | löschen | aktiv | keine Erlaubnis | topic-not-exist

Der Status "Keine Berechtigung" gibt an, dass Neptune keine Berechtigung mehr hat, an das SNS-Thema zu posten. Der Status "topic-not-exist" gibt an, dass das Thema nach der Erstellung des Abonnements gelöscht wurde.

- SubscriptionCreationTime – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Uhrzeit, zu der das Abonnement für die Ereignisbenachrichtigung erstellt wurde.

Fehler

- SubscriptionNotFoundFault
- InvalidEventSubscriptionStateFault

ModifyEventSubscription (Aktion)

Der AWS CLI-Name für diese API lautet:`modify-event-subscription`.

Ändert ein bestehendes Abonnement für Ereignisbenachrichtigungen. Beachten Sie, dass Sie nicht mithilfe dieses Aufrufs die Quell-IDs ändern können. Um die Quell-IDs für ein Abonnement zu ändern, verwenden Sie die Aufrufe the section called “AddSourceIdentifierToSubscription” und the section called “RemoveSourceIdentifierFromSubscription”.

Mithilfe der DescribeEventCategoriesAktion können Sie eine Liste der Ereigniskategorien für eine bestimmte SourceType Person anzeigen.

Anforderung

- Enabled(in der CLI:`--enabled`) — a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Ein boolescher Wert. Legen Sie ihn auf true fest, um das Abonnement zu aktivieren.

- EventCategories (in der CLI: `--event-categories`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Liste von Event-Kategorien für eine SourceType , die Sie abonnieren möchten. Mithilfe der DescribeEventCategoriesAktion können Sie sich eine Liste der Kategorien für ein bestimmtes SourceType Objekt anzeigen lassen.

- SnsTopicArn (in der CLI: `--sns-topic-arn`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Amazon-Ressourcenname (ARN) des SNS-Themas, das für die Ereignisbenachrichtigung erstellt wurde. Der ARN wird von Amazon SNS erstellt, wenn Sie ein Thema erstellen und es abonnieren.

- SourceType (in der CLI: `--source-type`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Typ der Quelle, die die Ereignisse generiert. Wenn Sie beispielsweise über Ereignisse benachrichtigt werden möchten, die von einer DB-Instance generiert werden, legen Sie diesen Parameter auf "db-instance" fest. Wenn dieser Wert nicht festgelegt ist, werden alle Ereignisse zurückgegeben.

  Gültige Werte: db-instance | | db-parameter-group | db-security-group db-snapshot

- SubscriptionName (in der CLI: `--subscription-name`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Name des Abonnements für die -Ereignis-Benachrichtigung.

Antwort

Enthält die Ergebnisse eines erfolgreichen Aufrufs der the section called “DescribeEventSubscriptions”-Aktion.

- CustomerAwsId – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Das Amazon-Kundenkonto, das dem Abonnement für Ereignisbenachrichtigungen zugeordnet ist.

- CustSubscriptionId – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Abonnement-ID für die Ereignisbenachrichtigung.

- Enabled – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Ein boolescher Wert, der angibt, ob das Abonnement aktiviert ist. "True" zeigt an, dass das Abonnement aktiviert ist.

- EventCategoriesList – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Liste von Ereigniskategorien für das Abonnement für die Ereignisbenachrichtigung.

- EventSubscriptionArn – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Amazon-Ressourcenname (ARN) für das Ereignisabonnement.
- SnsTopicArn – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der ARN des Themas des Abonnements für die Ereignisbenachrichtigung.
- SourceIdsList – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Quellenliste für das Abonnement IDs für die Ereignisbenachrichtigung.
- SourceType – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Quelltyp für das Abonnement für die Ereignisbenachrichtigung.
- Status – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Status des Abonnements für die Ereignis-Benachrichtigung.

  Einschränkungen:

  Dabei kann es sich um eine der folgenden Optionen handeln: erstellen | ändern | löschen | aktiv | keine Erlaubnis | topic-not-exist

  Der Status "Keine Berechtigung" gibt an, dass Neptune keine Berechtigung mehr hat, an das SNS-Thema zu posten. Der Status "topic-not-exist" gibt an, dass das Thema nach der Erstellung des Abonnements gelöscht wurde.
- SubscriptionCreationTime – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Uhrzeit, zu der das Abonnement für die Ereignisbenachrichtigung erstellt wurde.

Fehler

- EventSubscriptionQuotaExceededFault
- SubscriptionNotFoundFault
- SNSInvalidTopicFault
- SNSNoAuthorizationFault
- SNSTopicArnNotFoundFault
- SubscriptionCategoryNotFoundFault

DescribeEventSubscriptions (Aktion)

Der AWS CLI-Name für diese API lautet:`describe-event-subscriptions`.

Listet alle Abonnementbeschreibungen für ein Kundenkonto auf. Die Beschreibung für ein Abonnement umfasst SNSTopic ARN SubscriptionName, CustomerID SourceType, SourceID und Status CreationTime.

Wenn Sie eine angeben SubscriptionName, wird die Beschreibung für dieses Abonnement aufgeführt.

Anforderung

- Filters (in der CLI: `--filters`) – Ein Array von Filter Objekten.

  Dieser Parameter wird derzeit nicht unterstützt.
- Marker (in der CLI: `--marker`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Ein optionales Paginierungstoken, das in einer früheren DescribeOrderable DBInstance Options-Anfrage bereitgestellt wurde. Wenn Sie diesen Parameter angeben, enthält die Antwort nur die Datensätze zwischen der Markierung und dem durch `MaxRecords` angegebenen Wert.
- MaxRecords(in der CLI:`--max-records`) — ein IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Die maximale Anzahl der in der Antwort zurückgegebenen Datensätze. Wenn mehrere Datensätze vorhanden sind, als der Wert `MaxRecords` angibt, ist ein Paginierungstoken mit dem Namen einer Markierung in der Antwort enthalten, sodass die verbleibenden Ergebnisse abgerufen werden können.

  Standard: 100

  Einschränkungen: Mindestwert 20, Höchstwert 100.
- SubscriptionName (in der CLI: `--subscription-name`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Name des Abonnements für Ereignisbenachrichtigungen, das Sie beschreiben möchten.

Antwort

- EventSubscriptionsList – Ein Array mit EventSubscription-Objekten.

  Eine Liste von EventSubscriptions Datentypen.

- Marker – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Ein optionales Paginierungstoken, das in einer früheren DescribeOrderable DBInstance Options-Anfrage bereitgestellt wurde. Wenn Sie diesen Parameter angeben, enthält die Antwort nur die Datensätze zwischen der Markierung und dem durch `MaxRecords` angegebenen Wert.

Fehler

- SubscriptionNotFoundFault

AddSourceIdentifierToSubscription (Aktion)

Der AWS CLI-Name für diese API lautet:`add-source-identifier-to-subscription`.

Fügt eine Quell-ID einem Abonnement für Ereignisbenachrichtigungen hinzu.

Anforderung

- SourceIdentifier (in der CLI: `--source-identifier`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die ID der Ereignisquelle, die hinzugefügt werden soll.

  Einschränkungen:

  - Wenn der Quelltyp eine DB-Instance ist, muss ein `DBInstanceIdentifier` angegeben werden.
  - Wenn der Quelltyp eine DB-Sicherheitsgruppe ist, muss ein `DBSecurityGroupName` angegeben werden.
  - Wenn der Quelltyp eine DB-Parametergruppe ist, muss ein `DBParameterGroupName` angegeben werden.
  - Wenn der Quelltyp ein DB-Snapshot ist, muss ein `DBSnapshotIdentifier` angegeben werden.

- SubscriptionName (in der CLI: `--subscription-name`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Der Name des Abonnements für Ereignisbenachrichtigungen, dem Sie eine Quell-ID hinzufügen möchten.

Antwort

Enthält die Ergebnisse eines erfolgreichen Aufrufs der the section called “DescribeEventSubscriptions”-Aktion.

- CustomerAwsId – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Das Amazon-Kundenkonto, das dem Abonnement für Ereignisbenachrichtigungen zugeordnet ist.
- CustSubscriptionId – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Abonnement-ID für die Ereignisbenachrichtigung.
- Enabled – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Ein boolescher Wert, der angibt, ob das Abonnement aktiviert ist. "True" zeigt an, dass das Abonnement aktiviert ist.
- EventCategoriesList – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Liste von Ereigniskategorien für das Abonnement für die Ereignisbenachrichtigung.
- EventSubscriptionArn – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Amazon-Ressourcenname (ARN) für das Ereignisabonnement.
- SnsTopicArn – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der ARN des Themas des Abonnements für die Ereignisbenachrichtigung.
- SourceIdsList – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Quellenliste IDs für das Abonnement für Ereignisbenachrichtigungen.
- SourceType – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Quelltyp für das Abonnement für die Ereignisbenachrichtigung.
- Status – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Status des Abonnements für die Ereignis-Benachrichtigung.

  Einschränkungen:

Dabei kann es sich um eine der folgenden Optionen handeln: erstellen | ändern | löschen | aktiv | keine Erlaubnis | topic-not-exist

Der Status "Keine Berechtigung" gibt an, dass Neptune keine Berechtigung mehr hat, an das SNS-Thema zu posten. Der Status "topic-not-exist" gibt an, dass das Thema nach der Erstellung des Abonnements gelöscht wurde.

- SubscriptionCreationTime – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Uhrzeit, zu der das Abonnement für die Ereignisbenachrichtigung erstellt wurde.

Fehler

- SubscriptionNotFoundFault
- SourceNotFoundFault

RemoveSourceIdentifierFromSubscription (Aktion)

Der AWS CLI-Name für diese API lautet:`remove-source-identifier-from-subscription`.

Entfernt eine Quell-ID aus einem bestehenden Abonnement für Ereignisbenachrichtigungen.

Anforderung

- SourceIdentifier (in der CLI: `--source-identifier`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Quell-ID, die von dem Abonnement entfernt werden soll, wie z. B. die DB-Instance-Kennung für eine DB-Instance oder der Name einer Sicherheitsgruppe.
- SubscriptionName (in der CLI: `--subscription-name`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Name des Abonnements für Ereignisbenachrichtigungen, von dem Sie eine Quell-ID entfernen möchten.

Antwort

Enthält die Ergebnisse eines erfolgreichen Aufrufs der the section called “DescribeEventSubscriptions”-Aktion.

- CustomerAwsId – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Das Amazon-Kundenkonto, das dem Abonnement für Ereignisbenachrichtigungen zugeordnet ist.
- CustSubscriptionId – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Abonnement-ID für die Ereignisbenachrichtigung.
- Enabled – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Ein boolescher Wert, der angibt, ob das Abonnement aktiviert ist. "True" zeigt an, dass das Abonnement aktiviert ist.
- EventCategoriesList – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Liste von Ereigniskategorien für das Abonnement für die Ereignisbenachrichtigung.
- EventSubscriptionArn – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Amazon-Ressourcenname (ARN) für das Ereignisabonnement.
- SnsTopicArn – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der ARN des Themas des Abonnements für die Ereignisbenachrichtigung.
- SourceIdsList – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Quellenliste IDs für das Abonnement für Ereignisbenachrichtigungen.
- SourceType – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Quelltyp für das Abonnement für die Ereignisbenachrichtigung.
- Status – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Status des Abonnements für die Ereignis-Benachrichtigung.

  Einschränkungen:

  Dabei kann es sich um eine der folgenden Optionen handeln: erstellen | ändern | löschen | aktiv | keine Erlaubnis | topic-not-exist

Der Status "Keine Berechtigung" gibt an, dass Neptune keine Berechtigung mehr hat, an das SNS-Thema zu posten. Der Status "topic-not-exist" gibt an, dass das Thema nach der Erstellung des Abonnements gelöscht wurde.

- SubscriptionCreationTime – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Uhrzeit, zu der das Abonnement für die Ereignisbenachrichtigung erstellt wurde.

Fehler

- SubscriptionNotFoundFault
- SourceNotFoundFault

DescribeEvents (Aktion)

Der AWS CLI-Name für diese API lautet:`describe-events`.

Gibt Ereignisse zu DB-Instances, DB-Sicherheitsgruppen, DB-Snapshots und DB-Parametergruppen in den vergangenen 14 Tagen zurück. Ereignisse, die für eine bestimmte DB-Instance, DB-Sicherheitsgruppe, DB-Parametergruppe oder einen bestimmten Datenbank-Snapshot spezifisch sind, erhalten Sie, indem Sie den Namen als Parameter angeben. Standardmäßig wird die letzte Stunde von Ereignissen zurückgegeben.

Anforderung

- Duration(in der CLI:`--duration`) — ein IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Die Anzahl der Minuten, in denen Ereignisse abgerufen werden sollen.

  Standard: 60
- EndTime(in der CLI:`--end-time`) — a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Offset von Mitternacht 1970-01-01).

  Das Ende des Zeitintervalls, für das Ereignisse abgerufen werden sollen, angegeben im ISO 8601-Format. Weitere Informationen zu ISO 8601 finden Sie auf der Wikipedia-Seite ISO8601.

  Beispiel: 2009-07-08T18:00Z

- EventCategories (in der CLI: `--event-categories`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Liste von Ereigniskategorien, die Benachrichtigungen für ein Abonnement für die Ereignisbenachrichtigung auslösen.

- Filters (in der CLI: `--filters`) – Ein Array von Filter Objekten.

  Dieser Parameter wird derzeit nicht unterstützt.

- Marker (in der CLI: `--marker`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Ein optionales Paginierungstoken, das in einer früheren DescribeEvents Anfrage bereitgestellt wurde. Wenn Sie diesen Parameter angeben, enthält die Antwort nur die Datensätze zwischen der Markierung und dem durch `MaxRecords` angegebenen Wert.

- MaxRecords(in der CLI:`--max-records`) — ein IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Die maximale Anzahl der in der Antwort zurückgegebenen Datensätze. Wenn mehrere Datensätze vorhanden sind, als der Wert `MaxRecords` angibt, ist ein Paginierungstoken mit dem Namen einer Markierung in der Antwort enthalten, sodass die verbleibenden Ergebnisse abgerufen werden können.

  Standard: 100

  Einschränkungen: Mindestwert 20, Höchstwert 100.

- SourceIdentifier (in der CLI: `--source-identifier`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  ID der Ereignisquelle, für die Ereignisse zurückgegeben werden. Wenn nicht angegeben, werden alle Quellen zur Antwort hinzugefügt.

  Einschränkungen:

  - Wenn SourceIdentifier angegeben, SourceType muss es auch angegeben werden.
  - Wenn der Quelltyp `DBInstance` ist, muss ein `DBInstanceIdentifier` angegeben werden.
  - Wenn der Quelltyp `DBSecurityGroup` ist, muss ein `DBSecurityGroupName` angegeben werden.

  - Wenn der Quelltyp `DBParameterGroup` ist, muss ein `DBParameterGroupName` angegeben werden.
  - Wenn der Quelltyp `DBSnapshot` ist, muss ein `DBSnapshotIdentifier` angegeben werden.
  - Darf nicht mit einem Bindestrich enden oder zwei aufeinanderfolgende Bindestriche enthalten.
- SourceType(in der CLI:`--source-type`) — a SourceType, vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

  Die Ereignisquelle zum Abrufen von Ereignissen. Wenn kein Wert angegeben ist, werden alle Ereignisse zurückgegeben.
- StartTime(in der CLI:`--start-time`) — a TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Offset von Mitternacht 1970-01-01).

  Der Beginn des Zeitintervalls, für das Ereignisse abgerufen werden sollen, angegeben im ISO 8601-Format. Weitere Informationen zu ISO 8601 finden Sie auf der Wikipedia-Seite ISO8601.

  Beispiel: 2009-07-08T18:00Z

Antwort

- Events – Ein Array mit Ereignis-Objekten.

  Eine Liste der the section called “Ereignis”-Instances.
- Marker – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Ein optionaler Paginierungstoken, der von einer vorherigen Ereignisanforderung bereitgestellt wird. Wenn Sie diesen Parameter angeben, enthält die Antwort nur die Datensätze zwischen der Markierung und dem durch `MaxRecords` angegebenen Wert.

DescribeEventCategories (Aktion)

> Der AWS CLI-Name für diese API lautet:`describe-event-categories`.

Zeigt eine Liste der Kategorien für alle Ereignisquelltypen oder – falls angegeben – für einen angegebenen Quelltyp an.

Anforderung

- Filters (in der CLI: `--filters`) – Ein Array von Filter Objekten.

Dieser Parameter wird derzeit nicht unterstützt.

- SourceType (in der CLI: `--source-type`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Typ der Quelle, die die Ereignisse generiert.

  Gültige Werte: db-instance | | db-parameter-group | db-security-group db-snapshot

Antwort

- EventCategoriesMapList – Ein Array mit EventCategoriesMap-Objekten.

  Eine Liste von Datentypen. EventCategoriesMap

Strukturen:

Ereignis (Struktur)

Dieser Datentyp wird als Antwortelement in der Aktion the section called “DescribeEvents” verwendet.

Felder

- Date— Dies ist ein TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als Offset von Mitternacht 1970-01-01).

  Legt das Datum und die Uhrzeit des Ereignisses fest.
- EventCategories – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Legt die Kategorie für das Ereignis fest.
- Message – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Stellt den Text dieses Ereignisses bereit.
- SourceArn – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Amazon-Ressourcenname (ARN) für das Ereignis.
- SourceIdentifier – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Stellt die Kennung für die Quelle des Ereignisses bereit.

- SourceType— Dies ist ein SourceType, vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

  Gibt den Quelltyp für dieses Ereignis an.

EventCategoriesMap (Struktur)

Enthält die Ergebnisse eines erfolgreichen Aufrufs der the section called “DescribeEventCategories”-Aktion.

Felder

- EventCategories – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Ereigniskategorien für den festgelegten Quelltyp
- SourceType – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Quelltyp, zum dem die zurückgegebenen Kategorien gehören

EventSubscription (Struktur)

Enthält die Ergebnisse eines erfolgreichen Aufrufs der the section called “DescribeEventSubscriptions”-Aktion.

Felder

- CustomerAwsId – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Das Amazon-Kundenkonto, das dem Abonnement für Ereignisbenachrichtigungen zugeordnet ist.
- CustSubscriptionId – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Abonnement-ID für die Ereignisbenachrichtigung.
- Enabled – Dies ist ein boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Ein boolescher Wert, der angibt, ob das Abonnement aktiviert ist. "True" zeigt an, dass das Abonnement aktiviert ist.
- EventCategoriesList – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Liste von Ereigniskategorien für das Abonnement für die Ereignisbenachrichtigung.

- EventSubscriptionArn – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Amazon-Ressourcenname (ARN) für das Ereignisabonnement.
- SnsTopicArn – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der ARN des Themas des Abonnements für die Ereignisbenachrichtigung.
- SourceIdsList – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Quellenliste IDs für das Abonnement für die Ereignisbenachrichtigung.
- SourceType – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Quelltyp für das Abonnement für die Ereignisbenachrichtigung.
- Status – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Status des Abonnements für die Ereignis-Benachrichtigung.

  Einschränkungen:

  Dabei kann es sich um eine der folgenden Optionen handeln: erstellen | ändern | löschen | aktiv | keine Erlaubnis | topic-not-exist

  Der Status "Keine Berechtigung" gibt an, dass Neptune keine Berechtigung mehr hat, an das SNS-Thema zu posten. Der Status "topic-not-exist" gibt an, dass das Thema nach der Erstellung des Abonnements gelöscht wurde.
- SubscriptionCreationTime – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Uhrzeit, zu der das Abonnement für die Ereignisbenachrichtigung erstellt wurde.

`EventSubscription` wird als Antwortelement verwendet für:

- CreateEventSubscription
- ModifyEventSubscription
- AddSourceIdentifierToSubscription
- RemoveSourceIdentifierFromSubscription
- DeleteEventSubscription

Anderer Neptune APIs

Aktionen:

- AddTagsToResource (Aktion)
- ListTagsForResource (Aktion)
- RemoveTagsFromResource (Aktion)
- ApplyPendingMaintenanceAction (Aktion)
- DescribePendingMaintenanceActions (Aktion)
- DBEngineVersionen beschreiben (Aktion)

Strukturen:

- DBEngineVersion (Struktur)
- EngineDefaults (Struktur)
- PendingMaintenanceAction (Struktur)
- ResourcePendingMaintenanceActions (Struktur)
- UpgradeTarget (Struktur)
- Tag (Struktur)

AddTagsToResource (Aktion)

> Der AWS CLI-Name für diese API lautet:`add-tags-to-resource`.

Fügt einer Amazon Neptune-Ressource Metadaten-Tags hinzu. Diese Tags können auch mit Kostenzuordnungsberichten verwendet werden, um Kosten im Zusammenhang mit Amazon Neptune-Ressourcen nachzuverfolgen, oder in einer Bedingungsanweisung einer IAM-Richtlinie für Amazon Neptune.

Anforderung

- ResourceName (in der CLI: `--resource-name`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Die Amazon Neptune-Ressource, der die Tags hinzugefügt werden. Dieser Wert ist ein Amazon-Ressourcenname (ARN). Weitere Informationen zum Erstellen eines ARN finden Sie unter Erstellen eines Amazon-Ressourcennamens (ARN).

- Tags (in der CLI: `--tags`) – Erforderlich: Ein Array von Markierung-Objekten.

  Die Tags, die der Amazon Neptune-Ressource zugewiesen werden sollen.

Antwort

- Keine Antwortparameter.

Fehler

- DBInstanceNotFoundFault
- DBSnapshotNotFoundFault
- DBClusterNotFoundFault

ListTagsForResource (Aktion)

Der AWS CLI-Name für diese API lautet:`list-tags-for-resource`.

Listet alle Tags in einer Amazon Neptune-Ressource auf.

Anforderung

- Filters (in der CLI: `--filters`) – Ein Array von Filter Objekten.

  Dieser Parameter wird derzeit nicht unterstützt.

- ResourceName (in der CLI: `--resource-name`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die aufzulistende Amazon Neptune-Ressource mit Tags. Dieser Wert ist ein Amazon-Ressourcenname (ARN). Weitere Informationen zum Erstellen eines ARN finden Sie unter Erstellen eines Amazon-Ressourcennamens (ARN).

Antwort

- TagList – Ein Array mit Markierung-Objekten.

  Liste der von der ListTagsForResource Operation zurückgegebenen Tags.

Fehler

- DBInstanceNotFoundFault
- DBSnapshotNotFoundFault
- DBClusterNotFoundFault

RemoveTagsFromResource (Aktion)

> Der AWS CLI-Name für diese API lautet:`remove-tags-from-resource`.

Entfernt Metadaten-Tags aus einer Amazon Neptune-Ressource.

Anforderung

- ResourceName (in der CLI: `--resource-name`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Amazon Neptune-Ressource, von der die Tags entfernt werden. Dieser Wert ist ein Amazon-Ressourcenname (ARN). Weitere Informationen zum Erstellen eines ARN finden Sie unter Erstellen eines Amazon-Ressourcennamens (ARN).
- TagKeys (in der CLI: `--tag-keys`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Tag-Schlüssel (Name) des zu entfernenden Tags.

Antwort

- Keine Antwortparameter.

Fehler

- DBInstanceNotFoundFault
- DBSnapshotNotFoundFault

- DBClusterNotFoundFault

ApplyPendingMaintenanceAction (Aktion)

Der AWS CLI-Name für diese API lautet:`apply-pending-maintenance-action`.

Wendet eine ausstehende Wartungsaktion auf eine Ressource an (z. B. eine DB-Instance).

Anforderung

- ApplyAction (in der CLI: `--apply-action`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die anstehende Wartungsaktion, die auf diese Ressource angewendet werden soll.

  Zulässige Werte: `system-update`, `db-upgrade`
- OptInType (in der CLI: `--opt-in-type`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Ein Wert, der die Art der Opt-In-Anfrage angibt oder eine Opt-In-Anfrage rückgängig macht. Eine Opt-in-Anfrage vom Typ `immediate` kann nicht rückgängig gemacht werden.

  Zulässige Werte:
  - `immediate` – Die Wartungsmaßnahme sofort anwenden.
  - `next-maintenance` – Die Wartungsaktion im nächsten Wartungsfenster für die Ressource anwenden.
  - `undo-opt-in` – Alle bestehenden `next-maintenance`-Opt-In-Anfragen stornieren.
- ResourceIdentifier (in der CLI: `--resource-identifier`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Amazon-Ressourcenname (ARN) der Ressource, auf die sich die anstehende Wartungsaktion bezieht. Weitere Informationen zum Erstellen eines ARN finden Sie unter Erstellen eines Amazon-Ressourcennamens (ARN).

Antwort

Beschreibt die ausstehenden Wartungsaktionen für eine Ressource.

- PendingMaintenanceActionDetails – Ein Array mit PendingMaintenanceAction-Objekten.

Eine Liste mit Details zu den ausstehenden Wartungsaktionen für die Ressource.

- ResourceIdentifier – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der ARN der Ressource, die ausstehende Wartungsaktionen hat.

Fehler

- ResourceNotFoundFault

DescribePendingMaintenanceActions (Aktion)

Der AWS CLI-Name für diese API lautet:`describe-pending-maintenance-actions`.

Gibt eine Liste von Ressourcen (z. B. DB-Instances) zurück, für die mindestens eine Wartungsaktion aussteht.

Anforderung

- Filters (in der CLI: `--filters`) – Ein Array von Filter Objekten.

  Ein Filter, der eine oder mehrere Ressourcen angibt, für die ausstehende Wartungsaktionen zurückgegeben werden sollen.

  Unterstützte Filter:

  - `db-cluster-id`- Akzeptiert DB-Cluster-Identifikatoren und DB-Cluster-Amazon-Ressourcennamen (ARNs). Die Ergebnisliste wird nur ausstehende Wartungsaktionen für die anhand dieser ARNs Kriterien identifizierten DB-Cluster enthalten.
  - `db-instance-id`- Akzeptiert DB-Instance-Identifikatoren und DB-Instances ARNs. Die Ergebnisliste enthält nur ausstehende Wartungsaktionen für die DB-Instances, die durch diese ARNs identifiziert wurden.
- Marker (in der CLI: `--marker`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Ein optionales Paginierungstoken, das von einer vorherigen `DescribePendingMaintenanceActions`-Anforderung bereitgestellt wird. Wenn dieser Parameter angegeben ist, enthält die Antwort nur Datensätze jenseits der Markierung bis zu einer Anzahl von Datensätzen, die durch `MaxRecords` festgelegt wurde.

- MaxRecords(in der CLI:`--max-records`) — ein IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Die maximale Anzahl der in der Antwort zurückgegebenen Datensätze. Wenn mehrere Datensätze vorhanden sind, als der Wert `MaxRecords` angibt, ist ein Paginierungstoken mit dem Namen einer Markierung in der Antwort enthalten, sodass die verbleibenden Ergebnisse abgerufen werden können.

