Guide de l’utilisateur

Amazon EKS

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

Amazon EKS: Guide de l’utilisateur

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

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

Table of Contents

..... xxvi
..... xxvii
Présentation d'Amazon EKS 1
Amazon EKS : gestion simplifiée de Kubernetes 1
Création et mise à l'échelle avec Kubernetes : fonctionnalités d'Amazon EKS 2
Fonctions d'Amazon EKS 3
Services connexes 4
Tarification Amazon EKS 5
Cas d'utilisation courants 6
Architecture 8
Plan de contrôle 8
Calcul 8
Fonctionnalités d'EKS 10
Concepts Kubernetes 11
Pourquoi choisir Kubernetes ? 12
Clusters 17
Charges de travail 22
Étapes suivantes 29
Options de déploiement 29
Comprendre les options de déploiement Amazon EKS 29
Amazon EKS dans le cloud 30
Amazon EKS dans votre centre de données ou vos environnements périphériques 31
Amazon EKS Anywhere pour les environnements isolés 31
Amazon EKS Tooling 32
Configuration 34
Étapes suivantes 34
Configuration de l'interface de ligne de commande AWS 34
Pour créer une clé d'accès 35
Pour configurer l'interface de ligne de commande AWS 35
Pour obtenir un jeton de sécurité 36
Pour vérifier l'identité de l'utilisateur 36
Étapes suivantes 37
Configurer kubectl et eksctl 37
Installer ou mettre à jour `kubectl` 37

Étape 1 : vérifier si kubectl est installé 38
Étape 2 : installer ou mettre à niveau kubectl 38
Installer eksctl 53
Étapes suivantes 53
Démarrage rapide 54
Dans ce tutoriel 54
Conditions préalables 54
Configurer le cluster 55
Créez IngressClass 55
Déployer l'application exemple de jeu 2048 56
Conserver les données à l'aide du mode automatique Amazon EKS 60
Nettoyage 62
Découvrez Amazon EKS 63
Présentation 63
Atelier Amazon EKS 64
Tutoriels pratiques de configuration de clusters Amazon EKS 65
Echantillons Amazon EKS 66
Didacticiels AWS 67
Ateliers pour développeurs 67
Ateliers Terraform 67
AWS Formation Amazon EKS 67
Démarrer 69
Création d'un cluster (mode automatique EKS) 69
Créer un cluster (eksctl) 70
Prérequis 71
Étape 1 : créer votre cluster Amazon EKS et votre nœud 71
Étape 2 : Afficher les ressources Kubernetes 73
Étape 3 : supprimer votre cluster et vos nœuds 75
Étapes suivantes 75
Création d'un cluster (console et CLI) 75
Conditions préalables 76
Étape 1 : créer votre cluster Amazon EKS 77
Étape 2 : configurer votre ordinateur de façon à ce qu'il communique avec votre cluster 80
Étape 3 : créer des nœuds 81
Étape 4 : afficher les ressources 83
Étape 5 : Suppression des ressources 84

Étapes suivantes 85
Mode automatique EKS 86
Caractéristiques 86
Composants automatisés 88
Configuration 89
Modèle de responsabilité partagée 90
Créer un cluster 91
CLI eksctl 92
AWS CLI 94
Console de gestion 102
Activer les clusters existants 104
Référence de migration 105
Migration de volumes EBS 106
Migration des équilibreurs de charge 107
Activation sur un cluster 107
Migration à partir de Karpenter 112
Migration à partir des groupes de nœuds gérés 115
Migration à partir de Fargate 116
Exécution de charges de travail 124
Déploiement d'une charge de travail de gonflement 125
Déploiement de l'équilibreur de charge 128
Déployer une charge de travail avec état 134
Déployer une charge de travail accélérée 139
Configurer 146
Création d'une classe de nœuds 150
Création d'un groupe de nœuds 160
Pools de nœuds à capacité statique 167
Création d'une classe de ressources Ingress 174
Créer un service 182
Créez StorageClass 188
Désactivation du mode automatique EKS 198
Mises à jour de la version Kubernetes 200
Vérifier les groupes de nœuds créés 201
Contrôle du déploiement 203
Exécuter les modules complémentaires critiques 205
Utilisation des politiques réseau 207

Baliser les sous-réseaux 218
Génération d'un rapport CIS 220
Chiffrement des volumes racine 223
Mise