  Standard: 100

  Einschränkungen: Mindestwert 20, Höchstwert 100.
- ResourceIdentifier (in der CLI: `--resource-identifier`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der ARN einer Ressource, für die ausstehende Wartungsaktionen zurückgegeben werden sollen.

Antwort

- Marker – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Ein optionales Paginierungstoken, das von einer vorherigen `DescribePendingMaintenanceActions`-Anforderung bereitgestellt wird. Wenn dieser Parameter angegeben ist, enthält die Antwort nur Datensätze jenseits der Markierung bis zu einer Anzahl von Datensätzen, die durch `MaxRecords` festgelegt wurde.
- PendingMaintenanceActions – Ein Array mit ResourcePendingMaintenanceActions-Objekten.

  Eine Liste ausstehender Wartungsaktionen für die Ressource.

Fehler

- ResourceNotFoundFault

DBEngineVersionen beschreiben (Aktion)

Der AWS CLI-Name für diese API lautet:`describe-db-engine-versions`.

Gibt eine Liste der verfügbaren DB-Engines zurück.

Anforderung

- DBParameterGroupFamily (in der CLI: `--db-parameter-group-family`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Name einer spezifischen DB-Parametergruppenfamilie, für die Details zurückgegeben werden sollen.

  Einschränkungen:

  - Falls angegeben, muss es mit einem vorhandenen übereinstimmen DBParameterGroupFamily.
- DefaultOnly (in der CLI: `--default-only`) – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Gibt an, dass nur die standardmäßige Version der angegebenen Engine oder Kombination aus Engine und Hauptversion zurückgegeben wird.
- Engine (in der CLI: `--engine`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die zurückzugebende Datenbank-Engine.
- EngineVersion (in der CLI: `--engine-version`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die zurückzugebende Datenbank-Engine-Version.

  Beispiel: `5.1.49`
- Filters (in der CLI: `--filters`) – Ein Array von Filter Objekten.

  Wird derzeit nicht unterstützt.
- ListSupportedCharacterSets(in der CLI:`--list-supported-character-sets`) — a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Wenn dieser Parameter angegeben ist und die angeforderte Engine den Parameter `CharacterSetName` für `CreateDBInstance` unterstützt, enthält die Antwort eine Liste der unterstützten Zeichensätze für jede Engine-Version.
- ListSupportedTimezones(in der CLI:`--list-supported-timezones`) — a BooleanOptional, vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

  Wenn dieser Parameter angegeben ist und die angeforderte Engine den Parameter `TimeZone` für `CreateDBInstance` unterstützt, enthält die Antwort eine Liste der unterstützten Zeitzonen für jede Engine-Version.

- Marker (in der CLI: `--marker`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Ein optionales Paginierungstoken, das von einer vorherigen Anforderung bereitgestellt wird. Wenn Sie diesen Parameter angeben, enthält die Antwort nur die Datensätze zwischen der Markierung und dem durch `MaxRecords` angegebenen Wert.
- MaxRecords(in der CLI:`--max-records`) — ein IntegerOptional, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Die maximale Anzahl der in der Antwort zurückgegebenen Datensätze. Wenn mehr als der Wert `MaxRecords` verfügbar ist, wird ein als Markierung bezeichneter Paginierungs-Token zur Antwort hinzugefügt, damit die folgenden Ergebnisse abgerufen werden können.

  Standard: 100

  Einschränkungen: Mindestwert 20, Höchstwert 100.

Antwort

- DBEngineVersions – Ein Array mit DBEngineVersion-Objekten.

  Eine Liste mit `DBEngineVersion`-Elementen.
- Marker – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Ein optionales Paginierungstoken, das von einer vorherigen Anforderung bereitgestellt wird. Wenn Sie diesen Parameter angeben, enthält die Antwort nur die Datensätze zwischen der Markierung und dem durch `MaxRecords` angegebenen Wert.

Strukturen:

DBEngineVersion (Struktur)

Dieser Datentyp wird als Antwortelement in der Aktion the section called “DBEngineVersionen beschreiben” verwendet.

Felder

- DBEngineDescription – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Die Beschreibung der Datenbank-Engine.

- DBEngineVersionDescription – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Beschreibung der Datenbank-Engine-Version.

- DBParameterGroupFamily – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Name der Familie der DB-Parametergruppe für die Datenbank-Engine.

- Engine – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Name der Datenbank-Engine.

- EngineVersion – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Versionsnummer des Datenbank-Engines.

- ExportableLogTypes – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Protokolltypen, die die Datenbank-Engine für den Export in CloudWatch Logs zur Verfügung stellt.

- SupportedTimezones – Dies ist ein Array von Zeitzone-Objekten.

  Eine Liste der von dieser Engine für den Parameter `Timezone` der Aktion `CreateDBInstance` unterstützten Zeitzonen.

- SupportsGlobalDatabases – Dies ist ein boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Ein Wert, der angibt, ob Sie globale Aurora-Datenbanken mit einer bestimmten DB-Engine-Version verwenden können.

- SupportsLogExportsToCloudwatchLogs – Dies ist ein boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Ein Wert, der angibt, ob die Engine-Version den Export der in ExportableLogTypes to CloudWatch Logs angegebenen Protokolltypen unterstützt.

- SupportsReadReplica – Dies ist ein boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Gibt an, ob die Datenbank-Engine-Version Read Replicas unterstützt.

- ValidUpgradeTarget – Dies ist ein Array von UpgradeTarget-Objekten.

  Eine Liste der Engine-Versionen, auf die diese Datenbank-Engine-Version aktualisiert werden kann.

EngineDefaults (Struktur)

Enthält das Ergebnis eines erfolgreichen Aufrufs der the section called "DescribeEngineDefaultParameters"-Aktion.

Felder

- DBParameterGroupFamily – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Namen der DB-Parametergruppenfamilie an, auf die die Standard-Parameter der Engine angewendet werden.
- Marker – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Ein optionales Paginierungstoken, das in einer früheren EngineDefaults Anfrage bereitgestellt wurde. Wenn Sie diesen Parameter angeben, enthält die Antwort nur die Datensätze zwischen der Markierung und dem durch `MaxRecords` angegebenen Wert.
- Parameters – Dies ist ein Array von Parameter-Objekten.

  Enthält eine Liste der Standardparameter der Engine.

`EngineDefaults` wird als Antwortelement verwendet für:

- DescribeEngineDefaultParameters
- DescribeEngineDefaultClusterParameters

PendingMaintenanceAction (Struktur)

Stellt Informationen über eine ausstehende Wartungsaktion für eine Ressource bereit.

Felder

- Action – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Der Typ der ausstehenden Wartungsaktion, die für die Ressource verfügbar ist.

- AutoAppliedAfterDate— Dies ist ein TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als ein Offset von Mitternacht 1970-01-01).

  Das Datum des Wartungsfensters, in dem die Aktion angewendet wird. Die Wartungsaktion wird während ihres ersten Wartungsfensters nach diesem Datum auf die Ressource angewendet. Wenn dieses Datum angegeben ist, werden alle `next-maintenance`-Opt-In-Anfragen ignoriert.
- CurrentApplyDate— Dies ist ein TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als ein Offset von Mitternacht 1970-01-01).

  Das Datum des Inkrafttretens, wenn die ausstehende Wartungsaktion auf die Ressource angewendet wird. Dieses Datum berücksichtigt Opt-In-Anfragen, die von der the section called “ApplyPendingMaintenanceAction”-API, von `AutoAppliedAfterDate` und von `ForcedApplyDate` erhalten wurden. Dieser Wert ist leer, wenn eine Opt-In-Anforderung nicht empfangen wurde und nichts für `AutoAppliedAfterDate` oder `ForcedApplyDate` angegeben wurde.
- Description – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Beschreibung, die weitere Details zu der Wartungsaktion bereitstellt.
- ForcedApplyDate— Dies ist ein TStamp, vom Typ: `timestamp` (ein Zeitpunkt, allgemein definiert als ein Offset von Mitternacht 1970-01-01).

  Das Datum, an dem die Wartungsaktion automatisch angewendet wird. Die Wartungsaktion wird ungeachtet des Wartungsfensters für die Ressource an diesem Datum auf die Ressource angewendet. Wenn dieses Datum angegeben ist, werden alle `immediate`-Opt-In-Anfragen ignoriert.
- OptInStatus – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt den Typ der Opt-in-Anforderung an, die für die Ressource empfangen wurde.

ResourcePendingMaintenanceActions (Struktur)

Beschreibt die ausstehenden Wartungsaktionen für eine Ressource.

Felder

- PendingMaintenanceActionDetails – Dies ist ein Array von PendingMaintenanceAction-Objekten.

Eine Liste mit Details zu den ausstehenden Wartungsaktionen für die Ressource.

- ResourceIdentifier – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der ARN der Ressource, die ausstehende Wartungsaktionen hat.

`ResourcePendingMaintenanceActions` wird als Antwortelement verwendet für:

- ApplyPendingMaintenanceAction

UpgradeTarget (Struktur)

Die Version der Datenbank-Engine, auf die eine DB-Instance aktualisiert werden kann.

Felder

- AutoUpgrade – Dies ist ein boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Ein Wert, der angibt, ob die Zielversion auf Quell-DB-Instances angewendet wird, die auf true AutoMinorVersionUpgrade gesetzt sind.
- Description – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Version der Datenbank-Engine, auf die eine DB-Instance aktualisiert werden kann.
- Engine – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Name der Upgrade-Zieldatenbank-Engine.
- EngineVersion – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Versionsnummer der Upgrade-Zieldatenbank-Engine.
- IsMajorVersionUpgrade – Dies ist ein boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Ein Wert, der angibt, ob eine Datenbank-Engine auf eine Hauptversion aktualisiert wird.
- SupportsGlobalDatabases— Dies ist ein BooleanOptional Wert vom Typ: `boolean` (ein boolescher Wert (wahr oder falsch)).

Ein Wert, der angibt, ob Sie globale Neptune-Datenbanken mit der Version der Ziel-Engine verwenden können.

Tag (Struktur)

Einer Amazon Neptune-Ressource zugewiesene Metadaten, die aus einem Schlüssel-Wert-Paar bestehen.

Felder

- Key – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Ein Schlüssel ist der erforderliche Name des Tags. Der Zeichenfolgenwert kann aus 1 bis 128 Unicode-Zeichen bestehen. Ihm darf kein „`aws:`“ oder „`rds:`“ als Präfix vorangestellt werden. Die Zeichenfolge darf nur Unicode-Zeichen, Ziffern, Leerzeichen sowie „_“, „.“, „/“, „=“, „+“, „-“ enthalten (Java-Regex: „^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)“).
- Value – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Ein Wert ist der optionale Wert des Tags. Der Zeichenfolgenwert kann aus 1 bis 128 Unicode-Zeichen bestehen. Ihm darf kein „`aws:`“ oder „`rds:`“ als Präfix vorangestellt werden. Die Zeichenfolge darf nur Unicode-Zeichen, Ziffern, Leerzeichen sowie „_“, „.“, „/“, „=“, „+“, „-“ enthalten (Java-Regex: „^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)“).

Häufige Neptune-Datentypen

Strukturen:

- AvailabilityZone (Struktur)
- DBSecurityGroupMembership (Struktur)
- DomainMembership (Struktur)
- DoubleRange (Struktur)
- Endpunkt (Struktur)
- Filter (Struktur)
- Bereich (Struktur)
- ServerlessV2 ScalingConfiguration (Struktur)

- ScalingConfigurationInfo ServerlessV2 (Struktur)
- Zeitzone (Struktur)
- VpcSecurityGroupMembership (Struktur)

AvailabilityZone (Struktur)

Gibt eine Availability Zone an.

Felder

- Name – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Name der Availability Zone.

DBSecurityGroupMembership (Struktur)

Gibt die Mitgliedschaft in einer bestimmten DB-Sicherheitsgruppe an.

Felder

- DBSecurityGroupName – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Name der DB-Sicherheitsgruppe.
- Status – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Status der DB-Sicherheitsgruppe.

DomainMembership (Struktur)

Ein Active Directory Domain-Mitgliedschaftsdatensatz, der mit einer DB-Instance verknüpft ist.

Felder

- Domain – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Kennung der Active Directory Domain.
- FQDN – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Der vollqualifizierte Domänenname der Active Directory Domain.

- IAMRoleName – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Name der IAM-Rolle, die verwendet werden soll, wenn API-Aufrufe an den Directory Service ausgerichtet werden.
- Status – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Status der Active Directory Domain-Mitgliedschaft der DB-Instance, wie z. B. joined, pending-join, failed usw.

DoubleRange (Struktur)

Ein Bereich von doppelten Werten.

Felder

- From – Dies ist ein Double-Wert vom Typ: `double` (eine IEEE 754-Gleitkommazahl mit doppelter Genauigkeit).

  Der minimale Wert im Bereich.
- To – Dies ist ein Double-Wert vom Typ: `double` (eine IEEE 754-Gleitkommazahl mit doppelter Genauigkeit).

  Der maximale Wert im Bereich.

Endpunkt (Struktur)

Gibt einen Verbindungsendpunkt an.

Informationen zur Datenstruktur, die Amazon Neptune-DB-Cluster-Endpunkte repräsentiert, finden Sie unter `DBClusterEndpoint`.

Felder

- Address – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Gibt die DNS-Adresse der DB-Instance an.
- HostedZoneId – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Gibt die ID an, die Amazon Route 53 zuweist, wenn Sie eine gehostete Zone erstellen.

- Port – Dies ist eine Ganzzahl vom Typ `integer` (32-Bit-Ganzzahl mit Vorzeichen).

  Gibt die Portnummer an, die von der Datenbank-Engine überwacht wird.

Filter (Struktur)

Dieser Typ wird derzeit nicht unterstützt.

Felder

- Name – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Dieser Parameter wird derzeit nicht unterstützt.
- Values – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Dieser Parameter wird derzeit nicht unterstützt.

Bereich (Struktur)

Ein Bereich von Ganzzahlwerten.

Felder

- From – Dies ist eine Ganzzahl vom Typ `integer` (32-Bit-Ganzzahl mit Vorzeichen).

  Der minimale Wert im Bereich.
- Step— Dies ist eine IntegerOptional 32-Bit-Ganzzahl vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen).

  Der Schrittwert für den Bereich. Wenn Sie zum Beispiel über einen Bereich von 5.000 bis 10.000 verfügen, mit einem Schrittwert von 1.000, beginnen die gültigen Werte bei 5.000 und erhöhen sich um 1.000. Auch wenn 7.500 im Bereich liegt, ist es kein gültiger Wert für den Bereich. Die gültigen Werte sind 5.000, 6.000, 7.000 und 8.000...
- To – Dies ist eine Ganzzahl vom Typ `integer` (32-Bit-Ganzzahl mit Vorzeichen).

  Der maximale Wert im Bereich.

ServerlessV2 ScalingConfiguration (Struktur)

Enthält die Skalierungskonfiguration eines Neptune Serverless DB-Clusters.

Weitere Informationen finden Sie unter Verwendung von Amazon Neptune Serverless im Amazon Neptune-Benutzerhandbuch.

Felder

- MaxCapacity— Dies ist eine DoubleOptional, vom Typ: `double` (eine IEEE 754-Gleitkommazahl mit doppelter Genauigkeit).

  Die maximale Anzahl von Neptune-Kapazitätseinheiten (NCUs) für eine DB-Instance in einem Neptune Serverless-Cluster. Geben Sie NCU-Werte in Halbschritten an, z. B. 40, 40,5, 41 usw.
- MinCapacity— Dies ist ein Typ: `double` (eine DoubleOptional IEEE 754-Gleitkommazahl mit doppelter Genauigkeit).

  Die Mindestanzahl von Neptune-Kapazitätseinheiten (NCUs) für eine DB-Instance in einem Neptune Serverless-Cluster. Geben Sie NCU-Werte in Halbschritten an, z. B. 40, 40,5, 41 usw.

ScalingConfigurationInfo ServerlessV2 (Struktur)

Zeigt die Skalierungskonfiguration für einen Neptune Serverless DB-Cluster.

Weitere Informationen finden Sie unter Verwendung von Amazon Neptune Serverless im Amazon Neptune-Benutzerhandbuch.

Felder

- MaxCapacity— Dies ist eine DoubleOptional, vom Typ: `double` (eine IEEE 754-Gleitkommazahl mit doppelter Genauigkeit).

  Die maximale Anzahl von Neptune-Kapazitätseinheiten (NCUs) für eine DB-Instance in einem Neptune Serverless-Cluster. Geben Sie NCU-Werte in Halbschritten an, z. B. 40, 40,5, 41 usw.
- MinCapacity— Dies ist ein Typ: `double` (eine DoubleOptional IEEE 754-Gleitkommazahl mit doppelter Genauigkeit).

  Die Mindestanzahl von Neptune-Kapazitätseinheiten (NCUs) für eine DB-Instance in einem Neptune Serverless-Cluster. Geben Sie NCU-Werte in Halbschritten an, z. B. 40, 40,5, 41 usw.

Zeitzone (Struktur)

Eine Zeitzone, die mit einer the section called “DBInstance” verknüpft ist.

Felder

- TimezoneName – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Name der Zeitzone.

VpcSecurityGroupMembership (Struktur)

Dieser Datentyp wird als Antwortelement für Abfragen in VPC-Sicherheitsgruppenmitgliedschaften verwendet.

Felder

- Status – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Status der VPC-Sicherheitsgruppe.
- VpcSecurityGroupId – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Name der VPC-Sicherheitsgruppe.

Individuelle Neptun-Ausnahmen APIs

Ausnahmen:

- AuthorizationAlreadyExistsFault (Struktur)
- AuthorizationNotFoundFault (Struktur)
- AuthorizationQuotaExceededFault (Struktur)
- CertificateNotFoundFault (Struktur)
- DBClusterAlreadyExistsFault (Struktur)
- DBClusterNotFoundFault (Struktur)
- DBClusterParameterGroupNotFoundFault (Struktur)
- DBClusterQuotaExceededFault (Struktur)

- DBClusterRoleAlreadyExistsFault (Struktur)
- DBClusterRoleNotFoundFault (Struktur)
- DBClusterRoleQuotaExceededFault (Struktur)
- DBClusterSnapshotAlreadyExistsFault (Struktur)
- DBClusterSnapshotNotFoundFault (Struktur)
- DBInstanceAlreadyExistsFault (Struktur)
- DBInstanceNotFoundFault (Struktur)
- DBLogFileNotFoundFault (Struktur)
- DBParameterGroupAlreadyExistsFault (Struktur)
- DBParameterGroupNotFoundFault (Struktur)
- DBParameterGroupQuotaExceededFault (Struktur)
- DBSecurityGroupAlreadyExistsFault (Struktur)
- DBSecurityGroupNotFoundFault (Struktur)
- DBSecurityGroupNotSupportedFault (Struktur)
- DBSecurityGroupQuotaExceededFault (Struktur)
- DBSnapshotAlreadyExistsFault (Struktur)
- DBSnapshotNotFoundFault (Struktur)
- DBSubnetGroupAlreadyExistsFault (Struktur)
- DBSubnetGroupDoesNotCoverEnoughAZs (Struktur)
- DBSubnetGroupNotAllowedFault (Struktur)
- DBSubnetGroupNotFoundFault (Struktur)
- DBSubnetGroupQuotaExceededFault (Struktur)
- DBSubnetQuotaExceededFault (Struktur)
- DBUpgradeDependencyFailureFault (Struktur)
- DomainNotFoundFault (Struktur)
- EventSubscriptionQuotaExceededFault (Struktur)
- GlobalClusterAlreadyExistsFault (Struktur)
- GlobalClusterNotFoundFault (Struktur)
- GlobalClusterQuotaExceededFault (Struktur)

- InstanceQuotaExceededFault (Struktur)
- Ungenügend DBCluster CapacityFault (Struktur)
- Ungenügend DBInstance CapacityFault (Struktur)
- InsufficientStorageClusterCapacityFault (Struktur)
- Ungültig DBCluster EndpointStateFault (Struktur)
- Ungültig DBCluster SnapshotStateFault (Struktur)
- Ungültig DBCluster StateFault (Struktur)
- Ungültig DBInstance StateFault (Struktur)
- Ungültig DBParameter GroupStateFault (Struktur)
- Ungültig DBSecurity GroupStateFault (Struktur)
- Ungültig DBSnapshot StateFault (Struktur)
- Ungültig DBSubnet GroupFault (Struktur)
- Ungültig DBSubnet GroupStateFault (Struktur)
- Ungültig DBSubnet StateFault (Struktur)
- InvalidEventSubscriptionStateFault (Struktur)
- InvalidGlobalClusterStateFault (Struktur)
- InvalidOptionGroupStateFault (Struktur)
- InvalidRestoreFault (Struktur)
- InvalidSubnet (Struktur)
- Ungültig VPCNetwork StateFault (Struktur)
- KMSKeyNotAccessibleFault (Struktur)
- OptionGroupNotFoundFault (Struktur)
- PointInTimeRestoreNotEnabledFault (Struktur)
- ProvisionedIopsNotAvailableInAZFault (Struktur)
- ResourceNotFoundFault (Struktur)
- SNSInvalidTopicFault (Struktur)
- SNSNoAuthorizationFault (Struktur)
- SNSTopicArnNotFoundFault (Struktur)
- SharedSnapshotQuotaExceededFault (Struktur)

- SnapshotQuotaExceededFault (Struktur)
- SourceNotFoundFault (Struktur)
- StorageQuotaExceededFault (Struktur)
- StorageTypeNotSupportedFault (Struktur)
- SubnetAlreadyInUse (Struktur)
- SubscriptionAlreadyExistFault (Struktur)
- SubscriptionCategoryNotFoundFault (Struktur)
- SubscriptionNotFoundFault (Struktur)

AuthorizationAlreadyExistsFault (Struktur)

Zurückgegebener HTTP-Statuscode: 400.

Die angegebene CIDRIP- oder EC2 Sicherheitsgruppe ist bereits für die angegebene DB-Sicherheitsgruppe autorisiert.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer Beschreibung der Problemdetails.

AuthorizationNotFoundFault (Struktur)

Zurückgegebener HTTP-Statuscode: 404.

Die angegebene CIDRIP- oder EC2 Sicherheitsgruppe ist für die angegebene DB-Sicherheitsgruppe nicht autorisiert.

Neptune kann nicht auch über IAM autorisiert werden, um erforderliche Aktionen in Ihrem Namen durchzuführen.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

Eine Meldung mit einer Beschreibung der Problemdetails.

AuthorizationQuotaExceededFault (Struktur)

Zurückgegebener HTTP-Statuscode: 400.

Das Autorisierungs-Kontingent der DB-Sicherheitsgruppe wurde erreicht.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer Beschreibung der Problemdetails.

CertificateNotFoundFault (Struktur)

Zurückgegebener HTTP-Statuscode: 404.

CertificateIdentifierbezieht sich nicht auf ein vorhandenes Zertifikat.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer Beschreibung der Problemdetails.

DBClusterAlreadyExistsFault (Struktur)

Zurückgegebener HTTP-Statuscode: 400.

Der Benutzer verfügt bereits über ein DB-Cluster mit einer bestimmten ID.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer Beschreibung der Problemdetails.

DBClusterNotFoundFault (Struktur)

Zurückgegebener HTTP-Statuscode: 404.

DBClusterIdentifier bezieht sich nicht auf einen vorhandenen DB-Cluster.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer Beschreibung der Problemdetails.

DBClusterParameterGroupNotFoundFault (Struktur)

Zurückgegebener HTTP-Statuscode: 404.

DBClusterParameterGroupNamebezieht sich nicht auf eine bestehende DB-Cluster-Parametergruppe.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer Beschreibung der Problemdetails.

DBClusterQuotaExceededFault (Struktur)

Zurückgegebener HTTP-Statuscode: 403.

Der Benutzer versucht, einen neuen DB-Cluster zu erstellen, und der Benutzer hat bereits das maximal zulässige DB-Cluster-Kontingent erreicht.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer Beschreibung der Problemdetails.

DBClusterRoleAlreadyExistsFault (Struktur)

Zurückgegebener HTTP-Statuscode: 400.

Der angegebene Amazon-Ressourcenname (ARN) der IAM-Rolle ist dem angegebenen DB-Cluster bereits zugeordnet.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer Beschreibung der Problemdetails.

DBClusterRoleNotFoundFault (Struktur)

Zurückgegebener HTTP-Statuscode: 404.

Der angegebene Amazon-Ressourcenname (ARN) der IAM-Rolle ist dem angegebenen DB-Cluster nicht zugeordnet.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer Beschreibung der Problemdetails.

DBClusterRoleQuotaExceededFault (Struktur)

Zurückgegebener HTTP-Statuscode: 400.

Sie haben die maximale Anzahl an IAM-Rollen überschritten, die dem angegebenen DB-Cluster zugeordnet werden können.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

Eine Meldung mit einer Beschreibung der Problemdetails.

DBClusterSnapshotAlreadyExistsFault (Struktur)

Zurückgegebener HTTP-Statuscode: 400.

Der Benutzer verfügt bereits über einen DB-Cluster-Snapshot mit einer bestimmten ID.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer Beschreibung der Problemdetails.

DBClusterSnapshotNotFoundFault (Struktur)

Zurückgegebener HTTP-Statuscode: 404.

DBClusterSnapshotIdentifierbezieht sich nicht auf einen vorhandenen DB-Cluster-Snapshot.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer Beschreibung der Problemdetails.

DBInstanceAlreadyExistsFault (Struktur)

Zurückgegebener HTTP-Statuscode: 400.

Der Benutzer verfügt bereits über eine DB-Instance mit einer bestimmten ID.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

Eine Meldung mit einer Beschreibung der Problemdetails.

DBInstanceNotFoundFault (Struktur)

Zurückgegebener HTTP-Statuscode: 404.

DBInstanceIdentifier bezieht sich nicht auf eine bestehende DB-Instance.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer Beschreibung der Problemdetails.

DBLogFileNotFoundFault (Struktur)

Zurückgegebener HTTP-Statuscode: 404.

LogFileNamebezieht sich nicht auf eine bestehende DB-Protokolldatei.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer Beschreibung der Problemdetails.

DBParameterGroupAlreadyExistsFault (Struktur)

Zurückgegebener HTTP-Statuscode: 400.

Eine DB-Parametergruppe mit demselben Namen ist bereits vorhanden.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

Eine Meldung mit einer Beschreibung der Problemdetails.

DBParameterGroupNotFoundFault (Struktur)

Zurückgegebener HTTP-Statuscode: 404.

DBParameterGroupNamebezieht sich nicht auf eine bestehende DB-Parametergruppe.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer Beschreibung der Problemdetails.

DBParameterGroupQuotaExceededFault (Struktur)

Zurückgegebener HTTP-Statuscode: 400.

Die Anfrage würde dazu führen, dass der Benutzer die zulässige Anzahl an DB-Parametergruppen überschreitet.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer Beschreibung der Problemdetails.

DBSecurityGroupAlreadyExistsFault (Struktur)

Zurückgegebener HTTP-Statuscode: 400.

Eine DB-Sicherheitsgruppe mit dem in angegebenen Namen ist DBSecurityGroupNamebereits vorhanden.

Felder

- message— Dies ist eine ExceptionMessage, vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

Eine Meldung mit einer Beschreibung der Problemdetails.

DBSecurityGroupNotFoundFault (Struktur)

Zurückgegebener HTTP-Statuscode: 404.

DBSecurityGroupNamebezieht sich nicht auf eine bestehende DB-Sicherheitsgruppe.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer Beschreibung der Problemdetails.

DBSecurityGroupNotSupportedFault (Struktur)

Zurückgegebener HTTP-Statuscode: 400.

Eine DB-Sicherheitsgruppe ist für diese Aktion nicht zulässig.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer Beschreibung der Problemdetails.

DBSecurityGroupQuotaExceededFault (Struktur)

Zurückgegebener HTTP-Statuscode: 400.

Die Anfrage würde dazu führen, dass der Benutzer die zulässige Anzahl an DB-Sicherheitsgruppen überschreitet.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

Eine Meldung mit einer Beschreibung der Problemdetails.

DBSnapshotAlreadyExistsFault (Struktur)

Zurückgegebener HTTP-Statuscode: 400.

DBSnapshotDer Identifier wird bereits von einem vorhandenen Snapshot verwendet.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer Beschreibung der Problemdetails.

DBSnapshotNotFoundFault (Struktur)

Zurückgegebener HTTP-Statuscode: 404.

DBSnapshotDer Identifier bezieht sich nicht auf einen vorhandenen DB-Snapshot.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer Beschreibung der Problemdetails.

DBSubnetGroupAlreadyExistsFault (Struktur)

Zurückgegebener HTTP-Statuscode: 400.

DBSubnetGroupNamewird bereits von einer vorhandenen DB-Subnetzgruppe verwendet.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer Beschreibung der Problemdetails.

DBSubnetGroupDoesNotCoverEnoughAZs (Struktur)

Zurückgegebener HTTP-Statuscode: 400.

Subnetze in der DB-Subnetzgruppe sollten mindestens zwei Availability Zones abdecken, es sei denn, es gibt nur eine Availability Zone.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer Beschreibung der Problemdetails.

DBSubnetGroupNotAllowedFault (Struktur)

Zurückgegebener HTTP-Statuscode: 400.

Gibt an, dass die DBSubnet Gruppe beim Erstellen von Read Replicas, die sich in derselben Region wie die Quell-Instance befinden, nicht angegeben werden sollte.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer Beschreibung der Problemdetails.

DBSubnetGroupNotFoundFault (Struktur)

Zurückgegebener HTTP-Statuscode: 404.

DBSubnetGroupNamebezieht sich nicht auf eine bestehende DB-Subnetzgruppe.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer Beschreibung der Problemdetails.

DBSubnetGroupQuotaExceededFault (Struktur)

Zurückgegebener HTTP-Statuscode: 400.

Die Anfrage würde dazu führen, dass der Benutzer die zulässige Anzahl an DB-Subnetzgruppen überschreitet.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer Beschreibung der Problemdetails.

DBSubnetQuotaExceededFault (Struktur)

Zurückgegebener HTTP-Statuscode: 400.

Die Anfrage würde dazu führen, dass der Benutzer die zulässige Anzahl an Subnetzen in einer DB-Subnetzgruppen überschreitet.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer Beschreibung der Problemdetails.

DBUpgradeDependencyFailureFault (Struktur)

Zurückgegebener HTTP-Statuscode: 400.

Das DB-Upgrade ist fehlgeschlagen, da eine Ressource, von der die DB abhängig ist, nicht geändert werden konnte.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer Beschreibung der Problemdetails.

DomainNotFoundFault (Struktur)

Zurückgegebener HTTP-Statuscode: 404.

Domain (Domäne) bezieht sich nicht auf eine vorhandene Active Directory-Domäne.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer Beschreibung der Problemdetails.

EventSubscriptionQuotaExceededFault (Struktur)

Zurückgegebener HTTP-Statuscode: 400.

Sie haben die maximale Anzahl von Ereignissen erreicht, die Sie abonnieren können.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer Beschreibung der Problemdetails.

GlobalClusterAlreadyExistsFault (Struktur)

Zurückgegebener HTTP-Statuscode: 400.

`GlobalClusterIdentifier` ist bereits vorhanden. Wählen Sie eine neue globale Datenbank-ID (eindeutiger Name), um einen neuen globalen Datenbank-Cluster zu erstellen.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer Beschreibung der Problemdetails.

GlobalClusterNotFoundFault (Struktur)

Zurückgegebener HTTP-Statuscode: 404.

`GlobalClusterIdentifier` bezieht sich nicht auf einen vorhandenen globalen Datenbank-Cluster.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer Beschreibung der Problemdetails.

GlobalClusterQuotaExceededFault (Struktur)

Zurückgegebener HTTP-Statuscode: 400.

Die Anzahl der globalen Datenbank-Cluster für dieses Konto hat bereits den zulässigen Höchstwert erreicht.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer Beschreibung der Problemdetails.

InstanceQuotaExceededFault (Struktur)

Zurückgegebener HTTP-Statuscode: 400.

Die Anfrage würde dazu führen, dass der Benutzer die zulässige Anzahl an DB-Instances überschreitet.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

Eine Meldung mit einer Beschreibung der Problemdetails.

Ungenügend DBCluster CapacityFault (Struktur)

Zurückgegebener HTTP-Statuscode: 403.

Der DB-Cluster hat nicht genügend Kapazität für die aktuelle Operation.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer Beschreibung der Problemdetails.

Ungenügend DBInstance CapacityFault (Struktur)

Zurückgegebener HTTP-Statuscode: 400.

Die angegebene DB-Instance-Klasse ist in der angegebenen Availability Zone nicht verfügbar.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer Beschreibung der Problemdetails.

InsufficientStorageClusterCapacityFault (Struktur)

Zurückgegebener HTTP-Statuscode: 400.

Es ist nicht genügend Speicherplatz für die aktuelle Aktion verfügbar. Sie können diesen Fehler beheben, indem Sie Ihre Subnetzgruppe so aktualisieren, dass andere Availability Zones mit mehr Speicher verwendet werden.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

Eine Meldung mit einer Beschreibung der Problemdetails.

Ungültig DBCluster EndpointStateFault (Struktur)

Zurückgegebener HTTP-Statuscode: 400.

Der angeforderte Vorgang kann auf dem Endpunkt nicht ausgeführt werden, solange sich der Endpunkt in diesem Status befindet.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer Beschreibung der Problemdetails.

Ungültig DBCluster SnapshotStateFault (Struktur)

Zurückgegebener HTTP-Statuscode: 400.

Der angegebene Wert ist kein gültiger DB-Cluster-Snapshot-Status.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer Beschreibung der Problemdetails.

Ungültig DBCluster StateFault (Struktur)

Zurückgegebener HTTP-Statuscode: 400.

Der DB-Cluster hat keinen gültigen Zustand

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

Eine Meldung mit einer Beschreibung der Problemdetails.

Ungültig DBInstance StateFault (Struktur)

Zurückgegebener HTTP-Statuscode: 400.

Die angegebene DB-Instance ist nicht verfügbar.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer Beschreibung der Problemdetails.

Ungültig DBParameter GroupStateFault (Struktur)

Zurückgegebener HTTP-Statuscode: 400.

Die DB-Parametergruppe wird verwendet oder hat einen ungültigen Zustand. Wenn Sie versuchen, die Parametergruppe zu löschen, ist dies nicht möglich, wenn sie sich in diesem Zustand befindet.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer Beschreibung der Problemdetails.

Ungültig DBSecurity GroupStateFault (Struktur)

Zurückgegebener HTTP-Statuscode: 400.

Der Status der DB-Sicherheitsgruppe erlaubt keine Löschung.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

Eine Meldung mit einer Beschreibung der Problemdetails.

Ungültig DBSnapshot StateFault (Struktur)

Zurückgegebener HTTP-Statuscode: 400.

Der Status des DB-Snapshots erlaubt keine Löschung.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer Beschreibung der Problemdetails.

Ungültig DBSubnet GroupFault (Struktur)

Zurückgegebener HTTP-Statuscode: 400.

Zeigt an, dass die DBSubnet Gruppe nicht zu derselben VPC gehört wie die einer vorhandenen regionsübergreifenden Read Replica derselben Quell-Instance.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer Beschreibung der Problemdetails.

Ungültig DBSubnet GroupStateFault (Struktur)

Zurückgegebener HTTP-Statuscode: 400.

Die DB-Subnetzgruppe kann nicht gelöscht werden, da sie gerade verwendet wird.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

Eine Meldung mit einer Beschreibung der Problemdetails.

Ungültig DBSubnet StateFault (Struktur)

Zurückgegebener HTTP-Statuscode: 400.

Das DB-Subnetz ist nicht verfügbar.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer Beschreibung der Problemdetails.

InvalidEventSubscriptionStateFault (Struktur)

Zurückgegebener HTTP-Statuscode: 400.

Das Ereignisabonnement hat einen ungültigen Zustand.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer Beschreibung der Problemdetails.

InvalidGlobalClusterStateFault (Struktur)

Zurückgegebener HTTP-Statuscode: 400.

Der globale Cluster befindet sich in einem ungültigen Zustand und kann den angeforderten Vorgang nicht ausführen.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

Eine Meldung mit einer Beschreibung der Problemdetails.

InvalidOptionGroupStateFault (Struktur)

Zurückgegebener HTTP-Statuscode: 400.

Die Optionsgruppe ist nicht verfügbar.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer Beschreibung der Problemdetails.

InvalidRestoreFault (Struktur)

Zurückgegebener HTTP-Statuscode: 400.

Eine Wiederherstellung von der vpc-Sicherung auf die Nicht-VPC-DB-Instance ist nicht möglich.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer Beschreibung der Problemdetails.

InvalidSubnet (Struktur)

Zurückgegebener HTTP-Statuscode: 400.

Das angeforderte Subnetz ist ungültig oder mehrere Subnetze wurden angefordert, die sich nicht alle in einer gemeinsamen VPC befinden.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

Eine Meldung mit einer Beschreibung der Problemdetails.

Ungültig VPCNetwork StateFault (Struktur)

Zurückgegebener HTTP-Statuscode: 400.

Die DB-Subnetzgruppe deckt aufgrund der Änderung des Benutzers nicht alle Availability Zones ab, nachdem sie erstellt wurde.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer Beschreibung der Problemdetails.

KMSKeyNotAccessibleFault (Struktur)

Zurückgegebener HTTP-Statuscode: 400.

Fehler beim Zugriff auf den KMS-Schlüssel.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer Beschreibung der Problemdetails.

OptionGroupNotFoundFault (Struktur)

Zurückgegebener HTTP-Statuscode: 404.

Die designierte Optionsgruppe konnte nicht gefunden werden.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

Eine Meldung mit einer Beschreibung der Problemdetails.

PointInTimeRestoreNotEnabledFault (Struktur)

Zurückgegebener HTTP-Statuscode: 400.

Source DBInstance Identifier bezieht sich auf eine DB-Instance BackupRetentionPeriodmit dem Wert 0.

Felder

- message— Dies ist eine ExceptionMessage, vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer Beschreibung der Problemdetails.

ProvisionedIopsNotAvailableInAZFault (Struktur)

Zurückgegebener HTTP-Statuscode: 400.

Bereitgestellte IOPS ist in der angegebenen Availability Zone nicht verfügbar.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer Beschreibung der Problemdetails.

ResourceNotFoundFault (Struktur)

Zurückgegebener HTTP-Statuscode: 404.

Die angegebene Ressourcen-ID wurde nicht gefunden.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

Eine Meldung mit einer Beschreibung der Problemdetails.

SNSInvalidTopicFault (Struktur)

Zurückgegebener HTTP-Statuscode: 400.

Das SNS-Thema ist ungültig.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer Beschreibung der Problemdetails.

SNSNoAuthorizationFault (Struktur)

Zurückgegebener HTTP-Statuscode: 400.

Es gibt keine SNS-Autorisierung.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer Beschreibung der Problemdetails.

SNSTopicArnNotFoundFault (Struktur)

Zurückgegebener HTTP-Statuscode: 404.

Der ARN des SNS-Themas konnte nicht gefunden werden.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

Eine Meldung mit einer Beschreibung der Problemdetails.

SharedSnapshotQuotaExceededFault (Struktur)

Zurückgegebener HTTP-Statuscode: 400.

Sie haben die maximale Anzahl an Konten überschritten, für die Sie einen manuellen DB-Snapshot freigeben können.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer Beschreibung der Problemdetails.

SnapshotQuotaExceededFault (Struktur)

Zurückgegebener HTTP-Statuscode: 400.

Die Anfrage würde dazu führen, dass der Benutzer die zulässige Anzahl an DB-Snapshots überschreitet.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer Beschreibung der Problemdetails.

SourceNotFoundFault (Struktur)

Zurückgegebener HTTP-Statuscode: 404.

Die Quelle konnte nicht gefunden werden.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

Eine Meldung mit einer Beschreibung der Problemdetails.

StorageQuotaExceededFault (Struktur)

Zurückgegebener HTTP-Statuscode: 400.

Die Anfrage würde dazu führen, dass der Benutzer die zulässige Anzahl an Speicher, der in allen DB-Instances verfügbar ist, überschreitet.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer Beschreibung der Problemdetails.

StorageTypeNotSupportedFault (Struktur)

Zurückgegebener HTTP-Statuscode: 400.

StorageTypeDer angegebene Wert kann nicht mit der DB-Instance verknüpft werden.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer Beschreibung der Problemdetails.

SubnetAlreadyInUse (Struktur)

Zurückgegebener HTTP-Statuscode: 400.

Das DB-Subnetz wird bereits in der Availability Zone.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

Eine Meldung mit einer Beschreibung der Problemdetails.

SubscriptionAlreadyExistFault (Struktur)

Zurückgegebener HTTP-Statuscode: 400.

Das Abonnement ist bereits vorhanden.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer Beschreibung der Problemdetails.

SubscriptionCategoryNotFoundFault (Struktur)

Zurückgegebener HTTP-Statuscode: 404.

Die designierte Abonnementkategorie konnte nicht gefunden werden.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer Beschreibung der Problemdetails.

SubscriptionNotFoundFault (Struktur)

Zurückgegebener HTTP-Statuscode: 404.

Das designierte Abonnement konnte nicht gefunden werden.

Felder

- message— Dies ist eine Zeichenfolge ExceptionMessage vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer Beschreibung der Problemdetails.

Amazon Neptune-Daten-API – Referenz

In diesem Kapitel werden die Operationen der Neptune-Daten-API dokumentiert, mit denen Sie die Daten in Ihrem Neptune-Diagramm laden, darauf zugreifen und diese verwalten können.

Die Neptune-Daten-API bietet SDK-Unterstützung für das Laden von Daten, das Ausführen von Abfragen, das Abrufen von Informationen über die Daten und das Ausführen von Machine-Learning-Operationen. Sie unterstützt die Abfragesprachen Gremlin und openCypher in Neptune und ist in allen SDK-Sprachen verfügbar. Sie signiert automatisch API-Anfragen und vereinfacht die Integration von Neptune in Anwendungen erheblich.

Inhalt

- Neptune-Dataplane-Engine, schneller Reset und allgemeine Struktur APIs
  - GetEngineStatus (Aktion)
  - ExecuteFastReset (Aktion)
  - Strukturen der Engine-Operationen:
  - QueryLanguageVersion (Struktur)
  - FastResetToken (Struktur)
- Neptun-Abfrage APIs
  - ExecuteGremlinQuery (Aktion)
  - ExecuteGremlinExplainQuery (Aktion)
  - ExecuteGremlinProfileQuery (Aktion)
  - ListGremlinQueries (Aktion)
  - GetGremlinQueryStatus (Aktion)
  - CancelGremlinQuery (Aktion)
  - openCypher-Abfrageaktionen:
  - ExecuteOpenCypherQuery (Aktion)
  - ExecuteOpenCypherExplainQuery (Aktion)
  - ListOpenCypherQueries (Aktion)
  - GetOpenCypherQueryStatus (Aktion)
  - CancelOpenCypherQuery (Aktion)
  - Abfragestrukturen:

- QueryEvalStats (Struktur)
- GremlinQueryStatus (Struktur)
- GremlinQueryStatusAttributes (Struktur)
- Bulk-Loader-APIs der Neptune-Datenebene
  - StartLoaderJob (Aktion)
  - GetLoaderJobStatus (Aktion)
  - ListLoaderJobs (Aktion)
  - CancelLoaderJob (Aktion)
  - Bulk-Load-Struktur:
  - LoaderIdResult (Struktur)
- Neptune streamt die Datenebenen-API
  - GetPropertygraphStream (Aktion)
  - Stream-Datenstrukturen:
  - PropertygraphRecord (Struktur)
  - PropertygraphData (Struktur)
- Statistiken und grafische Zusammenfassung der Neptun-Datenebene APIs
  - GetPropertygraphStatistics (Aktion)
  - ManagePropertygraphStatistics (Aktion)
  - DeletePropertygraphStatistics (Aktion)
  - GetPropertygraphSummary (Aktion)
  - Statistikstrukturen:
  - Statistics (Struktur)
  - StatisticsSummary (Struktur)
  - DeleteStatisticsValueMap (Struktur)
  - RefreshStatisticsIdMap (Struktur)
  - NodeStructure (Struktur)
  - EdgeStructure (Struktur)
  - SubjectStructure (Struktur)
  - PropertygraphSummaryValueMap (Struktur)
  - PropertygraphSummary (Struktur)

- Neptune ML Datenverarbeitungs-API
  - Start MLData ProcessingJob (Aktion)
  - Liste MLData ProcessingJobs (Aktion)
  - Holen MLData ProcessingJob (Aktion)
  - Abbrechen MLData ProcessingJob (Aktion)
  - Allgemeine Strukturen von ML:
  - MlResourceDefinition (Struktur)
  - MlConfigDefinition (Struktur)
- Neptune ML-Modelltrainings-API
  - Start MLModel TrainingJob (Aktion)
  - Liste MLModel TrainingJobs (Aktion)
  - Holen MLModel TrainingJob (Aktion)
  - Abbrechen MLModel TrainingJob (Aktion)
  - Modelltrainingsstrukturen:
  - CustomModelTrainingParameters (Struktur)
- Neptune ML-Modelltransformations-API
  - Start MLModel TransformJob (Aktion)
  - Liste MLModel TransformJobs (Aktion)
  - Holen MLModel TransformJob (Aktion)
  - Abbrechen MLModel TransformJob (Aktion)
  - Transformationsstrukturen modellieren:
  - CustomModelTransformParameters (Struktur)
- Neptune ML-Inferenzendpunkt-API
  - Erstellen MLEndpoint (Aktion)
  - Liste MLEndpoints (Aktion)
  - Holen MLEndpoint (Aktion)
  - Löschen MLEndpoint (Aktion)
- API-Ausnahmen für Neptune-Datenebenen
  - AccessDeniedException (Struktur)
  - BadRequestException (Struktur)

- BulkLoadIdNotFoundException (Struktur)
- CancelledByUserException (Struktur)
- ClientTimeoutException (Struktur)
- ConcurrentModificationException (Struktur)
- ConstraintViolationException (Struktur)
- ExpiredStreamException (Struktur)
- FailureByQueryException (Struktur)
- IllegalArgumentException (Struktur)
- InternalFailureException (Struktur)
- InvalidArgumentException (Struktur)
- InvalidNumericDataException (Struktur)
- InvalidParameterException (Struktur)
- LoadUrlAccessDeniedException (Struktur)
- MalformedQueryException (Struktur)
- MemoryLimitExceededException (Struktur)
- MethodNotAllowedException (Struktur)
- MissingParameterException (Struktur)
- MLResourceNotFoundException (Struktur)
- ParsingException (Struktur)
- PreconditionsFailedException (Struktur)
- QueryLimitExceededException (Struktur)
- QueryLimitException (Struktur)
- QueryTooLargeException (Struktur)
- ReadOnlyViolationException (Struktur)
- S3Exception (Struktur)
- ServerShutdownException (Struktur)
- StatisticsNotAvailableException (Struktur)
- StreamRecordsNotFoundException (Struktur)
- ThrottlingException (Struktur)
- TimeLimitExceededException (Struktur)

- TooManyRequestsException (Struktur)
- UnsupportedOperationException (Struktur)
- UnloadUrlAccessDeniedException (Struktur)

Neptune-Dataplane-Engine, schneller Reset und allgemeine Struktur APIs

Engine-Operationen:

- GetEngineStatus (Aktion)
- ExecuteFastReset (Aktion)

Strukturen der Engine-Operationen:

- QueryLanguageVersion (Struktur)
- FastResetToken (Struktur)

GetEngineStatus (Aktion)

> Der AWS CLI-Name für diese API lautet:`get-engine-status`.

Ruft den Status der Graphdatenbank auf dem Host ab.

Wenn Sie diesen Vorgang in einem Neptune-Cluster aufrufen, für den die IAM-Authentifizierung aktiviert ist, muss dem IAM-Benutzer oder der Rolle, die die Anfrage gestellt hat, eine Richtlinie angehängt sein, die die Aktion neptune-db: IAM in diesem Cluster zulässt. GetEngineStatus

Anforderung

- Keine Anforderungsparameter.

Antwort

- dbEngineVersion – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Stellen Sie die Version der Neptune-Engine ein, die in Ihrem DB-Cluster ausgeführt wird. Wenn diese Engine-Version nach ihrer Freigabe manuell gepatcht wurde, wird der Versionsnummer der Zusatz `Patch-` vorangestellt.

- dfeQueryEngine – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Wird auf `enabled` gesetzt, wenn die DFE-Engine vollständig aktiviert ist, oder auf `viaQueryHint` (Standardeinstellung), wenn die DFE-Engine nur für Abfragen verwendet wird, bei denen der `useDFE`-Abfragehinweis auf `true` festgelegt ist.
- features – Ein Map-Array von Schlüssel-Wert-Paaren, wobei Folgendes gilt:

  Jeder Schlüssel ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Jeder Wert ist ein Dokument vom Typ `document` (protokollunabhängiger offener Inhalt, der durch ein JSON-ähnliches Datenmodell repräsentiert wird).

  Enthält Statusinformationen zu den im DB-Cluster aktivierten Funktionen.
- gremlin – Ein QueryLanguageVersion-Objekt.

  Enthält Informationen zur Gremlin-Abfragesprache, die im Cluster verfügbar ist. Insbesondere enthält es ein Versionsfeld, das die aktuelle Version angibt, die von der Engine verwendet wird. TinkerPop
- labMode – Ein Map-Array von Schlüssel-Wert-Paaren, wobei Folgendes gilt:

  Jeder Schlüssel ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Jeder Wert ist eine Zeichenfolge vom Typ `string` (eine UTF-8-kodierte Zeichenfolge).

  Enthält Einstellungen des Labor-Modus, die von der Engine verwendet werden.
- opencypher – Ein QueryLanguageVersion-Objekt.

  Enthält Informationen über die openCypher-Abfragesprache, die im Cluster verfügbar ist. Insbesondere enthält es ein Versionsfeld, in dem die aktuelle von der Engine verwendete operCypher-Version angegeben ist.
- role – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Dieses Feld wird auf `reader` festgelegt, wenn die Instance ein Read Replica ist, oder auf `writer`, wenn es sich um die primäre Instance handelt.
- rollingBackTrxCount – eine Ganzzahl vom Typ `integer` (32-Bit-Ganzzahl mit Vorzeichen).

Wenn Transaktionen rückgängig gemacht werden, wird dieses Feld auf die Anzahl solcher Transaktionen festgelegt. Wenn keine Transaktionen vorhanden sind, wird das Feld nicht angezeigt.

- rollingBackTrxEarliestStartTime – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Wird auf die Startzeit der frühesten Transaktion festgelegt, die rückgängig gemacht wird. Wenn keine Transaktionen zurückgesetzt werden, wird dieses Feld nicht im Status angezeigt.

- settings – Ein Map-Array von Schlüssel-Wert-Paaren, wobei Folgendes gilt:

  Jeder Schlüssel ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Jeder Wert ist eine Zeichenfolge vom Typ `string` (eine UTF-8-kodierte Zeichenfolge).

  Enthält Informationen über die aktuellen Einstellungen des DB-Clusters. Enthält beispielsweise die aktuelle Einstellung für den Timeout von Cluster-Abfragen (`clusterQueryTimeoutInMs`).

- sparql – Ein QueryLanguageVersion-Objekt.

  Enthält Informationen über die SPARQL-Abfragesprache, die im Cluster verfügbar ist. Insbesondere enthält es ein Versionsfeld, in dem die aktuelle von der Engine verwendete SPARQL-Version angegeben ist.

- startTime – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Ist auf die UTC-Zeit eingestellt, zu der der aktuelle Serverprozess gestartet wird.

- status – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Wird auf `healthy` gesetzt, wenn bei der Instance keine Probleme auftreten. Wenn die Instance nach einem Ausfall oder Neustart wiederhergestellt wird und noch aktive Transaktionen vom letzten Herunterfahren des Servers ausgeführt werden, wird als Status `recovery` festgelegt.

Fehler

- UnsupportedOperationException
- InternalFailureException
- ClientTimeoutException
- PreconditionsFailedException

- ConstraintViolationException
- InvalidArgumentException
- IllegalArgumentException
- TooManyRequestsException

ExecuteFastReset (Aktion)

Der AWS CLI-Name für diese API lautet:`execute-fast-reset`.

Mit der Fast-Reset-REST-API können Sie ein Neptune-Diagramm schnell und einfach zurücksetzen und dabei alle seine Daten entfernen.

Der Neptune-Fast-Reset für einen Cluster ist ein zweistufiger Prozess. Rufen Sie zuerst `ExecuteFastReset` auf, wobei `action` auf `initiateDatabaseReset` festgelegt sein muss. Dies gibt ein UUID-Token zurück, das Sie dann einbinden, wenn Sie `ExecuteFastReset` erneut aufrufen und `action` auf `performDatabaseReset` festgelegt ist. Siehe Leeren eines Amazon Neptune-DB-Clusters mithilfe der Fast-Reset-API.

Wenn Sie diesen Vorgang in einem Neptune-Cluster aufrufen, für den die IAM-Authentifizierung aktiviert ist, muss dem IAM-Benutzer oder der Rolle, die die Anfrage gestellt hat, eine Richtlinie angehängt sein, die die Aktion neptune-db: IAM in diesem Cluster zulässt. ResetDatabase

Anforderung

- action (in der CLI: `--action`) – Erforderlich: Eine Aktion vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Fast-Reset-Aktion. Einer der folgenden Werte:

  - **`initiateDatabaseReset`** – Durch diese Aktion wird ein eindeutiges Token generiert, das für die tatsächliche Ausführung des Fast Reset benötigt wird.
  - **`performDatabaseReset`** – Diese Aktion verwendet das von der `initiateDatabaseReset`-Aktion generierte Token, um den Fast Reset tatsächlich auszuführen.

- token (in der CLI: `--token`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Das Fast-Reset-Token zur Einleitung des Reset.

Antwort

- payload – Ein FastResetToken-Objekt.

  Die `payload` wird nur von der `initiateDatabaseReset`-Aktion zurückgegeben und enthält das eindeutige Token, das zusammen mit der `performDatabaseReset`-Aktion verwendet werden soll, um den Reset auszuführen.

- status – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der `status` wird nur für die `performDatabaseReset`-Aktion zurückgegeben und gibt an, ob die Fast-Reset-Anforderung akzeptiert wurde oder nicht.

Fehler

- InvalidParameterException
- ClientTimeoutException
- AccessDeniedException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- ServerShutdownException
- PreconditionsFailedException
- MethodNotAllowedException
- ReadOnlyViolationException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException

Strukturen der Engine-Operationen:

QueryLanguageVersion (Struktur)

Struktur zum Ausdrücken der Version der Abfragesprache.

Felder

- version – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Version der Abfragesprache.

FastResetToken (Struktur)

Eine Struktur, die das Fast-Reset-Token enthält, mit dem ein Fast-Reset initiiert wird.

Felder

- token – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine UUID, die von der Datenbank in der `initiateDatabaseReset`-Aktion generiert und dann von der `performDatabaseReset` verwendet wird, um die Datenbank zurückzusetzen.

Neptun-Abfrage APIs

Gremlin-Abfrageaktionen:

- ExecuteGremlinQuery (Aktion)
- ExecuteGremlinExplainQuery (Aktion)
- ExecuteGremlinProfileQuery (Aktion)
- ListGremlinQueries (Aktion)
- GetGremlinQueryStatus (Aktion)
- CancelGremlinQuery (Aktion)

openCypher-Abfrageaktionen:

- ExecuteOpenCypherQuery (Aktion)
- ExecuteOpenCypherExplainQuery (Aktion)
- ListOpenCypherQueries (Aktion)
- GetOpenCypherQueryStatus (Aktion)
- CancelOpenCypherQuery (Aktion)

Abfragestrukturen:

- QueryEvalStats (Struktur)
- GremlinQueryStatus (Struktur)
- GremlinQueryStatusAttributes (Struktur)

ExecuteGremlinQuery (Aktion)

> Der AWS CLI-Name für diese API lautet:`execute-gremlin-query`.

Mit diesem Befehl wird eine Gremlin-Abfrage ausgeführt. Amazon Neptune ist mit Apache TinkerPop 3 und Gremlin kompatibel, sodass Sie die Gremlin Traversal Language verwenden können, um das Diagramm abzufragen, wie unter The Graph in der Apache 3-Dokumentation beschrieben. TinkerPop Weitere Einzelheiten finden Sie auch unter Zugriff auf ein Neptun-Diagramm Graphen mit Gremlin.

Wenn diese Operation in einem Neptune-Cluster mit aktivierter IAM-Authentifizierung aufgerufen wird, muss mit dem IAM-Benutzer oder der Rolle, die die Anforderung gestellt hat, eine Richtlinie verknüpft sein, die je nach Abfrage eine der folgenden IAM-Aktionen in diesem Cluster zulässt:

- Neptun-DB: ReadDataViaQuery
- Neptun-db: WriteDataViaQuery
- Neptun-db: DeleteDataViaQuery

Beachten Sie, dass der QueryLanguageIAM-Bedingungsschlüssel neptune-db ::Gremlin im Richtliniendokument verwendet werden kann, um die Verwendung von Gremlin-Abfragen einzuschränken (siehe Bedingungsschlüssel verfügbar in Neptune IAM-Datenzugriffs-Richtlinienanweisungen).

Anforderung

- gremlinQuery (in der CLI: `--gremlin-query`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Mit dieser API können Sie Gremlin-Abfragen im Zeichenfolgenformat ausführen, ebenso können Sie den HTTP-Endpunkt verwenden. Die Schnittstelle ist mit jeder Gremlin-Version kompatibel, die Ihr DB-Cluster verwendet (im Abschnitt Tinkerpop-Client finden Sie Informationen zur unterstützten Gremlin-Version Ihrer Engine).

- serializer (in der CLI: `--serializer`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Wenn der Wert nicht null ist, werden die Abfrageergebnisse in einer serialisierten Antwortnachricht in dem von diesem Parameter angegebenen Format zurückgegeben. Eine Liste der Formate, die derzeit unterstützt werden, finden Sie im Abschnitt GraphSON in der TinkerPop Dokumentation.

Antwort

- meta – Dokument vom Typ `document` (protokollunabhängiger offener Inhalt, der durch ein JSON-ähnliches Datenmodell repräsentiert wird).

  Metadaten zur Gremlin-Abfrage.
- requestId – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die eindeutige Kennung der Gremlin-Abfrage.
- result – Dokument vom Typ `document` (protokollunabhängiger offener Inhalt, der durch ein JSON-ähnliches Datenmodell repräsentiert wird).

  Die Ausgabe der Gremlin-Abfrage des Servers.
- status – Ein GremlinQueryStatusAttributes-Objekt.

  Der Status der Gremlin-Abfrage.

Fehler

- QueryTooLargeException
- BadRequestException
- QueryLimitExceededException
- InvalidParameterException
- QueryLimitException
- ClientTimeoutException
- CancelledByUserException
- IllegalArgumentException
- TooManyRequestsException

- UnsupportedOperationException
- FailureByQueryException
- MemoryLimitExceededException
- PreconditionsFailedException
- MalformedQueryException
- ParsingException
- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException
- ConcurrentModificationException

ExecuteGremlinExplainQuery (Aktion)

> Der AWS CLI-Name für diese API lautet:`execute-gremlin-explain-query`.

Führt eine Gremlin-Explain-Abfrage aus.

Amazon Neptune hat eine Gremlin-Funktion mit dem Namen `explain` hinzugefügt, die ein Self-Service-Tool bereitstellt, mit dem Sie den Ausführungsansatz verstehen können, den die Neptune-Engine für die Abfrage verwendet. Sie rufen diese Funktion auf, indem Sie einem HTTP-Aufruf, der eine Gremlin-Abfrage sendet, einen `explain`-Parameter hinzufügen.

Die Explain-Funktion bietet Informationen zur logischen Struktur von Abfrageausführungsplänen. So können Sie potenzielle Engpässe bei Bewertung und Ausführung identifizieren und Ihre Abfrage optimieren, wie in Optimieren von Gremline-Abfragen beschrieben. Anschließend können Sie mit Abfragehinweisen die Abfrageausführungspläne verbessern.

Wenn diese Operation in einem Neptune-Cluster mit aktivierter IAM-Authentifizierung aufgerufen wird, muss mit dem IAM-Benutzer oder der Rolle, die die Anforderung gestellt hat, eine Richtlinie verknüpft sein, die je nach Abfrage eine der folgenden IAM-Aktionen in diesem Cluster zulässt:

- neptune-db: ReadDataViaQuery
- Neptun-db: WriteDataViaQuery
- Neptun-db: DeleteDataViaQuery

Beachten Sie, dass der QueryLanguageIAM-Bedingungsschlüssel neptune-db ::Gremlin im Richtliniendokument verwendet werden kann, um die Verwendung von Gremlin-Abfragen einzuschränken (siehe Bedingungsschlüssel verfügbar in Neptune IAM-Datenzugriffs-Richtlinienanweisungen).

Anforderung

- gremlinQuery (in der CLI: `--gremlin-query`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Zeichenfolge für die Gremlin-Explain-Abfrage.

Antwort

- output— a, vom Typ: (ein Block nicht interpretierter Binärdaten). ReportAsText `blob`

  Ein Textblob, der das Ergebnis der Gremlin-Explain-Abfrage enthält, wie unter Optimieren von Gremlin-Abfragen beschrieben.

Fehler

- QueryTooLargeException
- BadRequestException
- QueryLimitExceededException
- InvalidParameterException
- QueryLimitException
- ClientTimeoutException
- CancelledByUserException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- FailureByQueryException
- MemoryLimitExceededException
- PreconditionsFailedException
- MalformedQueryException

- ParsingException
- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException
- ConcurrentModificationException

ExecuteGremlinProfileQuery (Aktion)

Der AWS CLI-Name für diese API lautet:`execute-gremlin-profile-query`.

Führt eine Gremlin-Profile-Abfrage aus, die eine bestimmte Transversale ausführt, erfasst verschiedene Metriken über den Durchlauf und erstellt einen Profilbericht als Ausgabe. Einzelheiten finden Sie unter Gremlin-Profil-API in Neptune.

Wenn dieser Vorgang in einem Neptune-Cluster aufgerufen wird, für den die IAM-Authentifizierung aktiviert ist, muss dem IAM-Benutzer oder der Rolle, die die Anfrage gestellt hat, eine Richtlinie angehängt sein, die die Aktion neptune-db: IAM in diesem Cluster zulässt. ReadDataViaQuery

Beachten Sie, dass der QueryLanguageIAM-Bedingungsschlüssel neptune-db ::Gremlin im Richtliniendokument verwendet werden kann, um die Verwendung von Gremlin-Abfragen einzuschränken (siehe Bedingungsschlüssel verfügbar in Neptune IAM-Datenzugriffs-Richtlinienanweisungen).

Anforderung

- chop (in der CLI: `--chop`) – Ganzzahl vom Typ `integer` (32-Bit-Ganzzahl mit Vorzeichen).

  Wenn der Wert nicht Null ist, wird die Ergebniszeichenfolge mit dieser Anzahl von Zeichen abgeschnitten. Wenn auf Null gesetzt, enthält die Zeichenfolge alle Ergebnisse.
- gremlinQuery (in der CLI: `--gremlin-query`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Gremlin-Abfragezeichenfolge für das Profil.
- indexOps (in der CLI: `--index-ops`) – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

Wenn dieses Flag auf `TRUE` eingestellt ist, enthalten die Ergebnisse einen detaillierten Bericht aller Indexoperationen, die während der Abfrageausführung und -serialisierung durchgeführt wurden.

- results (in der CLI: `--results`) – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Wenn dieses Flag auf `TRUE` eingestellt ist, werden die Abfrageergebnisse gesammelt und als Teil des Profilberichts angezeigt. Bei `FALSE` wird nur die Ergebnisanzahl angezeigt.

- serializer (in der CLI: `--serializer`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Wenn der Wert nicht null ist, werden die gesammelten Ergebnisse in einer serialisierten Antwortnachricht in dem von diesem Parameter angegebenen Format zurückgegeben. Weitere Informationen finden Sie unter Gremlin-Profil-API in Neptune.

Antwort

- output— a, vom Typ: (ein Block nicht interpretierter Binärdaten). ReportAsText `blob`

  Ein Textblob, der das Ergebnis des Gremlin-Profils enthält. Einzelheiten finden Sie unter Gremlin-Profil-API in Neptune.

Fehler

- QueryTooLargeException
- BadRequestException
- QueryLimitExceededException
- InvalidParameterException
- QueryLimitException
- ClientTimeoutException
- CancelledByUserException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- FailureByQueryException

- MemoryLimitExceededException
- PreconditionsFailedException
- MalformedQueryException
- ParsingException
- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException
- ConcurrentModificationException

ListGremlinQueries (Aktion)

Der AWS CLI-Name für diese API lautet:`list-gremlin-queries`.

Listet aktive Gremlin-Abfragen auf. Einzelheiten zur Ausgabe finden Sie unter Gremlin-Abfragestatus-API.

Wenn dieser Vorgang in einem Neptune-Cluster aufgerufen wird, für den die IAM-Authentifizierung aktiviert ist, muss dem IAM-Benutzer oder der Rolle, die die Anfrage gestellt hat, eine Richtlinie angehängt sein, die die Aktion neptune-db: IAM in diesem Cluster zulässt. GetQueryStatus

Beachten Sie, dass der QueryLanguageIAM-Bedingungsschlüssel neptune-db ::Gremlin im Richtliniendokument verwendet werden kann, um die Verwendung von Gremlin-Abfragen einzuschränken (siehe Bedingungsschlüssel verfügbar in Neptune IAM-Datenzugriffs-Richtlinienanweisungen).

Anforderung

- includeWaiting (in der CLI: `--include-waiting`) – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Bei `TRUE` enthält die zurückgegebene Liste wartende Abfragen. Der Standardwert ist `FALSE`;

Antwort

- acceptedQueryCount – eine Ganzzahl vom Typ `integer` (32-Bit-Ganzzahl mit Vorzeichen).

Die Anzahl der Abfragen, die akzeptiert, aber noch nicht abgeschlossen wurden, einschließlich der Abfragen in der Warteschlange.

- queries – Ein Array mit GremlinQueryStatus-Objekten.

  Eine Liste der aktuellen Abfragen.

- runningQueryCount – eine Ganzzahl vom Typ `integer` (32-Bit-Ganzzahl mit Vorzeichen).

  Die Anzahl der derzeit laufenden Gremlin-Abfragen.

Fehler

- BadRequestException
- InvalidParameterException
- ClientTimeoutException
- AccessDeniedException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- FailureByQueryException
- PreconditionsFailedException
- ParsingException
- ReadOnlyViolationException
- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException
- ConcurrentModificationException

GetGremlinQueryStatus (Aktion)

Der AWS CLI-Name für diese API lautet:`get-gremlin-query-status`.

Ruft den Status einer angegebenen Gremlin-Abfrage ab.

Wenn dieser Vorgang in einem Neptune-Cluster aufgerufen wird, für den die IAM-Authentifizierung aktiviert ist, muss dem IAM-Benutzer oder der Rolle, die die Anfrage gestellt hat, eine Richtlinie angehängt sein, die die Aktion neptune-db: IAM in diesem Cluster zulässt. GetQueryStatus

Beachten Sie, dass der QueryLanguageIAM-Bedingungsschlüssel neptune-db ::Gremlin im Richtliniendokument verwendet werden kann, um die Verwendung von Gremlin-Abfragen einzuschränken (siehe Bedingungsschlüssel verfügbar in Neptune IAM-Datenzugriffs-Richtlinienanweisungen).

Anforderung

- queryId (in der CLI: `--query-id`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die eindeutige Kennung der Gremlin-Abfrage.

Antwort

- queryEvalStats – Ein QueryEvalStats-Objekt.

  Der Auswertungsstatus der Gremlin-Abfrage.
- queryId – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die ID der Abfrage, für die der Status zurückgegeben wird.
- queryString – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Gremlin-Abfragezeichenfolge.

Fehler

- BadRequestException
- InvalidParameterException
- ClientTimeoutException
- AccessDeniedException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException

- FailureByQueryException
- PreconditionsFailedException
- ParsingException
- ReadOnlyViolationException
- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException
- ConcurrentModificationException

CancelGremlinQuery (Aktion)

Der AWS CLI-Name für diese API lautet:`cancel-gremlin-query`.

Bricht eine Gremlin-Abfrage ab. Weitere Informationen finden Sie unter Gremlin-Abfrageabbruch.

Wenn dieser Vorgang in einem Neptune-Cluster aufgerufen wird, für den die IAM-Authentifizierung aktiviert ist, muss dem IAM-Benutzer oder der Rolle, die die Anfrage gestellt hat, eine Richtlinie angehängt sein, die die Aktion neptune-db: IAM in diesem Cluster zulässt. CancelQuery

Anforderung

- queryId (in der CLI: `--query-id`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die eindeutige Kennung der abzubrechenden Abfrage.

Antwort

- status – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Status des Abbruchs.

Fehler

- BadRequestException

- InvalidParameterException
- ClientTimeoutException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- FailureByQueryException
- PreconditionsFailedException
- ParsingException
- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException
- ConcurrentModificationException

openCypher-Abfrageaktionen:

ExecuteOpenCypherQuery (Aktion)

Der AWS CLI-Name für diese API lautet:`execute-open-cypher-query`.

Führt eine openCypher-Abfrage aus. Weitere Informationen finden Sie unter Zugriff auf das Neptun-Diagramm mit openCypher.

Neptune unterstützt die Erstellung von Diagrammanwendungen mit openCypher, die zu den zurzeit beliebtesten Abfragesprachen für Entwickler gehört, die mit Graphdatenbanken arbeiten. Entwickler, Geschäftsanalysten und Datenwissenschaftler schätzen die SQL-inspirierte Syntax von openCypher, weil sie eine vertraute Struktur für das Verfassen von Abfragen für Eigenschaftsdiagramme bietet.

Die Abfragesprache openCypher wurde ursprünglich von Neo4j entwickelt, 2015 als Open-Source-Software veröffentlicht und ist unter einer Apache 2-Open-Source-Lizenz für das openCypher-Projekt verfügbar.

Hinweis: Wenn diese Operation in einem Neptune-Cluster mit aktivierter IAM-Authentifizierung aufgerufen wird, muss mit dem IAM-Benutzer oder der Rolle, die die Anforderung gestellt hat, eine

Richtlinie verknüpft sein, die je nach Abfrage eine der folgenden IAM-Aktionen in diesem Cluster zulässt:

- neptune-db: ReadDataViaQuery
- Neptun-db: WriteDataViaQuery
- Neptun-db: DeleteDataViaQuery

Beachten Sie auch, dass der OpenCypher IAM-Bedingungsschlüssel neptune-dbQueryLanguage:: im Richtliniendokument verwendet werden kann, um die Verwendung von OpenCypher-Abfragen einzuschränken (siehe Bedingungsschlüssel verfügbar in Neptune IAM-Datenzugriffs-Richtlinienerklärungen).

Anforderung

- openCypherQuery (in der CLI: `--open-cypher-query`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die openCypher-Abfragezeichenfolge, die ausgeführt werden soll.
- parameters (in der CLI: `--parameters`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die openCypher-Abfrageparameter für die Ausführung der Abfrage. Weitere Informationen finden Sie unter Beispiele für parametrisierte openCypher-Abfragen.

Antwort

- results – Erforderlich Dokument vom Typ `document` (protokollunabhängiger offener Inhalt, der durch ein JSON-ähnliches Datenmodell repräsentiert wird).

  Die openCypherquery-Ergebnisse.

Fehler

- QueryTooLargeException
- InvalidNumericDataException
- BadRequestException
- QueryLimitExceededException

- InvalidParameterException
- QueryLimitException
- ClientTimeoutException
- CancelledByUserException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- FailureByQueryException
- MemoryLimitExceededException
- PreconditionsFailedException
- MalformedQueryException
- ParsingException
- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException
- ConcurrentModificationException

ExecuteOpenCypherExplainQuery (Aktion)

> Der AWS CLI-Name für diese API lautet:`execute-open-cypher-explain-query`.

Führt eine openCypher-`explain`-Abfrage aus. Weitere Informationen finden Sie unter Die openCypher-Explain-Funktion.

Wenn dieser Vorgang in einem Neptune-Cluster aufgerufen wird, für den die IAM-Authentifizierung aktiviert ist, muss dem IAM-Benutzer oder der Rolle, die die Anfrage gestellt hat, eine Richtlinie angehängt sein, die die Aktion neptune-db: IAM in diesem Cluster zulässt. ReadDataViaQuery

Beachten Sie, dass der OpenCypher IAM-Bedingungsschlüssel neptune-dbQueryLanguage:: im Richtliniendokument verwendet werden kann, um die Verwendung von OpenCypher-Abfragen einzuschränken (siehe Bedingungsschlüssel verfügbar in Neptune IAM-Datenzugriffs-Richtlinienerklärungen).

Anforderung

- explainMode(in der CLI:`--explain-mode`) — Erforderlich: an OpenCypherExplainMode, vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

  Der openCypher-`explain`-Modus. Dabei kann es sich um `static`, `dynamic` oder `details` handeln.
- openCypherQuery (in der CLI: `--open-cypher-query`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die openCypher-Abfragezeichenfolge.
- parameters (in der CLI: `--parameters`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die openCypher-Abfrageparameter.

Antwort

- results – Erforderlich: ein Blob vom Typ `blob` (ein Block mit nichtinterpretierten Binärdaten).

  Ein Textblob, der die openCypher-`explain`-Ergebnisse enthält.

Fehler

- QueryTooLargeException
- InvalidNumericDataException
- BadRequestException
- QueryLimitExceededException
- InvalidParameterException
- QueryLimitException
- ClientTimeoutException
- CancelledByUserException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException

- FailureByQueryException
- MemoryLimitExceededException
- PreconditionsFailedException
- MalformedQueryException
- ParsingException
- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException
- ConcurrentModificationException

ListOpenCypherQueries (Aktion)

Der AWS CLI-Name für diese API lautet:`list-open-cypher-queries`.

Listet aktive openCypher-Abfragen auf. Weitere Informationen finden Sie unter Neptune-openCypher-Statusendpunkt.

Wenn dieser Vorgang in einem Neptune-Cluster aufgerufen wird, für den die IAM-Authentifizierung aktiviert ist, muss dem IAM-Benutzer oder der Rolle, die die Anfrage gestellt hat, eine Richtlinie angehängt sein, die die Aktion neptune-db: IAM in diesem Cluster zulässt. GetQueryStatus

Beachten Sie, dass der OpenCypher IAM-Bedingungsschlüssel neptune-dbQueryLanguage:: im Richtliniendokument verwendet werden kann, um die Verwendung von OpenCypher-Abfragen einzuschränken (siehe Bedingungsschlüssel verfügbar in Neptune IAM-Datenzugriffs-Richtlinienerklärungen).

Anforderung

- includeWaiting (in der CLI: `--include-waiting`) – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Wenn auf `TRUE` festgelegt und keine anderen Parameter vorhanden sind, werden Statusinformationen für wartende und ausgeführte Abfragen zurückgegeben.

Antwort

- acceptedQueryCount – eine Ganzzahl vom Typ `integer` (32-Bit-Ganzzahl mit Vorzeichen).

  Die Anzahl der Abfragen, die akzeptiert, aber noch nicht abgeschlossen wurden, einschließlich der Abfragen in der Warteschlange.
- queries – Ein Array mit GremlinQueryStatus-Objekten.

  Eine Liste der aktuellen openCypher-Abfragen.
- runningQueryCount – eine Ganzzahl vom Typ `integer` (32-Bit-Ganzzahl mit Vorzeichen).

  Die Anzahl der aktuell ausgeführten openCypher-Abfragen.

Fehler

- InvalidNumericDataException
- BadRequestException
- InvalidParameterException
- ClientTimeoutException
- AccessDeniedException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- FailureByQueryException
- PreconditionsFailedException
- ParsingException
- ReadOnlyViolationException
- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException
- ConcurrentModificationException

GetOpenCypherQueryStatus (Aktion)

Der AWS CLI-Name für diese API lautet:`get-open-cypher-query-status`.

Ruft den Status einer angegebenen openCypher-Abfrage ab.

Wenn dieser Vorgang in einem Neptune-Cluster aufgerufen wird, für den die IAM-Authentifizierung aktiviert ist, muss dem IAM-Benutzer oder der Rolle, die die Anfrage gestellt hat, eine Richtlinie angehängt sein, die die Aktion neptune-db: IAM in diesem Cluster zulässt. GetQueryStatus

Beachten Sie, dass der OpenCypher IAM-Bedingungsschlüssel neptune-dbQueryLanguage:: im Richtliniendokument verwendet werden kann, um die Verwendung von OpenCypher-Abfragen einzuschränken (siehe Bedingungsschlüssel verfügbar in Neptune IAM-Datenzugriffs-Richtlinienerklärungen).

Anforderung

- queryId (in der CLI: `--query-id`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die eindeutige ID der openCypher-Abfrage, für die der Abfragestatus abgerufen werden soll.

Antwort

- queryEvalStats – Ein QueryEvalStats-Objekt.

  Der Evaluierungsstatus der openCypher-Abfrage.
- queryId – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die eindeutige ID der Abfrage, für die der Status zurückgegeben wird.
- queryString – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die openCypher-Abfragezeichenfolge.

Fehler

- InvalidNumericDataException
- BadRequestException
- InvalidParameterException
- ClientTimeoutException
- AccessDeniedException
- IllegalArgumentException

- TooManyRequestsException
- UnsupportedOperationException
- FailureByQueryException
- PreconditionsFailedException
- ParsingException
- ReadOnlyViolationException
- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException
- ConcurrentModificationException

CancelOpenCypherQuery (Aktion)

Der AWS CLI-Name für diese API lautet:`cancel-open-cypher-query`.

Bricht eine angegebene openCypher-Abfrage ab. Weitere Informationen finden Sie unter Neptune-openCypher-Statusendpunkt.

Wenn dieser Vorgang in einem Neptune-Cluster aufgerufen wird, für den die IAM-Authentifizierung aktiviert ist, muss dem IAM-Benutzer oder der Rolle, die die Anfrage gestellt hat, eine Richtlinie angehängt sein, die die Aktion neptune-db: IAM in diesem Cluster zulässt. CancelQuery

Anforderung

- queryId (in der CLI: `--query-id`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die eindeutige ID der openCypher-Abfrage, die abgebrochen werden soll.
- silent (in der CLI: `--silent`) – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Wenn auf `TRUE` festgelegt, erfolgt der Abbruch der openCypher-Abfrage im Hintergrund.

Antwort

- payload – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Die Abbruchnutzlast für die openCypher-Abfrage.
- status – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Abbruchstatus der openCypher-Abfrage.

Fehler

- InvalidNumericDataException
- BadRequestException
- InvalidParameterException
- ClientTimeoutException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- FailureByQueryException
- PreconditionsFailedException
- ParsingException
- ConstraintViolationException
- TimeLimitExceededException
- InvalidArgumentException
- MissingParameterException
- ConcurrentModificationException

Abfragestrukturen:

QueryEvalStats (Struktur)

Struktur zur Erfassung von Abfragestatistiken, z. B. die Anzahl der ausgeführten, akzeptierten oder wartenden Abfragen sowie und deren Details.

Felder

- cancelled – Dies ist ein boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

Wird auf TRUE gesetzt, wenn die Abfrage abgebrochen wurde, andernfalls auf FALSE.

- elapsed – Dies ist eine Ganzzahl vom Typ `integer` (32-Bit-Ganzzahl mit Vorzeichen).

  Die Dauer in Mikrosekunden, die die Abfrage bis jetzt ausgeführt wurde
- subqueries – Dies ist ein Dokument vom Typ `document` (protokollunabhängiger offener Inhalt, der durch ein JSON-ähnliches Datenmodell repräsentiert wird).

  Die Anzahl der Unterabfragen in dieser Abfrage.
- waited – Dies ist eine Ganzzahl vom Typ `integer` (32-Bit-Ganzzahl mit Vorzeichen).

  Gibt in Millisekunden an, wie lange die Abfrage gewartet hat.

GremlinQueryStatus (Struktur)

Erfasst den Status einer Gremlin-Abfrage (siehe Seite Gremlin-Abfragestatus-API).

Felder

- queryEvalStats – Dies ist ein QueryEvalStats-Objekt.

  Die Statistik der Gremlin-Abfrage.
- queryId – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der ID der Gremlin-Abfrage.
- queryString – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Zeichenfolge der Gremlin-Abfrage.

GremlinQueryStatusAttributes (Struktur)

Enthält Statuskomponenten einer Gremlin-Abfrage.

Felder

- attributes – Dies ist ein Dokument vom Typ `document` (protokollunabhängiger offener Inhalt, der durch ein JSON-ähnliches Datenmodell repräsentiert wird).

  Attribute des Gremlin-Abfragestatus.

- code – Dies ist eine Ganzzahl vom Typ `integer` (32-Bit-Ganzzahl mit Vorzeichen).

  Der von der Gremlin-Abfrageanforderung zurückgegebene HTTP-Antwortcode.

- message – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Statusmeldung.

Bulk-Loader-APIs der Neptune-Datenebene

Bulk-load Aktionen:

- StartLoaderJob (Aktion)
- GetLoaderJobStatus (Aktion)
- ListLoaderJobs (Aktion)
- CancelLoaderJob (Aktion)

Bulk-Load-Struktur:

- LoaderIdResult (Struktur)

StartLoaderJob (Aktion)

Der AWS CLI-Name für diese API lautet:`start-loader-job`.

Startet einen Neptune-Bulk-Loader-Auftrag, um Daten aus einem Amazon-S3-Bucket in eine Neptune-DB-Instance zu laden. Weitere Informationen finden Sie unter Verwenden des Amazon-Neptune-Massen-Loaders für die Aufnahme von Daten.

Wenn dieser Vorgang in einem Neptune-Cluster mit aktivierter IAM-Authentifizierung aufgerufen wird, muss dem IAM-Benutzer oder der Rolle, die die Anfrage gestellt hat, eine Richtlinie angehängt sein, die die Aktion neptune-db: IAM in diesem Cluster zulässt. StartLoaderJob

Anforderung

- dependencies(in der CLI:`--dependencies`) — eine Zeichenfolge vom Typ: `string` (eine UTF-8 kodierte Zeichenfolge).

Dies ist ein optionaler Parameter, mit dem eine Ladeanforderung in der Warteschlange vom erfolgreichen Abschluss eines oder mehrerer früherer Aufträge in der Warteschlange abhängig gemacht werden kann.

Neptune kann bis zu 64 Ladeanforderungen gleichzeitig in die Warteschlange einreihen, wenn die `queueRequest`-Parameter auf `"TRUE"` festgelegt sind. Mit dem Parameter `dependencies` können Sie die Ausführung einer solchen Anforderung in der Warteschlange vom erfolgreichen Abschluss einer oder mehrerer spezifizierter früherer Anforderungen in der Warteschlange abhängig machen.

Wenn Ladung `Job-A` und `Job-B` beispielsweise unabhängig voneinander sind, mit Ladung `Job-C` aber erst nach Abschluss von `Job-A` und `Job-B` begonnen werden kann, gehen Sie wie folgt vor:

1. Senden Sie `load-job-A` und `load-job-B` nacheinander in beliebiger Reihenfolge, und speichern Sie ihre Ladekennungen.
2. Senden Sie `load-job-C` mit den Ladekennungen der beiden Aufträge in seinem `dependencies`-Feld:

Example

```
"dependencies" : ["(job_A_load_id)", "(job_B_load_id)"]
```

Aufgrund des Parameters `dependencies` startet der Bulk-Loader `Job-C` erst dann, nachdem `Job-A` und `Job-B` erfolgreich abgeschlossen wurden. Wenn einer von ihnen fehlschlägt, Job-C wird er nicht ausgeführt und sein Status wird auf `LOAD_FAILED_BECAUSE_DEPENDENCY_NOT_SATISFIED` gesetzt.

Auf diese Weise können Sie mehrere Abhängigkeitsebenen einrichten, sodass das Fehlschlagen eines Auftrags dazu führt, dass alle Anforderungen, die direkt oder indirekt davon abhängig sind, abgebrochen werden.

- failOnError (in der CLI: `--fail-on-error`) – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  `failOnError` – Ein Flag für vollständiges Anhalten bei einem Fehler.

  Zulässige Werte: `"TRUE"`, `"FALSE"`.

  Standardwert: `"TRUE"`.

Wenn dieser Parameter auf "FALSE" gesetzt ist, versucht der Loader, alle Daten am angegebenen Speicherort zu laden, wobei alle Einträge mit Fehlern übersprungen werden.

Wenn dieser Parameter auf "TRUE" eingestellt ist, stoppt der Loader, sobald ein Fehler auftritt. Bis zu diesem Punkt geladene Daten bleiben bestehen.

- format(in der CLI:`--format`) — Erforderlich: ein Format vom Typ: `string` (eine UTF-8 codierte Zeichenfolge).

  Das Format der Daten. Weitere Informationen zu Datenformaten für den Neptune-Befehl `Loader` finden Sie unter Datenformate zum Laden.

  Zulässige Werte

  - **`csv`** für das Gremlin-CSV-Datenformat.
  - **`opencypher`** für das openCypher-CSV-Datenformat.
  - **`ntriples`**für das N-Triples RDF-Datenformat.
  - **`nquads`**für das N-Quads RDF-Datenformat.
  - **`rdfxml`** für das RDF\XML-RDF-Datenformat.
  - **`turtle`** für das Turtle-RDF-Datenformat.
- iamRoleArn(in der CLI:`--iam-role-arn`) — Erforderlich: eine Zeichenfolge vom Typ: `string` (eine UTF-8 kodierte Zeichenfolge).

  Der Amazon-Ressourcenname (ARN) für eine IAM-Rolle, die von der Neptune-DB-Instance für den Zugriff auf den S3-Bucket angenommen wird. Der hier angegebene ARN für die IAM-Rolle muss an den DB-Cluster angehängt werden (siehe Hinzufügen der IAM-Rolle zu einem Amazon-Neptune-Cluster).
- mode(in der CLI:`--mode`) — ein Mode vom Typ: `string` (eine UTF-8 kodierte Zeichenfolge).

  Der Ladeauftragsmodus.

  Zulässige Werte: `RESUME`, `NEW`, `AUTO`.

  Standardwert: `AUTO`.

  - `RESUME` – Im Modus RESUME sucht der Loader nach einem vorherigen Ladevorgang aus dieser Quelle. Wenn der Loader einen vorherigen Ladevorgang findet, setzt er diesen Ladeauftrag fort. Wenn kein vorheriger Ladeauftrag gefunden wird, stoppt der Loader.

Der Loader vermeidet das erneute Laden von Dateien, die in einem früheren Auftrag erfolgreich geladen wurden. Er versucht nur, fehlgeschlagene Dateien zu verarbeiten. Wenn Sie zuvor geladene Daten aus dem Neptune-Cluster gelöscht haben, werden diese Daten in diesem Modus nicht neu geladen. Wenn ein vorheriger Ladeauftrag alle Dateien aus derselben Quelle erfolgreich geladen hat, wird nichts neu geladen und der Loader gibt eine Erfolgsmeldung zurück.

- `NEW` – Der Modus NEW erstellt eine neue Ladeanforderung unabhängig von vorherigen Ladevorgängen. Sie können diesen Modus verwenden, um alle Daten aus einer Quelle erneut zu laden, nachdem die zuvor aus Ihrem Neptune-Cluster geladenen Daten abgelegt wurden, oder um neue Daten zu laden, die in derselben Quelle verfügbar sind.
- `AUTO` – Im Modus AUTO sucht der Loader nach einem vorherigen Ladeauftrag aus derselben Quelle. Wenn er einen Ladeauftrag findet, setzt er diesen Auftrag wie im Modus `RESUME` fort.

  Wenn der Loader keinen vorherigen Ladeauftrag aus derselben Quelle findet, lädt er wie im Modus `NEW` alle Daten aus der Quelle.

- parallelism(in der CLI:`--parallelism`) — ein Parallelismus vom Typ: `string` (eine UTF-8 kodierte Zeichenfolge).

  Der optionale Parameter `parallelism` kann so eingestellt werden, dass die Anzahl der vom Massen-Ladeprozess verwendeten Threads reduziert wird.

  Zulässige Werte:

  - `LOW` – Die Anzahl der verwendeten Threads entspricht der Anzahl der verfügbaren vCPUs dividiert durch 8.
  - `MEDIUM` – Die Anzahl der verwendeten Threads entspricht der Anzahl der verfügbaren vCPUs dividiert durch 2.
  - `HIGH` – Die Anzahl der verwendeten Threads entspricht der Anzahl der verfügbaren vCPUs.
  - `OVERSUBSCRIBE` – Die Anzahl der verwendeten Threads entspricht der Anzahl der verfügbaren vCPUs multipliziert mit 2. Wenn dieser Wert verwendet wird, nimmt der Massen-Loader alle verfügbaren Ressourcen in Anspruch.

    Das bedeutet jedoch nicht, dass die Einstellung `OVERSUBSCRIBE` zu einer CPU-Auslastung von 100 % führt. Da der Ladevorgang I/O gebunden ist, liegt die höchste zu erwartende CPU-Auslastung im Bereich von 60 bis 70%

  Standardwert: `HIGH`

Die Einstellung `parallelism` kann manchmal beim Laden von openCypher-Daten zu einem Deadlock zwischen Threads führen. In diesem Fall gibt Neptune den Fehler `LOAD_DATA_DEADLOCK` zurück. Sie können das Problem im Allgemeinen beheben, indem Sie `parallelism` auf einen niedrigeren Wert festlegen und den Ladebefehl wiederholen.

- parserConfiguration (in der CLI: `--parser-configuration`) – Ein Map-Array von Schlüssel-Wert-Paaren, wobei Folgendes gilt:

  Jeder Schlüssel ist eine Zeichenfolge vom Typ: `string` (eine UTF-8 kodierte Zeichenfolge).

  Jeder Wert ist eine Zeichenfolge vom Typ: `string` (eine UTF-8 kodierte Zeichenfolge).

  `parserConfiguration` – Ein optionales Objekt mit zusätzlichen Parser-Konfigurationswerten. Jeder der untergeordneten Parameter ist ebenfalls optional:

  - **`namedGraphUri`** – Das Standarddiagramm für alle RDF-Formate, wenn kein Graph angegeben ist (für nicht-Quads-Formate und NQUAD-Einträge ohne Graph).

    Der Standardwert ist `https://aws.amazon.com/neptune/vocab/v01/DefaultNamedGraph`.

  - **`baseUri`**— Der Basis-URI für die Formate RDF/XML und Turtle.

    Der Standardwert ist `https://aws.amazon.com/neptune/default`.

  - **`allowEmptyStrings`** – Gremlin-Benutzer müssen beim Laden von CSV-Daten leere Zeichenfolgenwerte ("") als Knoten- und Kanteneigenschaften übergeben können. Wenn `allowEmptyStrings` auf `false` (Standard) festgelegt ist, werden diese leeren Zeichenfolgen als Nullen behandelt und nicht geladen.

    Wenn `allowEmptyStrings` auf `true` festgelegt ist, behandelt der Loader leere Zeichenfolgen als gültige Eigenschaftswerte und lädt sie entsprechend.

- queueRequest (in der CLI: `--queue-request`) – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Dies ist ein optionaler Flag-Parameter, der angibt, ob die Ladeanforderung in die Warteschlange gestellt werden kann oder ob nicht.

  Sie müssen mit dem nächsten Ladeauftrag nicht warten, bis ein Ladeauftrag abgeschlossen ist, da Neptune bis zu 64 Aufträge gleichzeitig in die Warteschlange einreihen kann, wenn alle

`queueRequest`-Parameter auf `"TRUE"` festgelegt sind. Die Aufträge werden nach dem First-In-First-Out (FIFO)-Prinzip eingereiht.

Wenn der Parameter `queueRequest` ausgelassen oder auf `"FALSE"` festgelegt wird, schlägt die Ladeanforderung fehl, wenn bereits ein anderer Ladeauftrag ausgeführt wird.

Zulässige Werte: `"TRUE"`, `"FALSE"`.

Standardwert: `"FALSE"`.

- s3BucketRegion(in der CLI:`--s-3-bucket-region`) — Erforderlich: a S3BucketRegion, vom Typ: `string` (eine UTF-8 kodierte Zeichenfolge).

  Die Amazon-Region Ihres S3-Buckets. Diese muss mit der Amazon-Region des DB-Clusters übereinstimmen.

- source(in der CLI:`--source`) — Erforderlich: eine Zeichenfolge vom Typ: `string` (eine UTF-8 kodierte Zeichenfolge).

  Der Parameter `source` akzeptiert S3-URIs, die eine einzelne Datei, mehrere Dateien, einen Ordner oder mehrere Ordner angeben. Neptune lädt jede Datendatei in den Ordner, der angegeben ist.

  Der URI kann in einem der folgenden Formate angegeben sein.

  - `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)`

  Das `object-key-name` Element der URI entspricht dem Präfixparameter in einem ListObjectsS3-API-Aufruf. Es gibt alle Objekte im angegebenen S3-Bucket an, deren Namen mit diesem Präfix beginnen. Das kann eine einzelne Datei oder ein Ordner oder mehrere and/or Dateiordner sein.

  Die angegebenen Ordner können mehrere Eckpunkt- und Kantendateien enthalten.

- updateSingleCardinalityProperties (in der CLI: `--update-single-cardinality-properties`) – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  `updateSingleCardinalityProperties` ist ein optionaler Parameter, der steuert, wie der Massen-Loader einen neuen Wert für Einzel-Kardinalität-Vertex- oder Edge-Eigenschaften behandelt.

Zulässige Werte: `"TRUE"`, `"FALSE"`.

Standardwert: `"FALSE"`.

Standardmäßig oder wenn `updateSingleCardinalityProperties` explizit als `"FALSE"` festgelegt ist, behandelt der Loader einen neuen Wert als Fehler, da er gegen die Einzel-Kardinalität verstößt.

Wenn `updateSingleCardinalityProperties` als `"TRUE"` festgelegt ist, ersetzt der Massen-Loader auf der anderen Seite den vorhandenen Wert durch den neuen. Wenn in den Quelldateien, die geladen werden, mehrere Edge- oder Einzel-Kardinalität-Vertex-Eigenschaftswerte angegeben werden, kann der endgültige Wert am Ende des Massenladevorgangs jeder dieser neuen Werte sein. Der Loader stellt nur sicher, dass der vorhandene Wert durch einen der neuen ersetzt wurde.

- userProvidedEdgeIds (in der CLI: `--user-provided-edge-ids`) – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Dieser Parameter ist nur erforderlich, wenn openCypher-Daten geladen werden, die Beziehungs-IDs enthalten. Er muss enthalten und auf `True` festgelegt sein, wenn openCypher-Beziehungs-IDs explizit in den Ladedaten angegeben werden (empfohlen).

  Wenn `userProvidedEdgeIds` fehlt oder auf `True` festgelegt ist, muss in jeder Beziehungsdatei im Ladevorgang die Spalte `:ID` vorhanden sein.

  Wenn `userProvidedEdgeIds` vorhanden und auf `False` festgelegt ist, dürfen Beziehungsdateien im Ladevorgang die Spalte `:ID` nicht enthalten. Stattdessen generiert der Neptune-Loader automatisch eine ID für jede Beziehung.

  Wenn Beziehungs-IDs explizit angegeben sind, kann der Loader den Ladevorgang nach der Behebung eines Fehlers in den CSV-Daten fortsetzen, ohne bereits geladene Beziehungen erneut laden zu müssen. Wenn Beziehungs-IDs nicht explizit angegeben sind, kann der Loader einen fehlgeschlagenen Ladevorgang nach der Korrektur einer Beziehungsdatei nicht fortsetzen und muss stattdessen alle Beziehungen erneut laden.

Antwort

- payload – Erforderlich: Es handelt sich um ein Map-Array von Schlüssel-Wert-Paaren, wobei Folgendes gilt:

Jeder Schlüssel ist eine Zeichenfolge vom Typ: `string` (eine UTF-8 kodierte Zeichenfolge).

Jeder Wert ist eine Zeichenfolge vom Typ: `string` (eine UTF-8 kodierte Zeichenfolge).

Enthält ein `loadId`-Name-Wert-Paar, das einen Bezeichner für den Ladevorgang bereitstellt.

- status— Erforderlich: eine Zeichenfolge vom Typ: `string` (eine UTF-8 kodierte Zeichenfolge).

  Der HTTP-Rückgabecode, der den Status des Ladeauftrags angibt.

Fehler

- BadRequestException
- InvalidParameterException
- BulkLoadIdNotFoundException
- ClientTimeoutException
- LoadUrlAccessDeniedException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- InternalFailureException
- S3Exception
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException

GetLoaderJobStatus (Aktion)

Der AWS CLI-Name für diese API lautet:`get-loader-job-status`.

Ruft Statusinformationen zu einem angegebenen Ladeauftratg ab. Neptune verfolgt die letzten 1.024 Massenladeaufgaben und speichert pro Auftrag die letzten 10.000 Fehlerdetails.

Weitere Informationen finden Sie unter Neptune Loader Get-Status API.

Wenn Sie diesen Vorgang in einem Neptune-Cluster aufrufen, für den die IAM-Authentifizierung aktiviert ist, muss dem IAM-Benutzer oder der Rolle, die die Anfrage gestellt hat, eine Richtlinie angehängt sein, die die Aktion neptune-db: IAM in diesem Cluster zulässt. GetLoaderJobStatus

Anforderung

- details (in der CLI: `--details`) – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Flag, das angibt, ob neben dem Gesamtstatus weitere Details einbezogen werden sollen (`TRUE`oder `FALSE`; die Standardeinstellung lautet `FALSE`).

- errors (in der CLI: `--errors`) – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Flag, das angibt, ob eine Liste der aufgetretenen Fehler einbezogen werden soll (`TRUE`oder`FALSE`; die Standardeinstellung lautet `FALSE`).

  Die Fehlerliste ist segmentiert. Die Parameter `page` und `errorsPerPage` erlauben Ihnen das seitenweise Durchlaufen aller Fehler.

- errorsPerPage(in der CLI:`--errors-per-page`) — a PositiveInteger, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen), mindestens 1? st?.

  Die Anzahl der auf jeder Seite zurückgegebenen Fehler (eine positive Ganzzahl; die Standardeinstellung lautet `10`). Dieser Wert ist nur gültig, wenn der `errors`-Parameter auf `TRUE` eingestellt ist.

- loadId(in der CLI:`--load-id`) — Erforderlich: eine Zeichenfolge vom Typ: `string` (eine UTF-8 kodierte Zeichenfolge).

  Die Lade-ID des Ladeauftrags, dessen Status abgerufen werden soll.

- page(in der CLI:`--page`) — a PositiveInteger, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen), mindestens 1? st?.

  Die Fehlerseitenzahl (eine positive Ganzzahl; die Standardeinstellung lautet 1). Dieser Wert ist nur gültig, wenn der `errors`-Parameter auf `TRUE` eingestellt ist.

Antwort

- payload— Erforderlich: ein Dokument des Typs: `document` (ein protokollunabhängiger offener Inhalt, der durch ein Datenmodell repräsentiert wird). JSON-like

  Statusinformationen über den Ladeauftrag in einem Layout, das wie folgt aussehen könnte:

  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— Erforderlich: eine Zeichenfolge vom Typ: `string` (eine kodierte Zeichenfolge). UTF-8

  Den HTTP-Statuscode für die Antwort der Anforderung.

Fehler

- BadRequestException
- InvalidParameterException
- BulkLoadIdNotFoundException
- ClientTimeoutException
- LoadUrlAccessDeniedException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- InternalFailureException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException

ListLoaderJobs (Aktion)

Der AWS CLI-Name für diese API lautet:`list-loader-jobs`.

Ruft eine Liste der `loadIds` für alle aktiven Ladeaufträge ab.

Wenn Sie diesen Vorgang in einem Neptune-Cluster aufrufen, für den die IAM-Authentifizierung aktiviert ist, muss dem IAM-Benutzer oder der Rolle, die die Anfrage gestellt hat, eine Richtlinie angehängt sein, die die Aktion neptune-db: IAM in diesem Cluster zulässt. ListLoaderJobs

Anforderung

- includeQueuedLoads (in der CLI: `--include-queued-loads`) – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Ein optionaler Parameter, mit dem die Ladekennungen von Ladeanforderungen in der Warteschlange ausgeschlossen werden können, wenn eine Liste von Ladekennungen angefordert wird, indem der Parameter auf `FALSE` festgelegt wird. Der Standardwert ist `TRUE`.
- limit(in der CLI:`--limit`) — a ListLoaderJobsInputLimitInteger, vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen), nicht weniger als 1 oder mehr als 100? st? s

  Die Anzahl der Lade-IDs, die aufgelistet werden sollen. Muss eine positive Ganzzahl sein, die größer als Null und kleiner als `100` (Standardeinstellung) ist.

Antwort

- payload – Erforderlich: Ein LoaderIdResult-Objekt.

  Die angeforderte Liste der Auftrags-IDs.
- status— Erforderlich: eine Zeichenfolge vom Typ: `string` (eine UTF-8 kodierte Zeichenfolge).

  Gibt den Status der Anforderung der Auftragsliste zurück.

Fehler

- UnsupportedOperationException
- BadRequestException
- InvalidParameterException
- BulkLoadIdNotFoundException
- InternalFailureException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException

- InvalidArgumentException
- LoadUrlAccessDeniedException
- IllegalArgumentException
- TooManyRequestsException

CancelLoaderJob (Aktion)

Der AWS CLI-Name für diese API lautet:`cancel-loader-job`.

Bricht einen angegebenen Ladeauftrag ab. Dies ist eine HTTP-`DELETE`-Anforderung. Weitere Informationen finden Sie unter Neptune Loader Get-Status API.

Wenn Sie diesen Vorgang in einem Neptune-Cluster aufrufen, für den die IAM-Authentifizierung aktiviert ist, muss dem IAM-Benutzer oder der Rolle, die die Anfrage gestellt hat, eine Richtlinie angehängt sein, die die Aktion neptune-db: IAM in diesem Cluster zulässt. CancelLoaderJob

Anforderung

- loadId(in der CLI:`--load-id`) — Erforderlich: eine Zeichenfolge vom Typ: `string` (eine UTF-8 kodierte Zeichenfolge).

  Die ID des zu löschenden Ladeauftrags.

Antwort

- status— eine Zeichenfolge vom Typ: `string` (eine UTF-8 kodierte Zeichenfolge).

  Der Abbruchstatus.

Fehler

- BadRequestException
- InvalidParameterException
- BulkLoadIdNotFoundException
- ClientTimeoutException
- LoadUrlAccessDeniedException
- IllegalArgumentException

- TooManyRequestsException
- UnsupportedOperationException
- InternalFailureException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException

Bulk-Load-Struktur:

LoaderIdResult (Struktur)

Enthält eine Liste von Lade-IDs.

Felder

- loadIds— Dies ist eine Zeichenfolge vom Typ: `string` (eine UTF-8 kodierte Zeichenfolge).

  Eine Liste von Lade-IDs.

Neptune streamt die Datenebenen-API

Stream-Zugriff-Aktionen:

- GetPropertygraphStream (Aktion)

Stream-Datenstrukturen:

- PropertygraphRecord (Struktur)
- PropertygraphData (Struktur)

GetPropertygraphStream (Aktion)

> Der AWS CLI-Name für diese API lautet:`get-propertygraph-stream`.

Ruft einen Stream für ein Eigenschaftsdiagramm ab.

Mit der Streams-Funktion von Neptune können Sie eine vollständige Abfolge von Änderungsprotokolleinträgen generieren, die jede Änderung an Ihren Diagrammdaten aufzeichnen, sobald sie erfolgt. Mit `GetPropertygraphStream` erfassen Sie diese Änderungsprotokollierungen für ein Eigenschaftsdiagramm.

Die Neptun-Streams-Funktion muss auf deinem Neptune aktiviert sein. DBcluster Um Streams zu aktivieren, setzen Sie den Parameter neptune_streams für den DB-Cluster auf 1.

Weitere Informationen finden Sie unter Erfassen von Diagrammänderungen in Echtzeit mithilfe von Neptune-Streams.

Wenn dieser Vorgang in einem Neptune-Cluster mit aktivierter IAM-Authentifizierung aufgerufen wird, muss dem IAM-Benutzer oder der Rolle, die die Anfrage gestellt hat, eine Richtlinie angehängt sein, die die Aktion neptune-db: IAM in diesem Cluster zulässt. GetStreamRecords

Wenn diese Operation in einem Neptune-Cluster mit aktivierter IAM-Authentifizierung aufgerufen wird, muss mit dem IAM-Benutzer oder der Rolle, die die Anforderung gestellt hat, eine Richtlinie verknüpft sein, die je nach Abfrage eine der folgenden IAM-Aktionen zulässt.

Hinweis: Sie können Abfragen des Eigenschaftsdiagramms mithilfe der folgenden IAM-Kontextschlüssel eingrenzen:

- neptune-db ::Gremlin QueryLanguage
- Neptun-DB:: QueryLanguage OpenCypher

Weitere Informationen finden Sie unter In Neptune-IAM-Datenzugriffs-Richtlinienanweisungen verfügbare Bedingungsschlüssel).

Anforderung

- commitNum (in der CLI: `--commit-num`) – Long vom Typ `long` (64-Bit-Ganzzahl mit Vorzeichen).

  Die Commit-Nummer des Startdatensatzes, der aus dem Änderungsprotokoll-Stream gelesen werden soll. Dieser Parameter ist erforderlich, wenn `iteratorType` den Wert `AT_SEQUENCE_NUMBER` oder `AFTER_SEQUENCE_NUMBER` aufweist, und wird ignoriert, wenn `iteratorType` den Wert `TRIM_HORIZON` oder `LATEST` aufweist.
- encoding (in der CLI: `--encoding`) – Codierung vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Wenn auf TRUE gesetzt, komprimiert Neptune die Antwort mithilfe der Gzip-Kodierung.

- iteratorType(in der CLI:`--iterator-type`) — ein IteratorType, vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

  Kann einer der folgenden sein:

  - `AT_SEQUENCE_NUMBER` – Gibt an, dass der Lesevorgang bei der Ereignissequenznummer beginnen soll, die gemeinsam durch die Parameter `commitNum` und `opNum` angegeben wird.
  - `AFTER_SEQUENCE_NUMBER` – Gibt an, dass der Lesevorgang direkt nach der Ereignissequenznummer beginnen soll, die gemeinsam durch die Parameter `commitNum` und `opNum` angegeben wird.
  - `TRIM_HORIZON` – Gibt an, dass der Lesevorgang mit dem letzten nicht gekürzten Datensatz im System beginnen soll. Dabei handelt es sich um den ältesten nicht abgelaufenen (noch nicht gelöschten) Datensatz im Änderungsprotokoll-Stream.
  - `LATEST` – Gibt an, dass der Lesevorgang mit dem neuesten Datensatz im System beginnen soll. Dabei handelt es sich um den letzten nicht abgelaufenen (noch nicht gelöschten) Datensatz im Änderungsprotokoll-Stream.
- limit(in der CLI:`--limit`) — a GetPropertygraphStreamInputLimitLong, vom Typ: `long` (eine 64-Bit-Ganzzahl mit Vorzeichen), nicht weniger als 1 oder mehr als 100000? st? s

  Gibt die maximale Anzahl der zurückzugebenden Datensätze an. Es gibt auch eine Größenbeschränkung von 10 MB für die Antwort, die nicht geändert werden kann und Vorrang vor der Anzahl der Datensätze hat, die im `limit`-Parameter angegeben ist. Die Antwort enthält einen Schwellenwertüberschreitungsdatensatz, wenn das 10 MB-Limit erreicht wurde.

  Der Bereich für `limit` liegt zwischen 1 und 100 000, der Standardwert lautet 10.
- opNum (in der CLI: `--op-num`) – Long vom Typ `long` (64-Bit-Ganzzahl mit Vorzeichen).

  Die Operationssequenznummer innerhalb des angegebenen Commitments, ab der in den Änderungsprotokoll-Streamdaten gelesen werden soll. Der Standardwert ist `1`.

Antwort

- format – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Serialisierungsformat für die zurückgegebenen Änderungsdatensätze. Derzeit wird als einziger Wert unterstütz `PG_JSON`.
- lastEventId – Erforderlich: Es handelt sich um ein Map-Array von Schlüssel-Wert-Paaren, wobei Folgendes gilt:

Jeder Schlüssel ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Jeder Wert ist eine Zeichenfolge vom Typ `string` (eine UTF-8-kodierte Zeichenfolge).

Sequenzkennung der letzten Änderung in der Stream-Antwort.

Eine Ereignis-ID besteht aus zwei Feldern: `commitNum` identifiziert eine Transaktion, die das Diagramm geändert hat, und `opNum` identifiziert eine bestimmte Operation innerhalb dieser Transaktion:

Example

```
"eventId": {
          "commitNum": 12,
          "opNum": 1
        }
```

- lastTrxTimestampInMillis – Erforderlich: Long vom Typ `long` (eine 64-Bit-Ganzzahl mit Vorzeichen).

  Der Zeitpunkt, zu dem das Commitment für die Transaktion angefordert wurde, in Millisekunden ab der Unix-Epoche.

- records – Erforderlich: Ein Array mit PropertygraphRecord-Objekten.

  Ein Array serialisierter Änderungsprotokoll-Stream-Datensätze, die in der Antwort enthalten sind.

- totalRecords – Erforderlich: eine Ganzzahl vom Typ `integer` (32-Bit-Ganzzahl mit Vorzeichen).

  Die Gesamtanzahl der Datensätze in der Antwort.

Fehler

- UnsupportedOperationException
- ExpiredStreamException
- InvalidParameterException
- MemoryLimitExceededException
- StreamRecordsNotFoundException
- ClientTimeoutException

- PreconditionsFailedException
- ThrottlingException
- ConstraintViolationException
- InvalidArgumentException
- IllegalArgumentException
- TooManyRequestsException

Stream-Datenstrukturen:

PropertygraphRecord (Struktur)

Struktur eines Eigenschaftsdiagramm-Datensatzes.

Felder

- commitTimestampInMillis – Dies ist erforderlich: Long vom Typ `long` (eine 64-Bit-Ganzzahl mit Vorzeichen).

  Der Zeitpunkt, zu dem das Commitment für die Transaktion angefordert wurde, in Millisekunden ab der Unix-Epoche.
- data – Dies ist erforderlich: Ein PropertygraphData-Objekt.

  Der serialisierte Gremlin- oder openCypher-Änderungsdatensatz.
- eventId – Dies ist erforderlich: Es handelt sich um ein Map-Array von Schlüssel-Wert-Paaren, wobei Folgendes gilt:

  Jeder Schlüssel ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Jeder Wert ist eine Zeichenfolge vom Typ `string` (eine UTF-8-kodierte Zeichenfolge).

  Die Sequenzkennung des Stream-Änderungsdatensatzes.
- isLastOp – Dies ist ein boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Nur vorhanden, wenn diese Operation die letzte in ihrer Transaktion ist. Falls vorhanden, wird der Wert auf "true" festgelegt. Nützlich, um sicherzustellen, dass die gesamte Transaktion genutzt wird.
- op – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Die Operation, die die Änderung erstellt hat.

PropertygraphData (Struktur)

Ein Gremlin- oder openCypher-Änderungsdatensatz.

Felder

- from – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Wenn dies ein Edge ist (Typ = `e`), die ID des entsprechenden `from`-Eckpunkts oder Quellknotens.
- id – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die ID des Gremlin- oder openCypher-Elements.
- key – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Name der Eigenschaft. Für Elementbezeichnungen ist dies `label`.
- to – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Wenn dies ein Edge ist (Typ = `e`), die ID des entsprechenden `to`-Eckpunkts oder Zielknotens.
- type – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Typ dieses Gremlin- oder openCypher-Elements. Zulässige Werte sind:
  - **`v1`** – Eckpunktbezeichnung für Gremlin oder Knotenbezeichnung für openCypher.
  - **`vp`** – Eckpunkteigenschaften für Gremlin oder Knoteneigenschaften für openCypher.
  - **`e`** – Kante und Kantenbezeichnung für Gremlin oder Beziehung und Beziehungstyp für openCypher.
  - **`ep`** – Kanteneigenschaften für Gremlin oder Beziehungseigenschaften für openCypher.
- value – Dies ist erforderlich: Dokument vom Typ `document` (protokollunabhängiger offener Inhalt, der durch ein JSON-ähnliches Datenmodell repräsentiert wird).

  Dies ist ein JSON-Objekt, das ein Feld "value" für den Wert selbst und ein Feld "datatype" für den JSON-Datentyp dieses Werts enthält:

  Example

```
"value": {
```

```
    "value": "(the new value"),
    "dataType": "(the JSON datatypenew value")
}
```

Statistiken und grafische Zusammenfassung der Neptun-Datenebene APIs

Aktionen für die Statistik von Eigenschaftsdiagrammen:

- GetPropertygraphStatistics (Aktion)
- ManagePropertygraphStatistics (Aktion)
- DeletePropertygraphStatistics (Aktion)
- GetPropertygraphSummary (Aktion)

Statistikstrukturen:

- Statistics (Struktur)
- StatisticsSummary (Struktur)
- DeleteStatisticsValueMap (Struktur)
- RefreshStatisticsIdMap (Struktur)
- NodeStructure (Struktur)
- EdgeStructure (Struktur)
- SubjectStructure (Struktur)
- PropertygraphSummaryValueMap (Struktur)
- PropertygraphSummary (Struktur)

GetPropertygraphStatistics (Aktion)

> Der AWS CLI-Name für diese API lautet:`get-propertygraph-statistics`.

Ruft Statistiken zu Eigenschaftsdiagrammen ab (Gremlin und openCypher).

Wenn dieser Vorgang in einem Neptune-Cluster aufgerufen wird, für den die IAM-Authentifizierung aktiviert ist, muss dem IAM-Benutzer oder der Rolle, die die Anfrage gestellt hat, eine Richtlinie angehängt sein, die die Aktion neptune-db: IAM in diesem Cluster zulässt. GetStatisticsStatus

Anforderung

- Keine Anforderungsparameter.

Antwort

- payload – Erforderlich: Ein Statistiken-Objekt.

  Statistiken für Daten in Eigenschaftsdiagrammen.
- status – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der HTTP-Rückgabecode der Anforderung. Wenn die Anforderung erfolgreich ist, lautet der Code 200. Eine Liste der häufigsten Fehler finden Sie unter Häufige Fehlercodes für DFE-Statistikanforderungen.

Fehler

- BadRequestException
- InvalidParameterException
- StatisticsNotAvailableException
- ClientTimeoutException
- AccessDeniedException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- PreconditionsFailedException
- ReadOnlyViolationException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException

ManagePropertygraphStatistics (Aktion)

Der AWS CLI-Name für diese API lautet:`manage-propertygraph-statistics`.

Verwaltet die Generierung und Verwendung der Statistiken von Eigenschaftsdiagrammen.

Wenn dieser Vorgang in einem Neptune-Cluster aufgerufen wird, für den die IAM-Authentifizierung aktiviert ist, muss dem IAM-Benutzer oder der Rolle, die die Anfrage gestellt hat, eine Richtlinie angehängt sein, die die Aktion neptune-db: IAM in diesem Cluster zulässt. ManageStatistics

Anforderung

- mode(in der CLI:`--mode`) — a StatisticsAutoGenerationMode, vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

  Der Modus der Statistikgenerierung. Es kann sich um einen der folgenden Modi handeln: `DISABLE_AUTOCOMPUTE`, `ENABLE_AUTOCOMPUTE` oder `REFRESH`, wobei letzterer die Generierung von DFE-Statistiken manuell auslöst.

Antwort

- payload – Ein RefreshStatisticsIdMap-Objekt.

  Dies wird nur für den Aktualisierungsmodus zurückgegeben.
- status – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der HTTP-Rückgabecode der Anforderung. Wenn die Anforderung erfolgreich ist, lautet der Code 200.

Fehler

- BadRequestException
- InvalidParameterException
- StatisticsNotAvailableException
- ClientTimeoutException
- AccessDeniedException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- PreconditionsFailedException
- ReadOnlyViolationException

- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException

DeletePropertygraphStatistics (Aktion)

Der AWS CLI-Name für diese API lautet:`delete-propertygraph-statistics`.

Löscht Statistiken für Gremlin- und openCypher(-Eigenschaftsdiagramm)-Daten.

Wenn dieser Vorgang in einem Neptune-Cluster aufgerufen wird, für den die IAM-Authentifizierung aktiviert ist, muss dem IAM-Benutzer oder der Rolle, die die Anfrage gestellt hat, eine Richtlinie angehängt sein, die die Aktion neptune-db: IAM in diesem Cluster zulässt. DeleteStatistics

Anforderung

- Keine Anforderungsparameter.

Antwort

- payload – Ein DeleteStatisticsValueMap-Objekt.

  Die Nutzlast des Löschvorgangs.
- status – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Abbruchstatus.
- statusCode – eine Ganzzahl vom Typ `integer` (32-Bit-Ganzzahl mit Vorzeichen).

  HTTP-Antwortcode 200, wenn das Löschen erfolgreich war, oder 204, wenn keine Statistiken zum Löschen vorhanden waren.

Fehler

- BadRequestException
- InvalidParameterException
- StatisticsNotAvailableException
- ClientTimeoutException

- AccessDeniedException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- PreconditionsFailedException
- ReadOnlyViolationException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException

GetPropertygraphSummary (Aktion)

Der AWS CLI-Name für diese API lautet:`get-propertygraph-summary`.

Ruft eine Diagrammübersicht für ein Eigenschaftsdiagramm ab.

Wenn dieser Vorgang in einem Neptune-Cluster aufgerufen wird, für den die IAM-Authentifizierung aktiviert ist, muss dem IAM-Benutzer oder der Rolle, die die Anfrage gestellt hat, eine Richtlinie angehängt sein, die die Aktion neptune-db: IAM in diesem Cluster zulässt. GetGraphSummary

Anforderung

- mode(in der CLI:`--mode`) — a GraphSummaryType, vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

  Der Modus kann einen der folgenden beiden Werte annehmen: `BASIC` (Standard) und `DETAILED`.

Antwort

- payload – Ein PropertygraphSummaryValueMap-Objekt.

  Nutzlast, die Antwort der Eigenschaftsdiagrammübersicht enthält.
- statusCode – eine Ganzzahl vom Typ `integer` (32-Bit-Ganzzahl mit Vorzeichen).

  Der HTTP-Rückgabecode der Anforderung. Wenn die Anforderung erfolgreich ist, lautet der Code 200.

Fehler

- BadRequestException
- InvalidParameterException
- StatisticsNotAvailableException
- ClientTimeoutException
- AccessDeniedException
- IllegalArgumentException
- TooManyRequestsException
- UnsupportedOperationException
- PreconditionsFailedException
- ReadOnlyViolationException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException

Statistikstrukturen:

Statistics (Struktur)

Enthält statistische Informationen. Die DFE-Engine verwendet Informationen zu den Daten in Ihrem Neptune-Diagramm, um bei der Planung der Abfrageausführung effektive Kompromisse einzugehen. Bei diesen Informationen handelt es sich um Statistiken, die sogenannte Merkmalssätze und Prädikatstatistiken umfassen, die als Anleitung für die Abfrageplanung dienen können. Siehe Verwalten von Statistiken, die die Neptune-DFE-Engine verwenden soll.

Felder

- active – Dies ist ein boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Gibt an, ob die automatische Generierung von DFE-Statistiken aktiviert ist oder nicht.
- autoCompute – Dies ist ein boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Gibt an, ob die automatische Generierung von Statistiken aktiviert ist oder nicht.

- date— Dies ist eine SyntheticTimestamp _date_time vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

  Die UTC-Zeit, zu der die DFE-Statistiken zuletzt generiert wurden.
- note – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Ein Hinweis zu Problemen in dem Fall, dass Statistiken ungültig sind.
- signatureInfo – Dies ist ein StatisticsSummary-Objekt.

  StatisticsSummary Eine Struktur, die Folgendes enthält:
  - `signatureCount` – Gesamtzahl der Signaturen für alle Merkmalsätze.
  - `instanceCount` – Gesamtzahl der Merkmalsatz-Instances.
  - `predicateCount` – Gesamtzahl der eindeutigen Prädikate.
- statisticsId – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Meldet die ID der aktuellen Statistikgenerierungsausführung. Der Wert -1 gibt an, dass keine Statistiken generiert wurden.

StatisticsSummary (Struktur)

Informationen zu den in der Statistik generierten Merkmalssätzen.

Felder

- instanceCount – Dies ist eine Ganzzahl vom Typ `integer` (32-Bit-Ganzzahl mit Vorzeichen).

  Gesamtzahl der Merkmalsatz-Instances.
- predicateCount – Dies ist eine Ganzzahl vom Typ `integer` (32-Bit-Ganzzahl mit Vorzeichen).

  Gesamtzahl der eindeutigen Prädikate.
- signatureCount – Dies ist eine Ganzzahl vom Typ `integer` (32-Bit-Ganzzahl mit Vorzeichen).

  Gesamtzahl der Signaturen für alle Merkmalsätze.

DeleteStatisticsValueMap (Struktur)

Die Nutzlast für DeleteStatistics.

Felder

- active – Dies ist ein boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Aktueller Status der Statistik.
- statisticsId – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die ID der Statistikgenerierung, die zurzeit ausgeführt wird.

RefreshStatisticsIdMap (Struktur)

Statistiken für den Modus `REFRESH`.

Felder

- statisticsId – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die ID der Statistikgenerierung, die zurzeit ausgeführt wird.

NodeStructure (Struktur)

Eine Knotenstruktur.

Felder

- count – Dies ist ein Long vom Typ `long` (eine 64-Bit-Ganzzahl mit Vorzeichen).

  Anzahl der Knoten, die diese spezifische Struktur aufweisen.
- distinctOutgoingEdgeLabels – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Liste der eindeutigen Kantenbezeichnungen, die in dieser spezifischen Struktur vorhanden sind.
- nodeProperties – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Liste der Knoteneigenschaften, die in dieser spezifischen Struktur vorhanden sind.

EdgeStructure (Struktur)

Eine Edge-Struktur.

Felder

- count – Dies ist ein Long vom Typ `long` (eine 64-Bit-Ganzzahl mit Vorzeichen).

  Anzahl der Kanten, die diese spezifische Struktur aufweisen.
- edgeProperties – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Liste der Kanteneigenschaften, die in dieser spezifischen Struktur vorhanden sind.

SubjectStructure (Struktur)

Eine Subjektstruktur.

Felder

- count – Dies ist ein Long vom Typ `long` (eine 64-Bit-Ganzzahl mit Vorzeichen).

  Anzahl der Vorkommen dieser spezifischen Struktur.
- predicates – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Liste der Prädikate, die in dieser spezifischen Struktur vorhanden sind.

PropertygraphSummaryValueMap (Struktur)

Nutzlast für die Antwort der Eigenschaftsdiagrammübersicht.

Felder

- graphSummary – Dies ist ein PropertygraphSummary-Objekt.

  Die Diagrammübersicht.
- lastStatisticsComputationTime— Dies ist eine SyntheticTimestamp _date_time vom Typ: `string` (eine UTF-8-kodierte Zeichenfolge).

  Zeitstempel nach ISO 8601 für den Zeitpunkt, an dem Neptune die Statistiken zuletzt berechnet hat.
- version – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Version dieser Diagrammübersichtsantwort.

PropertygraphSummary (Struktur)

Die Diagrammübersichts-API gibt eine schreibgeschützte Liste von Knoten- und Kantenbezeichnungen und Eigenschaftsschlüsseln sowie die Anzahl der Knoten, Kanten und Eigenschaften zurück. Siehe Diagrammübersichtantwort für ein Eigenschaftsdiagramm (PG).

Felder

- edgeLabels – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Liste der eindeutigen Kantenbezeichnungen im Diagramm.
- edgeProperties— Das sind LongValuedMap Objekte Es ist ein Map-Array von Schlüssel-Wert-Paaren, wobei:

  Jeder Schlüssel ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Jeder Wert ist ein Long vom Typ `long` (eine 64-Bit-Ganzzahl mit Vorzeichen).

  Eine Liste der eindeutigen Kanteneigenschaften im Diagramm zusammen mit der Zahl der Kanten, bei denen die einzelnen Eigenschaften jeweils verwendet werden.
- edgeStructures – Dies ist ein Array von EdgeStructure-Objekten.

  Dieses Feld ist nur vorhanden, wenn der angeforderte Modus `DETAILED` lautet. Es enthält eine Liste von Kantenstrukturen.
- nodeLabels – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Liste der eindeutigen Knotenbezeichnungen im Diagramm.
- nodeProperties— Das sind LongValuedMap Objekte Es ist ein Map-Array von Schlüssel-Wert-Paaren, wobei:

  Jeder Schlüssel ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Jeder Wert ist ein Long vom Typ `long` (eine 64-Bit-Ganzzahl mit Vorzeichen).

  Die Anzahl der eindeutigen Knoteneigenschaften im Diagramm.
- nodeStructures – Dies ist ein Array von NodeStructure-Objekten.

  Dieses Feld ist nur vorhanden, wenn der angeforderte Modus `DETAILED` lautet. Es enthält eine Liste von Knotenstrukturen.

- numEdgeLabels – Dies ist ein Long vom Typ `long` (eine 64-Bit-Ganzzahl mit Vorzeichen).

  Die Anzahl der eindeutigen Kantenbezeichnungen im Diagramm.
- numEdgeProperties – Dies ist ein Long vom Typ `long` (eine 64-Bit-Ganzzahl mit Vorzeichen).

  Die Anzahl der eindeutigen Kanteneigenschaften im Diagramm.
- numEdges – Dies ist ein Long vom Typ `long` (eine 64-Bit-Ganzzahl mit Vorzeichen).

  Die Anzahl der Kanten im Diagramm.
- numNodeLabels – Dies ist ein Long vom Typ `long` (eine 64-Bit-Ganzzahl mit Vorzeichen).

  Die Anzahl der eindeutigen Knotenbezeichnungen im Diagramm.
- numNodeProperties – Dies ist ein Long vom Typ `long` (eine 64-Bit-Ganzzahl mit Vorzeichen).

  Eine Liste der eindeutigen Knoteneigenschaften im Diagramm zusammen mit der Zahl der Knoten, bei denen die einzelnen Eigenschaften jeweils verwendet werden.
- numNodes – Dies ist ein Long vom Typ `long` (eine 64-Bit-Ganzzahl mit Vorzeichen).

  Die Anzahl der Knoten im Diagramm.
- totalEdgePropertyValues – Dies ist ein Long vom Typ `long` (eine 64-Bit-Ganzzahl mit Vorzeichen).

  Die Gesamtzahl der Nutzungen aller Kanteneigenschaften.
- totalNodePropertyValues – Dies ist ein Long vom Typ `long` (eine 64-Bit-Ganzzahl mit Vorzeichen).

  Die Gesamtzahl der Nutzungen aller Knoteneigenschaften.

Neptune ML Datenverarbeitungs-API

Datenverarbeitungsaktionen:

- Start MLData ProcessingJob (Aktion)
- Liste MLData ProcessingJobs (Aktion)
- Holen MLData ProcessingJob (Aktion)
- Abbrechen MLData ProcessingJob (Aktion)

Allgemeine Strukturen für ML:

- MlResourceDefinition (Struktur)
- MlConfigDefinition (Struktur)

Start MLData ProcessingJob (Aktion)

Der AWS CLI-Name für diese API lautet:`start-ml-data-processing-job`.

Erzeugt einen neuen Neptune ML-Datenverarbeitungsauftrag für die Verarbeitung der aus Neptune für das Training exportierten Graphdaten. Siehe den `dataprocessing`-Befehl.

Wenn dieser Vorgang in einem Neptune-Cluster mit aktivierter IAM-Authentifizierung aufgerufen wird, muss dem IAM-Benutzer oder der Rolle, die die Anfrage gestellt hat, eine Richtlinie angehängt sein, die die IAM-Aktion MLModelDataProcessingJobNeptune-DB:Start in diesem Cluster zulässt.

Anforderung

- configFileName (in der CLI: `--config-file-name`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Datenspezifikationsdatei, die beschreibt, wie die exportierten Graphdaten für das Training geladen werden. Die Datei wird automatisch vom Neptune-Export-Toolkit generiert. Der Standardwert ist `training-data-configuration.json`.
- id (in der CLI: `--id`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine eindeutige Kennung für den neuen Auftrag. Die Standardeinstellung ist eine automatisch generierte UUID.
- inputDataS3Location (in der CLI: `--input-data-s3-location`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die URI des Amazon S3 S3-Standorts SageMaker , an den Sie die für die Ausführung des Datenverarbeitungsjobs erforderlichen Daten herunterladen möchten.
- modelType (in der CLI: `--model-type`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Einer der beiden Modelltypen, die Neptune ML derzeit unterstützt: heterogene Graphmodelle (`heterogeneous`) und Wissensgraph (`kge`). Die Standardeinstellung ist None (Kein). Wenn nicht angegeben, wählt Neptune ML den Modelltyp automatisch auf der Grundlage der Daten aus.

- neptuneIamRoleArn (in der CLI: `--neptune-iam-role-arn`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Amazon-Ressourcenname (ARN) einer IAM-Rolle, die Aufgaben in Ihrem Namen ausführen SageMaker kann. Dies muss in Ihrer DB-Cluster-Parametergruppe aufgeführt sein, sonst tritt ein Fehler auf.

- previousDataProcessingJobId (in der CLI: `--previous-data-processing-job-id`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Auftrags-ID eines abgeschlossenen Datenverarbeitungsauftrags, der auf einer früheren Version der Daten ausgeführt wurde.

- processedDataS3Location (in der CLI: `--processed-data-s3-location`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die URI des Amazon S3 S3-Speicherorts SageMaker , an dem Sie die Ergebnisse eines Datenverarbeitungsauftrags speichern möchten.

- processingInstanceType (in der CLI: `--processing-instance-type`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Typ der ML-Instance, die während der Datenverarbeitung verwendet wird. Sein Speicher sollte groß genug sein, um den verarbeiteten Datensatz aufzunehmen. Die Standardeinstellung ist der kleinste ml.r5-Typ, dessen Arbeitsspeicher zehnmal größer als die Größe der exportierten Graphdaten auf der Festplatte ist.

- processingInstanceVolumeSizeInGB (in der CLI: `--processing-instance-volume-size-in-gb`) – Ganzzahl vom Typ `integer` (32-Bit-Ganzzahl mit Vorzeichen).

  Die Größe des Festplattenvolumens der verarbeitenden Instance. Sowohl Eingabedaten als auch verarbeitete Daten werden auf der Festplatte gespeichert, daher muss die Volumengröße groß genug sein, um beide Datensätze aufzunehmen. Der Standardwert ist 0. Wenn nicht angegeben oder 0, wählt Neptune ML die Volumengröße automatisch auf der Grundlage der Datengröße.

- processingTimeOutInSeconds (in der CLI: `--processing-time-out-in-seconds`) – Ganzzahl vom Typ `integer` (32-Bit-Ganzzahl mit Vorzeichen).

  Timeout in Sekunden für den Datenverarbeitungsauftrag. Der Standardwert ist 86.400 (1 Tag).

- s3OutputEncryptionKMSKey (in der CLI: `--s-3-output-encryption-kms-key`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Der Amazon Key Management Service (Amazon KMS) -Schlüssel, SageMaker mit dem die Ausgabe des Verarbeitungsauftrags verschlüsselt wird. Die Standardeinstellung ist None (Kein).

- sagemakerIamRoleArn (in der CLI: `--sagemaker-iam-role-arn`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der ARN einer IAM-Rolle für die SageMaker Ausführung. Dies muss in Ihrer DB-Cluster-Parametergruppe aufgeführt sein, sonst tritt ein Fehler auf.
- securityGroupIds (in der CLI: `--security-group-ids`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die VPC-Sicherheitsgruppe IDs. Die Standardeinstellung ist None (Kein).
- subnets (in der CLI: `--subnets`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Das IDs der Subnetze in der Neptune VPC. Die Standardeinstellung ist None (Kein).
- volumeEncryptionKMSKey (in der CLI: `--volume-encryption-kms-key`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Amazon Key Management Service (Amazon KMS) -Schlüssel, der zur Verschlüsselung von Daten auf dem Speichervolume SageMaker verwendet wird, das den ML-Compute-Instances zugeordnet ist, die den Trainingsjob ausführen. Die Standardeinstellung ist None (Kein).

Antwort

- arn – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der ARN des Datenverarbeitungsauftrags.
- creationTimeInMillis – eine Long vom Typ: `long` (64-Bit-Ganzzahl mit Vorzeichen).

  Die Zeit, die für die Erstellung des neuen Verarbeitungsauftrags benötigt wurde, in Millisekunden.
- id – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die eindeutige ID des neuen Datenverarbeitungsauftrags.

Fehler

- UnsupportedOperationException

- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

Liste MLData ProcessingJobs (Aktion)

Der AWS CLI-Name für diese API lautet:`list-ml-data-processing-jobs`.

Gibt eine Liste von Neptune ML-Datenverarbeitungsaufträgen aus. Siehe Auflisten aktiver Datenverarbeitungsaufträge mit dem Neptune ML-Datenverarbeitungsbefehl.

Wenn dieser Vorgang in einem Neptune-Cluster mit aktivierter IAM-Authentifizierung aufgerufen wird, muss dem IAM-Benutzer oder der Rolle, die die Anfrage gestellt hat, eine Richtlinie angehängt sein, die die IAM-Aktion MLDataProcessingJobsNeptune-DB:List in diesem Cluster zulässt.

Anforderung

- maxItems(in der CLI:`--max-items`) — eine Liste MLData ProcessingJobsInputMaxItemsInteger vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen), nicht weniger als 1 oder mehr als 1024? st? s

  Die maximale Anzahl der Elemente, die ausgegeben werden sollen (von 1 bis 1024; die Voreinstellung ist 10).
- neptuneIamRoleArn (in der CLI: `--neptune-iam-role-arn`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der ARN einer IAM-Rolle, die Neptune Zugriff auf Amazon S3 S3-Ressourcen SageMaker bietet. Dieser muss in Ihrer DB-Cluster-Parametergruppe aufgeführt sein, andernfalls tritt ein Fehler auf.

Antwort

- ids – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Seite mit einer Liste von Datenverarbeitungsaufträgen. IDs

Fehler

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

Holen MLData ProcessingJob (Aktion)

Der AWS CLI-Name für diese API lautet:`get-ml-data-processing-job`.

Ruft Informationen zu einem angegebenen Datenverarbeitungsauftrag ab. Siehe den `dataprocessing`-Befehl.

Wenn dieser Vorgang in einem Neptune-Cluster mit aktivierter IAM-Authentifizierung aufgerufen wird, muss dem IAM-Benutzer oder der Rolle, die die Anfrage gestellt hat, eine Richtlinie angehängt sein, die die IAM-Aktion MLDataProcessingJobStatusNeptune-DB:Neptune-DB:Get IAM in diesem Cluster zulässt.

Anforderung

- id (in der CLI: `--id`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Die eindeutige Kennung des abzurufenden Datenverarbeitungsauftrags.

- neptuneIamRoleArn (in der CLI: `--neptune-iam-role-arn`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der ARN einer IAM-Rolle, die Neptune Zugriff auf Amazon S3 S3-Ressourcen SageMaker bietet. Dieser muss in Ihrer DB-Cluster-Parametergruppe aufgeführt sein, andernfalls tritt ein Fehler auf.

Antwort

- id – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die eindeutige Kennung dieses Datenverarbeitungsauftrags.

- processingJob – Ein MlResourceDefinition-Objekt.

  Definition des Datenverarbeitungsauftrags.

- status – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Status des Datenverarbeitungsauftrags.

Fehler

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

Abbrechen MLData ProcessingJob (Aktion)

> Der AWS CLI-Name für diese API lautet:`cancel-ml-data-processing-job`.

Bricht einen Neptune ML-Datenverarbeitungsauftrag ab. Siehe den `dataprocessing`-Befehl.

Wenn dieser Vorgang in einem Neptune-Cluster mit aktivierter IAM-Authentifizierung aufgerufen wird, muss dem IAM-Benutzer oder der Rolle, die die Anfrage gestellt hat, eine Richtlinie angehängt sein, die die IAM-Aktion MLDataProcessingJobNeptune-DB:Cancel IAM in diesem Cluster zulässt.

Anforderung

- clean (in der CLI: `--clean`) – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Falls auf `TRUE` gesetzt, gibt dieses Flag an, dass alle Neptune ML S3-Artefakte gelöscht werden sollen, wenn der Auftrag gestoppt wird. Der Standardwert ist `FALSE`.
- id (in der CLI: `--id`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die eindeutige Kennung des Datenverarbeitungsauftrags.
- neptuneIamRoleArn (in der CLI: `--neptune-iam-role-arn`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der ARN einer IAM-Rolle, die Neptune Zugriff auf Amazon S3 S3-Ressourcen SageMaker bietet. Dieser muss in Ihrer DB-Cluster-Parametergruppe aufgeführt sein, andernfalls tritt ein Fehler auf.

Antwort

- status – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Status der Abbruchsanforderung.

Fehler

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException

- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

Allgemeine Strukturen von ML:

MlResourceDefinition (Struktur)

Definiert eine Neptune-ML-Ressource.

Felder

- arn – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der ARN der Ressource.
- cloudwatchLogUrl – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die CloudWatch Protokoll-URL für die Ressource.
- failureReason – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Ursache des Fehlers im Falle eines Fehlers.
- name – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Name der Ressource.
- outputLocation – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Ausgabeort.
- status – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Status der Ressource.

MlConfigDefinition (Struktur)

Enthält eine Neptune ML-Konfiguration.

Felder

- arn – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der ARN der Konfiguration.
- name – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Name der Konfiguration.

Neptune ML-Modelltrainings-API

Trainingsmaßnahmen modellieren:

- Start MLModel TrainingJob (Aktion)
- Liste MLModel TrainingJobs (Aktion)
- Holen MLModel TrainingJob (Aktion)
- Abbrechen MLModel TrainingJob (Aktion)

Modelltrainingsstrukturen:

- CustomModelTrainingParameters (Struktur)

Start MLModel TrainingJob (Aktion)

Der AWS CLI-Name für diese API lautet:`start-ml-model-training-job`.

Erstellt einen neuen Neptune-ML-Modell-Trainingsauftrag. Siehe Modelltraining mit dem `modeltraining`-Befehl.

Wenn dieser Vorgang in einem Neptune-Cluster aufgerufen wird, für den die IAM-Authentifizierung aktiviert ist, muss dem IAM-Benutzer oder der Rolle, die die Anfrage gestellt hat, eine Richtlinie angehängt sein, die die IAM-Aktion MLModelTrainingJobNeptune-DB:Start in diesem Cluster zulässt.

Anforderung

- baseProcessingInstanceType (in der CLI: `--base-processing-instance-type`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Typ der ML-Instance, die bei der Vorbereitung und Verwaltung des Trainings von ML-Modellen verwendet wird. Dies ist eine CPU-Instance, die auf der Grundlage der Speicheranforderungen für die Verarbeitung der Trainingsdaten und des Modells ausgewählt wurde.

- customModelTrainingParameters (in der CLI:`--custom-model-training-parameters`) – Ein CustomModelTrainingParameters-Objekt.

  Die Konfiguration für das Training benutzerdefinierter Modelle. Dies ist ein JSON-Objekt.

- dataProcessingJobId (in der CLI: `--data-processing-job-id`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Auftrags-ID des abgeschlossenen Datenverarbeitungsauftrags, der die Daten erstellt hat, mit denen das Training arbeiten wird.

- enableManagedSpotTraining (in der CLI: `--enable-managed-spot-training`) – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Optimiert die Kosten für das Training von Machine Learning-Modellen mithilfe von Amazon Elastic Compute Cloud-Spot-Instances. Der Standardwert ist `False`.

- id (in der CLI: `--id`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine eindeutige Kennung für den neuen Auftrag. Die Standardeinstellung ist eine automatisch generierte UUID.

- maxHPONumberOfTrainingJobs (in der CLI: `--max-hpo-number-of-training-jobs`) – Ganzzahl vom Typ `integer` (32-Bit-Ganzzahl mit Vorzeichen).

  Maximale Gesamtzahl der Trainingsaufträge, die für den Hyperparameter-Optimierungsauftrag gestartet werden sollen. Der Standardwert ist 2. Neptune ML passt automatisch die Hyperparameter des Machine Learning-Modells an. Um ein Modell mit guter Leistung zu erhalten, verwenden Sie mindestens 10 Aufträge (mit anderen Worten, setzen Sie `maxHP0Number0fTrainingJobs` auf 10). Im Allgemeinen sind die Ergebnisse umso besser, je mehr Optimierungsläufe durchgeführt werden.

- maxHPOParallelTrainingJobs (in der CLI: `--max-hpo-parallel-training-jobs`) – Ganzzahl vom Typ `integer` (32-Bit-Ganzzahl mit Vorzeichen).

Maximale Anzahl parallel Trainingsaufträge, die für den Hyperparameter-Optimierungsauftrag gestartet werden sollen. Der Standardwert ist 2. Die Anzahl der Aufträge, die Sie parallel ausführen können, ist durch die verfügbaren Ressourcen auf Ihrer Trainings-Instance begrenzt.

- neptuneIamRoleArn (in der CLI: `--neptune-iam-role-arn`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der ARN einer IAM-Rolle, die Neptune Zugriff auf Amazon S3 S3-Ressourcen SageMaker bietet. Dies muss in Ihrer DB-Cluster-Parametergruppe aufgeführt sein, sonst tritt ein Fehler auf.

- previousModelTrainingJobId (in der CLI: `--previous-model-training-job-id`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Auftrags-ID eines abgeschlossenen Modelltrainingsauftrags, den Sie auf der Grundlage aktualisierter Daten schrittweise aktualisieren möchten.

- s3OutputEncryptionKMSKey (in der CLI: `--s-3-output-encryption-kms-key`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Amazon Key Management Service (KMS) -Schlüssel, SageMaker mit dem die Ausgabe des Verarbeitungsauftrags verschlüsselt wird. Die Standardeinstellung ist None (Kein).

- sagemakerIamRoleArn (in der CLI: `--sagemaker-iam-role-arn`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der ARN einer IAM-Rolle für die SageMaker Ausführung. Dieser muss in Ihrer DB-Cluster-Parametergruppe aufgeführt sein, sonst tritt ein Fehler auf.

- securityGroupIds (in der CLI: `--security-group-ids`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die VPC-Sicherheitsgruppe IDs. Die Standardeinstellung ist None (Kein).

- subnets (in der CLI: `--subnets`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Das IDs der Subnetze in der Neptune VPC. Die Standardeinstellung ist None (Kein).

- trainingInstanceType (in der CLI: `--training-instance-type`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Typ der ML-Instance, die für das Modelltraining verwendet wird. Alle Neptune ML-Modelle unterstützen CPU-, GPU- und MultiGPU-Training. Der Standardwert ist `ml.p3.2xlarge`. Die

Wahl des richtigen Instance-Typs für das Training hängt von der Art der Aufgabe, der Größe des Graphs und Ihrem Budget ab.

- trainingInstanceVolumeSizeInGB (in der CLI: `--training-instance-volume-size-in-gb`) – Ganzzahl vom Typ `integer` (32-Bit-Ganzzahl mit Vorzeichen).

  Die Größe des Festplattenvolumens der Trainings-Instance. Sowohl die Eingabedaten als auch das Ausgabemodell werden auf der Festplatte gespeichert. Daher muss das Volumen groß genug sein, um beide Datensätze aufzunehmen. Der Standardwert ist 0. Falls nicht angegeben oder 0, wählt Neptune ML eine Festplatten-Volumengröße auf der Grundlage der im Datenverarbeitungsschritt generierten Empfehlung aus.

- trainingTimeOutInSeconds (in der CLI: `--training-time-out-in-seconds`) – Ganzzahl vom Typ `integer` (32-Bit-Ganzzahl mit Vorzeichen).

  Timeout in Sekunden für den Trainingsauftrag. Der Standardwert ist 86.400 (1 Tag).

- trainModelS3Location (in der CLI: `--train-model-s3-location`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Ort in Amazon S3, an dem die Modellartefakte gespeichert werden sollen.

- volumeEncryptionKMSKey (in der CLI: `--volume-encryption-kms-key`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Amazon Key Management Service (KMS) -Schlüssel, der zur Verschlüsselung von Daten auf dem Speichervolume SageMaker verwendet wird, das den ML-Compute-Instances zugeordnet ist, die den Trainingsjob ausführen. Die Standardeinstellung ist None (Kein).

Antwort

- arn – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der ARN des neuen Modell-Trainingsauftrags.

- creationTimeInMillis (in der CLI:) – Long vom Typ `long` (32-Bit-Ganzzahl mit Vorzeichen).

  Die Zeit der Schaffung von Arbeitsplätzen im Rahmen der Modellausbildung in Millisekunden.

- id – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die eindeutige ID des neuen Modell-Trainingsauftrags.

Fehler

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

Liste MLModel TrainingJobs (Aktion)

> Der AWS CLI-Name für diese API lautet:`list-ml-model-training-jobs`.

Listet Neptune ML-Modelltrainingsaufträge auf. Siehe Modelltraining mit dem `modeltraining`-Befehl.

Wenn dieser Vorgang in einem Neptune-Cluster mit aktivierter IAM-Authentifizierung aufgerufen wird, muss dem IAM-Benutzer oder der Rolle, die die Anfrage stellt, eine Richtlinie angehängt sein, die die IAM-Aktion MLModelTrainingJobsNeptune-DB:Neptune-DB:List in diesem Cluster zulässt.

Anforderung

- maxItems(in der CLI:`--max-items`) — eine Liste MLModel TrainingJobsInputMaxItemsInteger vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen), nicht weniger als 1 oder mehr als 1024? st? s

  Die maximale Anzahl der Elemente, die ausgegeben werden sollen (von 1 bis 1024; die Voreinstellung ist 10).
- neptuneIamRoleArn (in der CLI: `--neptune-iam-role-arn`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Der ARN einer IAM-Rolle, die Neptune Zugriff auf Amazon S3 S3-Ressourcen SageMaker bietet. Dieser muss in Ihrer DB-Cluster-Parametergruppe aufgeführt sein, andernfalls tritt ein Fehler auf.

Antwort

- ids – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Seite mit der Liste der Modelausbildungsjobs. IDs

Fehler

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

Holen MLModel TrainingJob (Aktion)

Der AWS CLI-Name für diese API lautet:`get-ml-model-training-job`.

Ruft Informationen über einen Trainingsauftrag für ein Neptune-ML-Modell ab. Siehe Modelltraining mit dem `modeltraining`-Befehl.

Wenn dieser Vorgang in einem Neptune-Cluster mit aktivierter IAM-Authentifizierung aufgerufen wird, muss dem IAM-Benutzer oder der Rolle, die die Anfrage stellt, eine Richtlinie angehängt sein, die die MLModelTrainingJobStatusNeptune-DB:Get IAM-Aktion in diesem Cluster zulässt.

Anforderung

- id (in der CLI: `--id`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die eindeutige Kennung des abzurufenden Modelltrainingsauftrags.

- neptuneIamRoleArn (in der CLI: `--neptune-iam-role-arn`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der ARN einer IAM-Rolle, die Neptune Zugriff auf Amazon S3 S3-Ressourcen SageMaker bietet. Dieser muss in Ihrer DB-Cluster-Parametergruppe aufgeführt sein, andernfalls tritt ein Fehler auf.

Antwort

- hpoJob – Ein MlResourceDefinition-Objekt.

  Der HPO-Auftrag.

- id – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die eindeutige Kennung dieses Modelltrainingsauftrags.

- mlModels – Ein Array mit MlConfigDefinition-Objekten.

  Eine Liste der Konfigurationen der verwendeten ML-Modelle.

- modelTransformJob – Ein MlResourceDefinition-Objekt.

  Der Modell-Transformationsauftrag.

- processingJob – Ein MlResourceDefinition-Objekt.

  Der Datenverarbeitungsauftrag.

- status – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Anzeigen des Status der Trainingsaufträge.

Fehler

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException

- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

Abbrechen MLModel TrainingJob (Aktion)

Der AWS CLI-Name für diese API lautet:`cancel-ml-model-training-job`.

Bricht einen Trainingsauftrag für ein Neptune-ML-Modell ab. Siehe Modelltraining mit dem `modeltraining`-Befehl.

Wenn dieser Vorgang in einem Neptune-Cluster mit aktivierter IAM-Authentifizierung aufgerufen wird, muss dem IAM-Benutzer oder der Rolle, die die Anfrage stellt, eine Richtlinie angehängt sein, die die IAM-Aktion MLModelTrainingJobNeptune-DB:Cancel IAM in diesem Cluster zulässt.

Anforderung

- clean (in der CLI: `--clean`) – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Wenn auf `TRUE` gesetzt, gibt dieses Flag an, dass alle Amazon S3-Artefakte gelöscht werden sollen, wenn der Auftrag gestoppt wird. Der Standardwert ist `FALSE`.
- id (in der CLI: `--id`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die eindeutige Kennung des Modelltrainingsauftrags, der abgebrochen werden soll.
- neptuneIamRoleArn (in der CLI: `--neptune-iam-role-arn`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der ARN einer IAM-Rolle, die Neptune Zugriff auf Amazon S3 S3-Ressourcen SageMaker bietet. Dieser muss in Ihrer DB-Cluster-Parametergruppe aufgeführt sein, andernfalls tritt ein Fehler auf.

Antwort

- status – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Status des Abbruchs.

Fehler

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

Modelltrainingsstrukturen:

CustomModelTrainingParameters (Struktur)

Enthält Trainingsparameter für benutzerdefinierte Modelle. Siehe Benutzerdefinierte Modelle in Neptune ML.

Felder

- sourceS3DirectoryPath – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Pfad zum Amazon S3-Speicherort, an dem sich das Python-Modul befindet, das Ihr Modell implementiert. Muss auf einen gültigen vorhandenen Amazon S3-Speicherort verweisen, der mindestens ein Trainingsskript, ein Transformationsskript und eine `model-hpo-configuration.json`-Datei enthält.
- trainingEntryPointScript – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Der Name des Einstiegspunkts in Ihrem Modul eines Skripts, das Modelltraining durchführt und Hyperparameter als Befehlszeilenargumente verwendet, einschließlich fester Hyperparameter. Der Standardwert ist `training.py`.

- transformEntryPointScript – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Name des Einstiegspunkts in Ihrem Modul eines Skripts, das ausgeführt werden sollte, nachdem das beste Modell aus der Hyperparametersuche identifiziert wurde, um die für die Modellbereitstellung erforderlichen Modellartefakte zu berechnen. Es sollte ohne Befehlszeilenargumente ausgeführt werden können. Die Standardeinstellung ist `transform.py`.

Neptune ML-Modelltransformations-API

Modelltransformationsaktionen:

- Start MLModel TransformJob (Aktion)
- Liste MLModel TransformJobs (Aktion)
- Holen MLModel TransformJob (Aktion)
- Abbrechen MLModel TransformJob (Aktion)

Modelltransformationsstrukturen:

- CustomModelTransformParameters (Struktur)

Start MLModel TransformJob (Aktion)

Der AWS CLI-Name für diese API lautet:`start-ml-model-transform-job`.

Erstellt einen neuen Modelltransformationsauftrag. Siehe Verwenden eines trainierten Modells zum Generieren neuer Modellartefakte.

Wenn dieser Vorgang in einem Neptune-Cluster aufgerufen wird, für den die IAM-Authentifizierung aktiviert ist, muss dem IAM-Benutzer oder der Rolle, die die Anfrage gestellt hat, eine Richtlinie angehängt sein, die die IAM-Aktion MLModelTransformJobNeptune-DB:Start in diesem Cluster zulässt.

Anforderung

- baseProcessingInstanceType (in der CLI: `--base-processing-instance-type`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Typ der ML-Instance, die bei der Vorbereitung und Verwaltung des Trainings von ML-Modellen verwendet wird. Dies ist eine ML-Recheninstance, die auf der Grundlage der Speicheranforderungen für die Verarbeitung der Trainingsdaten und des Modells ausgewählt wurde.
- baseProcessingInstanceVolumeSizeInGB (in der CLI: `--base-processing-instance-volume-size-in-gb`) – Ganzzahl vom Typ `integer` (32-Bit-Ganzzahl mit Vorzeichen).

  Die Größe des Festplattenvolumens der Trainingsinstance in Gigabyte. Der Standardwert ist 0. Sowohl die Eingabedaten als auch das Ausgabemodell werden auf der Festplatte gespeichert. Daher muss die Volumegröße groß genug sein, um beide Datensätze aufzunehmen. Falls nicht angegeben oder 0, wählt Neptune ML eine Festplatten-Volumengröße auf der Grundlage der im Datenverarbeitungsschritt generierten Empfehlung aus.
- customModelTransformParameters(in der CLI:`--custom-model-transform-parameters`) – Ein CustomModelTransformParameters-Objekt.

  Konfigurationsinformationen für eine Modelltransformation unter Verwendung eines benutzerdefinierten Modells. Das `customModelTransformParameters`-Objekt enthält die folgenden Felder, deren Werte mit den gespeicherten Modellparametern aus dem Trainingsjob kompatibel sein müssen:
- dataProcessingJobId (in der CLI: `--data-processing-job-id`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Auftrags-ID eines abgeschlossenen Datenverarbeitungsauftrags. Sie müssen entweder `dataProcessingJobId` und `mlModelTrainingJobId` oder `trainingJobName` einschließen.
- id (in der CLI: `--id`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine eindeutige Kennung für den neuen Auftrag. Die Standardeinstellung ist eine automatisch generierte UUID.
- mlModelTrainingJobId (in der CLI: `--ml-model-training-job-id`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Auftrags-ID eines abgeschlossenen Modelltrainingsauftrags. Sie müssen entweder `dataProcessingJobId` und `mlModelTrainingJobId` oder `trainingJobName` einschließen.

- modelTransformOutputS3Location (in der CLI: `--model-transform-output-s3-location`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Ort in Amazon S3, an dem die Modellartefakte gespeichert werden sollen.
- neptuneIamRoleArn (in der CLI: `--neptune-iam-role-arn`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der ARN einer IAM-Rolle, die Neptune Zugriff auf Amazon S3 S3-Ressourcen SageMaker bietet. Dies muss in Ihrer DB-Cluster-Parametergruppe aufgeführt sein, sonst tritt ein Fehler auf.
- s3OutputEncryptionKMSKey (in der CLI: `--s-3-output-encryption-kms-key`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Amazon Key Management Service (KMS) -Schlüssel, SageMaker mit dem die Ausgabe des Verarbeitungsauftrags verschlüsselt wird. Die Standardeinstellung ist None (Kein).
- sagemakerIamRoleArn (in der CLI: `--sagemaker-iam-role-arn`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der ARN einer IAM-Rolle für die SageMaker Ausführung. Dies muss in Ihrer DB-Cluster-Parametergruppe aufgeführt sein, sonst tritt ein Fehler auf.
- securityGroupIds (in der CLI: `--security-group-ids`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die VPC-Sicherheitsgruppe IDs. Die Standardeinstellung ist None (Kein).
- subnets (in der CLI: `--subnets`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Das IDs der Subnetze in der Neptune VPC. Die Standardeinstellung ist None (Kein).
- trainingJobName (in der CLI: `--training-job-name`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Name eines abgeschlossenen Schulungsjobs. SageMaker Sie müssen entweder `dataProcessingJobId` und `mlModelTrainingJobId` oder `trainingJobName` einschließen.
- volumeEncryptionKMSKey (in der CLI: `--volume-encryption-kms-key`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Amazon Key Management Service (KMS) -Schlüssel, der zur Verschlüsselung von Daten auf dem Speichervolume SageMaker verwendet wird, das den ML-Compute-Instances zugeordnet ist, die den Trainingsjob ausführen. Die Standardeinstellung ist None (Kein).

Antwort

- arn – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der ARN des Modelltransformationsauftrags.
- creationTimeInMillis – Erforderlich: Long vom Typ (eine 64-Bit-Ganzzahl mit Vorzeichen).

  Die Erstellungszeit des Modelltransformationsauftrags in Millisekunden.
- id – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die eindeutige ID des neuen Modelltransformationsauftrags.

Fehler

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

Liste MLModel TransformJobs (Aktion)

Der AWS CLI-Name für diese API lautet:`list-ml-model-transform-jobs`.

Gibt eine Liste von Modelltransformationsjobs zurück IDs. Siehe Verwenden eines trainierten Modells zum Generieren neuer Modellartefakte.

Wenn dieser Vorgang in einem Neptune-Cluster mit aktivierter IAM-Authentifizierung aufgerufen wird, muss dem IAM-Benutzer oder der Rolle, die die Anfrage stellt, eine Richtlinie angehängt sein, die die IAM-Aktion MLModelTransformJobsNeptune-DB:List in diesem Cluster zulässt.

Anforderung

- maxItems(in der CLI:`--max-items`) — eine Liste MLModel TransformJobsInputMaxItemsInteger vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen), nicht weniger als 1 oder mehr als 1024? st? s

  Die maximale Anzahl der Elemente, die ausgegeben werden sollen (von 1 bis 1024; die Voreinstellung ist 10).
- neptuneIamRoleArn (in der CLI: `--neptune-iam-role-arn`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der ARN einer IAM-Rolle, die Neptune Zugriff auf Amazon S3 S3-Ressourcen SageMaker bietet. Dieser muss in Ihrer DB-Cluster-Parametergruppe aufgeführt sein, andernfalls tritt ein Fehler auf.

Antwort

- ids – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Seite aus der Liste der Modelltransformationen. IDs

Fehler

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

Holen MLModel TransformJob (Aktion)

Der AWS CLI-Name für diese API lautet:`get-ml-model-transform-job`.

Ruft Informationen zum angegebenen Modelltransformationsauftrag ab. Siehe Verwenden eines trainierten Modells zum Generieren neuer Modellartefakte.

Wenn dieser Vorgang in einem Neptune-Cluster mit aktivierter IAM-Authentifizierung aufgerufen wird, muss dem IAM-Benutzer oder der Rolle, die die Anfrage stellt, eine Richtlinie angehängt sein, die die MLModelTransformJobStatusNeptune-DB:Get IAM-Aktion in diesem Cluster zulässt.

Anforderung

- id (in der CLI: `--id`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die eindeutige Kennung des abzurufenden Modelltransformationsauftrags.
- neptuneIamRoleArn (in der CLI: `--neptune-iam-role-arn`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der ARN einer IAM-Rolle, die Neptune Zugriff auf Amazon S3 S3-Ressourcen SageMaker bietet. Dieser muss in Ihrer DB-Cluster-Parametergruppe aufgeführt sein, andernfalls tritt ein Fehler auf.

Antwort

- baseProcessingJob – Ein MlResourceDefinition-Objekt.

  Der grundlegende Datenverarbeitungsauftrag.
- id – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die eindeutige Kennung des Modelltransformationsauftrags, der abgerufen werden soll.
- models – Ein Array mit MlConfigDefinition-Objekten.

  Eine Liste der Konfigurationsinformationen für die verwendeten Modelle.
- remoteModelTransformJob – Ein MlResourceDefinition-Objekt.

  Der Remote-Modelltransformationsauftrag.
- status – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Status des Modelltranskriptionsauftrags.

Fehler

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

Abbrechen MLModel TransformJob (Aktion)

> Der AWS CLI-Name für diese API lautet:`cancel-ml-model-transform-job`.

Bricht einen angegebenen Modelltransformationsauftrag ab. Siehe Verwenden eines trainierten Modells zum Generieren neuer Modellartefakte.

Wenn dieser Vorgang in einem Neptune-Cluster aufgerufen wird, für den die IAM-Authentifizierung aktiviert ist, muss dem IAM-Benutzer oder der Rolle, die die Anfrage gestellt hat, eine Richtlinie angehängt sein, die die IAM-Aktion MLModelTransformJobNeptune-DB:Cancel IAM in diesem Cluster zulässt.

Anforderung

- clean (in der CLI: `--clean`) – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Wenn dieses Flag auf `TRUE` gesetzt ist, sollten alle Neptune ML S3-Artefakte gelöscht werden, wenn der Auftrag gestoppt wird. Der Standardwert ist `FALSE`.
- id (in der CLI: `--id`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Die eindeutige ID des Modelltransformationsauftrags, der abgebrochen werden soll.

- neptuneIamRoleArn (in der CLI: `--neptune-iam-role-arn`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der ARN einer IAM-Rolle, die Neptune Zugriff auf Amazon S3 S3-Ressourcen SageMaker bietet. Dieser muss in Ihrer DB-Cluster-Parametergruppe aufgeführt sein, andernfalls tritt ein Fehler auf.

Antwort

- status – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  der Status des Abbruchs.

Fehler

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

Transformationsstrukturen modellieren:

CustomModelTransformParameters (Struktur)

Enthält benutzerdefinierte Transformationsparameter für Modelle. Siehe Verwenden eines trainierten Modells zum Generieren neuer Modellartefakte.

Felder

- sourceS3DirectoryPath – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Pfad zum Amazon S3-Speicherort, an dem sich das Python-Modul befindet, das Ihr Modell implementiert. Muss auf einen gültigen vorhandenen Amazon S3-Speicherort verweisen, der mindestens ein Trainingsskript, ein Transformationsskript und eine `model-hpo-configuration.json`-Datei enthält.
- transformEntryPointScript – Dies ist eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Name des Einstiegspunkts in Ihrem Modul eines Skripts, das ausgeführt werden sollte, nachdem das beste Modell aus der Hyperparametersuche identifiziert wurde, um die für die Modellbereitstellung erforderlichen Modellartefakte zu berechnen. Es sollte ohne Befehlszeilenargumente ausgeführt werden können. Der Standardwert ist `transform.py`.

Neptune ML-Inferenzendpunkt-API

Aktionen am Inferenzendpunkt:

- Erstellen MLEndpoint (Aktion)
- Liste MLEndpoints (Aktion)
- Holen MLEndpoint (Aktion)
- Löschen MLEndpoint (Aktion)

Erstellen MLEndpoint (Aktion)

> Der AWS CLI-Name für diese API lautet:`create-ml-endpoint`.

Erstellt einen neuen Neptune-ML-Inferenzendpunkt, mit dem Sie ein bestimmtes Modell abfragen können, das der Modelltrainingsprozess erstellt hat. Siehe Verwaltung von Inferenzendpunkten mit dem Endpunkt-Befehl.

Wenn dieser Vorgang in einem Neptune-Cluster aufgerufen wird, für den die IAM-Authentifizierung aktiviert ist, muss dem IAM-Benutzer oder der Rolle, die die Anfrage gestellt hat, eine Richtlinie angehängt sein, die die Aktion MLEndpointNeptune-DB:Create IAM in diesem Cluster zulässt.

Anforderung

- id (in der CLI: `--id`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Ein eindeutiger Bezeichner für den neuen Interferenzendpunkt. Die Standardeinstellung ist ein automatisch generierter Name mit Zeitstempel.

- instanceCount (in der CLI: `--instance-count`) – Ganzzahl vom Typ `integer` (32-Bit-Ganzzahl mit Vorzeichen).

  Die Mindestanzahl von EC2 Amazon-Instances, die zur Vorhersage auf einem Endpunkt bereitgestellt werden sollen. Der Standardwert ist 1

- instanceType (in der CLI: `--instance-type`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Typ der Neptune ML-Instance, die für die Online-Wartung verwendet werden soll. Der Standardwert ist `ml.m5.xlarge`. Die Wahl der ML-Instance für einen Inferenzendpunkt hängt vom Aufgabentyp, der Größe des Graphs und Ihrem Budget ab.

- mlModelTrainingJobId (in der CLI: `--ml-model-training-job-id`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Auftrags-ID des abgeschlossenen Modelltrainingsauftrags, mit dem das Modell erstellt wurde, auf das der Inferenzendpunkt verweisen wird. Sie müssen entweder die `mlModelTrainingJobId` oder die `mlModelTransformJobId` angeben.

- mlModelTransformJobId (in der CLI: `--ml-model-transform-job-id`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Auftrags-ID des abgeschlossenen Modelltransformationsauftrags. Sie müssen entweder den `mlModelTrainingJobId` oder den `mlModelTransformJobId` angeben.

- modelName (in der CLI: `--model-name`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Modelltyp für das Training. Standardmäßig basiert das Neptune ML-Modell automatisch auf dem in der Datenverarbeitung verwendeten `modelType`, aber Sie können hier einen anderen Modelltyp angeben. Die Standardeinstellung ist `rgcn` für heterogene Graphen und `kge` für Wissensgraphen. Der einzige gültige Wert für heterogene Graphen ist `rgcn`. Gültige Werte für Wissensgraphen sind: `kge`, `transe`, `distmult`, und `rotate`.

- neptuneIamRoleArn (in der CLI: `--neptune-iam-role-arn`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der ARN einer IAM-Rolle, die Neptune Zugriff auf Amazon S3 SageMaker S3-Ressourcen bietet. Dies muss in Ihrer DB-Cluster-Parametergruppe aufgeführt sein, andernfalls wird ein Fehler ausgegeben.
- update (in der CLI: `--update`) – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Falls auf `true` gesetzt, bedeutet `update`, dass es sich um eine Aktualisierungsanforderung handelt. Der Standardwert ist `false`. Sie müssen entweder den `mlModelTrainingJobId` oder den `mlModelTransformJobId` angeben.
- volumeEncryptionKMSKey (in der CLI: `--volume-encryption-kms-key`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Amazon Key Management Service (Amazon KMS) -Schlüssel, der zur Verschlüsselung von Daten auf dem Speichervolume SageMaker verwendet wird, das den ML-Compute-Instances zugeordnet ist, die den Trainingsjob ausführen. Die Standardeinstellung ist None (Kein).

Antwort

- arn – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der ARN für den neuen Inferenzendpunkt.
- creationTimeInMillis – Dies ist ein Long vom Typ `long` (eine 64-Bit-Ganzzahl mit Vorzeichen).

  Die Erstellungszeit des Endpunkts in Millisekunden.
- id – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die eindeutige ID des neuen Interferenzendpunkts.

Fehler

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException

- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

Liste MLEndpoints (Aktion)

Der AWS CLI-Name für diese API lautet:`list-ml-endpoints`.

Listet die vorhandenen Inferenzendpunkte auf. Siehe Verwaltung von Inferenzendpunkten mit dem Endpunkt-Befehl.

Wenn dieser Vorgang in einem Neptune-Cluster mit aktivierter IAM-Authentifizierung aufgerufen wird, muss dem IAM-Benutzer oder der Rolle, die die Anfrage stellt, eine Richtlinie angehängt sein, die die IAM-Aktion MLEndpointsNeptune-DB:List in diesem Cluster zulässt.

Anforderung

- maxItems(in der CLI:`--max-items`) — eine Liste MLEndpoints InputMaxItemsInteger vom Typ: `integer` (eine 32-Bit-Ganzzahl mit Vorzeichen), nicht weniger als 1 oder mehr als 1024? st? s

  Die maximale Anzahl der auszugebenden Objekte (von 1 bis 1024; der Standardwert ist 10.
- neptuneIamRoleArn (in der CLI: `--neptune-iam-role-arn`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der ARN einer IAM-Rolle, die Neptune Zugriff auf Amazon S3 S3-Ressourcen SageMaker bietet. Dieser muss in Ihrer DB-Cluster-Parametergruppe aufgeführt sein, andernfalls tritt ein Fehler auf.

Antwort

- ids – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Seite aus der Liste der Inferenzendpunkte. IDs

Fehler

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

Holen MLEndpoint (Aktion)

Der AWS CLI-Name für diese API lautet:`get-ml-endpoint`.

Ruft Details zu einem Inferenzendpunkt ab. Siehe Verwaltung von Inferenzendpunkten mit dem Endpunkt-Befehl.

Wenn dieser Vorgang in einem Neptune-Cluster mit aktivierter IAM-Authentifizierung aufgerufen wird, muss dem IAM-Benutzer oder der Rolle, die die Anfrage gestellt hat, eine Richtlinie angehängt sein, die die IAM-Aktion MLEndpointNeptune-DB:Get Status in diesem Cluster zulässt.

Anforderung

- id (in der CLI: `--id`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die eindeutige ID des Interferenzendpunkts.
- neptuneIamRoleArn (in der CLI: `--neptune-iam-role-arn`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der ARN einer IAM-Rolle, die Neptune Zugriff auf Amazon S3 S3-Ressourcen SageMaker bietet. Dieser muss in Ihrer DB-Cluster-Parametergruppe aufgeführt sein, andernfalls tritt ein Fehler auf.

Antwort

- endpoint – Ein MlResourceDefinition-Objekt.

  Die Definition des Endpunkts.
- endpointConfig – Ein MlConfigDefinition-Objekt.

  Die Konfiguration des Endpunkts
- id – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die eindeutige ID des Endpunkts.
- status – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Status des Interferenzendpunkts.

Fehler

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException
- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

Löschen MLEndpoint (Aktion)

Der AWS CLI-Name für diese API lautet:`delete-ml-endpoint`.

Bricht die Erstellung eines Neptune-ML-Inferenzendpunkts ab. Siehe Verwaltung von Inferenzendpunkten mit dem Endpunkt-Befehl.

Wenn dieser Vorgang in einem Neptune-Cluster mit aktivierter IAM-Authentifizierung aufgerufen wird, muss dem IAM-Benutzer oder der Rolle, die die Anfrage gestellt hat, eine Richtlinie angehängt sein, die die Aktion MLEndpointNeptune-DB:Delete IAM in diesem Cluster zulässt.

Anforderung

- clean (in der CLI: `--clean`) – boolescher Wert vom Typ `boolean` (boolescher Wert (wahr oder falsch)).

  Wenn dieses Flag auf `TRUE` gesetzt ist, sollten alle Neptune ML S3-Artefakte gelöscht werden, wenn der Auftrag gestoppt wird. Der Standardwert ist `FALSE`.
- id (in der CLI: `--id`) – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die eindeutige ID des Interferenzendpunkts.
- neptuneIamRoleArn (in der CLI: `--neptune-iam-role-arn`) – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der ARN einer IAM-Rolle, die Neptune Zugriff auf Amazon S3 SageMaker S3-Ressourcen bietet. Dies muss in Ihrer DB-Cluster-Parametergruppe aufgeführt sein, andernfalls wird ein Fehler ausgegeben.

Antwort

- status – eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der Status des Abbruchs.

Fehler

- UnsupportedOperationException
- BadRequestException
- MLResourceNotFoundException
- InvalidParameterException
- ClientTimeoutException
- PreconditionsFailedException
- ConstraintViolationException

- InvalidArgumentException
- MissingParameterException
- IllegalArgumentException
- TooManyRequestsException

API-Ausnahmen für Neptune-Datenebenen

Ausnahmen:

- AccessDeniedException (Struktur)
- BadRequestException (Struktur)
- BulkLoadIdNotFoundException (Struktur)
- CancelledByUserException (Struktur)
- ClientTimeoutException (Struktur)
- ConcurrentModificationException (Struktur)
- ConstraintViolationException (Struktur)
- ExpiredStreamException (Struktur)
- FailureByQueryException (Struktur)
- IllegalArgumentException (Struktur)
- InternalFailureException (Struktur)
- InvalidArgumentException (Struktur)
- InvalidNumericDataException (Struktur)
- InvalidParameterException (Struktur)
- LoadUrlAccessDeniedException (Struktur)
- MalformedQueryException (Struktur)
- MemoryLimitExceededException (Struktur)
- MethodNotAllowedException (Struktur)
- MissingParameterException (Struktur)
- MLResourceNotFoundException (Struktur)
- ParsingException (Struktur)
- PreconditionsFailedException (Struktur)

- QueryLimitExceededException (Struktur)
- QueryLimitException (Struktur)
- QueryTooLargeException (Struktur)
- ReadOnlyViolationException (Struktur)
- S3Exception (Struktur)
- ServerShutdownException (Struktur)
- StatisticsNotAvailableException (Struktur)
- StreamRecordsNotFoundException (Struktur)
- ThrottlingException (Struktur)
- TimeLimitExceededException (Struktur)
- TooManyRequestsException (Struktur)
- UnsupportedOperationException (Struktur)
- UnloadUrlAccessDeniedException (Struktur)

AccessDeniedException (Struktur)

Wird ausgelöst, wenn die Authentifizierung oder Autorisierung fehlschlägt.

Felder

- code – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der HTTP-Statuscode wurde mit der Ausnahme ausgegeben.
- detailedMessage – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer detaillierten Beschreibung des Problems.
- requestId – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die ID der jeweiligen Anforderung.

BadRequestException (Struktur)

Wird ausgelöst, wenn eine Anforderung übermittelt wird, die nicht bearbeitet werden kann.

Felder

- code – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der HTTP-Statuscode wurde mit der Ausnahme ausgegeben.
- detailedMessage – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer detaillierten Beschreibung des Problems.
- requestId – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die ID der fehlerhaften Anfrage.

BulkLoadIdNotFoundException (Struktur)

Wird ausgelöst, wenn eine angegebene Auftrags-ID für das Massenladen nicht gefunden werden kann.

Felder

- code – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der HTTP-Statuscode wurde mit der Ausnahme ausgegeben.
- detailedMessage – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer detaillierten Beschreibung des Problems.
- requestId – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die Auftrags-ID für das Massenladen, die nicht gefunden werden konnte.

CancelledByUserException (Struktur)

Wird ausgelöst, wenn ein Benutzer eine Anforderung abgebrochen hat.

Felder

- code – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Der HTTP-Statuscode wurde mit der Ausnahme ausgegeben.

- detailedMessage – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer detaillierten Beschreibung des Problems.
- requestId – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die ID der jeweiligen Anforderung.

ClientTimeoutException (Struktur)

Wird ausgelöst, wenn bei einer Anfrage im Client das Zeitlimit überschritten wurde.

Felder

- code – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der HTTP-Statuscode wurde mit der Ausnahme ausgegeben.
- detailedMessage – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer detaillierten Beschreibung des Problems.
- requestId – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die ID der jeweiligen Anforderung.

ConcurrentModificationException (Struktur)

Wird ausgelöst, wenn eine Anforderung versucht, Daten zu ändern, die gleichzeitig von einem anderen Prozess geändert werden.

Felder

- code – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der HTTP-Statuscode wurde mit der Ausnahme ausgegeben.
- detailedMessage – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Eine Meldung mit einer detaillierten Beschreibung des Problems.

- requestId – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die ID der jeweiligen Anforderung.

ConstraintViolationException (Struktur)

Wird ausgelöst, wenn ein Wert in einem Anforderungsfeld die erforderlichen Einschränkungen nicht erfüllt hat.

Felder

- code – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der HTTP-Statuscode wurde mit der Ausnahme ausgegeben.
- detailedMessage – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer detaillierten Beschreibung des Problems.
- requestId – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die ID der jeweiligen Anforderung.

ExpiredStreamException (Struktur)

Wird ausgelöst, wenn eine Anforderung versucht, auf einen Stream zuzugreifen, der abgelaufen ist.

Felder

- code – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der HTTP-Statuscode wurde mit der Ausnahme ausgegeben.
- detailedMessage – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer detaillierten Beschreibung des Problems.
- requestId – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Die ID der jeweiligen Anforderung.

FailureByQueryException (Struktur)

Wird ausgelöst, wenn eine Anforderung fehlschlägt.

Felder

- code – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der HTTP-Statuscode wurde mit der Ausnahme ausgegeben.
- detailedMessage – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer detaillierten Beschreibung des Problems.
- requestId – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die ID der jeweiligen Anforderung.

IllegalArgumentException (Struktur)

Wird ausgelöst, wenn ein Argument in einer Anforderung nicht unterstützt wird.

Felder

- code – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der HTTP-Statuscode wurde mit der Ausnahme ausgegeben.
- detailedMessage – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer detaillierten Beschreibung des Problems.
- requestId – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die ID der jeweiligen Anforderung.

InternalFailureException (Struktur)

Wird ausgelöst, wenn die Verarbeitung der Anforderung unerwartet fehlgeschlagen ist.

Felder

- code – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der HTTP-Statuscode wurde mit der Ausnahme ausgegeben.
- detailedMessage – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer detaillierten Beschreibung des Problems.
- requestId – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die ID der jeweiligen Anforderung.

InvalidArgumentException (Struktur)

Wird ausgelöst, wenn ein Argument in einer Anforderung einen ungültigen Wert hat.

Felder

- code – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der HTTP-Statuscode wurde mit der Ausnahme ausgegeben.
- detailedMessage – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer detaillierten Beschreibung des Problems.
- requestId – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die ID der jeweiligen Anforderung.

InvalidNumericDataException (Struktur)

Wird ausgelöst, wenn bei der Bearbeitung einer Anforderung auf ungültige numerische Daten gestoßen wird.

Felder

- code – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der HTTP-Statuscode wurde mit der Ausnahme ausgegeben.
- detailedMessage – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer detaillierten Beschreibung des Problems.
- requestId – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die ID der jeweiligen Anforderung.

InvalidParameterException (Struktur)

Wird ausgelöst, wenn ein Parameterwert nicht gültig ist.

Felder

- code – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der HTTP-Statuscode wurde mit der Ausnahme ausgegeben.
- detailedMessage – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer detaillierten Beschreibung des Problems.
- requestId – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die ID der Anforderung, die einen ungültigen Parameter enthält.

LoadUrlAccessDeniedException (Struktur)

Wird ausgelöst, wenn der Zugriff auf eine angegebene Lade-URL verweigert wird.

Felder

- code – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der HTTP-Statuscode wurde mit der Ausnahme ausgegeben.

- detailedMessage – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer detaillierten Beschreibung des Problems.
- requestId – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die ID der jeweiligen Anforderung.

MalformedQueryException (Struktur)

Wird ausgelöst, wenn eine Abfrage gesendet wird, die syntaktisch falsch ist oder die zusätzliche Überprüfung nicht besteht.

Felder

- code – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der HTTP-Statuscode wurde mit der Ausnahme ausgegeben.
- detailedMessage – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer detaillierten Beschreibung des Problems.
- requestId – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die ID der falsch formatierten Abfrageanforderung.

MemoryLimitExceededException (Struktur)

Wird ausgelöst, wenn eine Anforderung aufgrund unzureichender Speicherressourcen fehlschlägt. Die Anforderung kann erneut versucht werden.

Felder

- code – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der HTTP-Statuscode wurde mit der Ausnahme ausgegeben.
- detailedMessage – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Eine Meldung mit einer detaillierten Beschreibung des Problems.

- requestId – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die ID der Anforderung, die fehlgeschlagen ist.

MethodNotAllowedException (Struktur)

Wird ausgelöst, wenn die von einer Anforderung verwendete HTTP-Methode vom verwendeten Endpunkt nicht unterstützt wird.

Felder

- code – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der HTTP-Statuscode wurde mit der Ausnahme ausgegeben.

- detailedMessage – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer detaillierten Beschreibung des Problems.

- requestId – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die ID der jeweiligen Anforderung.

MissingParameterException (Struktur)

Wird ausgelöst, wenn ein erforderlicher Parameter fehlt.

Felder

- code – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der HTTP-Statuscode wurde mit der Ausnahme ausgegeben.

- detailedMessage – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer detaillierten Beschreibung des Problems.

- requestId – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Die ID der Anforderung, in der der Parameter fehlt.

MLResourceNotFoundException (Struktur)

Wird ausgelöst, wenn eine angegebene Machine Learning-Ressource nicht gefunden werden konnte.

Felder

- code – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der HTTP-Statuscode wurde mit der Ausnahme ausgegeben.
- detailedMessage – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer detaillierten Beschreibung des Problems.
- requestId – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die ID der jeweiligen Anforderung.

ParsingException (Struktur)

Wird ausgelöst, wenn ein Parsing-Problem auftritt.

Felder

- code – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der HTTP-Statuscode wurde mit der Ausnahme ausgegeben.
- detailedMessage – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer detaillierten Beschreibung des Problems.
- requestId – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die ID der jeweiligen Anforderung.

PreconditionsFailedException (Struktur)

Wird ausgelöst, wenn eine Vorbedingung für die Bearbeitung einer Anforderung nicht erfüllt ist.

Felder

- code – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der HTTP-Statuscode wurde mit der Ausnahme ausgegeben.
- detailedMessage – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer detaillierten Beschreibung des Problems.
- requestId – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die ID der jeweiligen Anforderung.

QueryLimitExceededException (Struktur)

Wird ausgelöst, wenn die Anzahl der aktiven Anforderungen das übersteigt, was der Server verarbeiten kann. Die fragliche Anforderung kann wiederholt werden, wenn das System weniger ausgelastet ist.

Felder

- code – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der HTTP-Statuscode wurde mit der Ausnahme ausgegeben.
- detailedMessage – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer detaillierten Beschreibung des Problems.
- requestId – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die ID der Anforderung, die das Limit überschritten hat.

QueryLimitException (Struktur)

Wird ausgelöst, wenn die Größe einer Abfrage das Systemlimit überschreitet.

Felder

- code – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der HTTP-Statuscode wurde mit der Ausnahme ausgegeben.
- detailedMessage – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer detaillierten Beschreibung des Problems.
- requestId – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die ID der Anforderung, die das Limit überschritten hat.

QueryTooLargeException (Struktur)

Wird ausgelöst, wenn der Text einer Abfrage zu groß ist.

Felder

- code – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der HTTP-Statuscode wurde mit der Ausnahme ausgegeben.
- detailedMessage – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer detaillierten Beschreibung des Problems.
- requestId – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die ID der Anforderung, die zu groß ist.

ReadOnlyViolationException (Struktur)

Wird ausgelöst, wenn eine Anforderung versucht, in eine schreibgeschützte Ressource zu schreiben.

Felder

- code – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der HTTP-Statuscode wurde mit der Ausnahme ausgegeben.

- detailedMessage – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer detaillierten Beschreibung des Problems.
- requestId – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die ID der Anforderung, in der der Parameter fehlt.

S3Exception (Struktur)

Wird ausgelöst, wenn ein Problem beim Zugriff auf Amazon S3 auftritt.

Felder

- code – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der HTTP-Statuscode wurde mit der Ausnahme ausgegeben.
- detailedMessage – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer detaillierten Beschreibung des Problems.
- requestId – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die ID der jeweiligen Anforderung.

ServerShutdownException (Struktur)

Wird ausgelöst, wenn der Server während der Bearbeitung einer Anforderung heruntergefahren wird.

Felder

- code – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der HTTP-Statuscode wurde mit der Ausnahme ausgegeben.
- detailedMessage – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer detaillierten Beschreibung des Problems.
- requestId – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Die ID der jeweiligen Anforderung.

StatisticsNotAvailableException (Struktur)

Wird ausgelöst, wenn Statistiken, die zur Erfüllung einer Anforderung benötigt werden, nicht verfügbar sind.

Felder

- code – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der HTTP-Statuscode wurde mit der Ausnahme ausgegeben.
- detailedMessage – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer detaillierten Beschreibung des Problems.
- requestId – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die ID der jeweiligen Anforderung.

StreamRecordsNotFoundException (Struktur)

Wird ausgelöst, wenn von einer Abfrage angeforderte Stream-Datensätze nicht gefunden werden können.

Felder

- code – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der HTTP-Statuscode wurde mit der Ausnahme ausgegeben.
- detailedMessage – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer detaillierten Beschreibung des Problems.
- requestId – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die ID der jeweiligen Anforderung.

ThrottlingException (Struktur)

Wird ausgelöst, wenn die Anforderungsrate den maximalen Durchsatz übersteigt. Anforderungen können erneut versucht werden, wenn diese Ausnahme aufgetreten ist.

Felder

- code – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der HTTP-Statuscode wurde mit der Ausnahme ausgegeben.
- detailedMessage – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer detaillierten Beschreibung des Problems.
- requestId – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die ID der Anforderung, die aus diesem Grund nicht bearbeitet werden konnte.

TimeLimitExceededException (Struktur)

Wird ausgelöst, wenn ein Vorgang das für ihn zulässige Zeitlimit überschreitet.

Felder

- code – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der HTTP-Statuscode wurde mit der Ausnahme ausgegeben.
- detailedMessage – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer detaillierten Beschreibung des Problems.
- requestId – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die ID der Anforderung, die aus diesem Grund nicht bearbeitet werden konnte.

TooManyRequestsException (Struktur)

Wird ausgelöst, wenn die Anzahl der verarbeiteten Anforderungen das Limit überschreitet.

Felder

- code – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der HTTP-Statuscode wurde mit der Ausnahme ausgegeben.
- detailedMessage – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer detaillierten Beschreibung des Problems.
- requestId – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die ID der Anforderung, die aus diesem Grund nicht bearbeitet werden konnte.

UnsupportedOperationException (Struktur)

Wird ausgelöst, wenn eine Anforderung versucht, einen Vorgang zu initiieren, der nicht unterstützt wird.

Felder

- code – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Der HTTP-Statuscode wurde mit der Ausnahme ausgegeben.
- detailedMessage – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer detaillierten Beschreibung des Problems.
- requestId – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die ID der jeweiligen Anforderung.

UnloadUrlAccessDeniedException (Struktur)

Wird ausgelöst, wenn der Zugriff auf eine URL verweigert wird, die ein Entladeziel ist.

Felder

- code – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

Der HTTP-Statuscode wurde mit der Ausnahme ausgegeben.

- detailedMessage – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Eine Meldung mit einer detaillierten Beschreibung des Problems.

- requestId – Erforderlich: eine Zeichenfolge vom Typ `string` (UTF-8-kodierte Zeichenfolge).

  Die ID der jeweiligen Anforderung.

Die vorliegende Übersetzung wurde maschinell erstellt. Im Falle eines Konflikts oder eines Widerspruchs zwischen dieser übersetzten Fassung und der englischen Fassung (einschließlich infolge von Verzögerungen bei der Übersetzung) ist die englische Fassung maßgeblich.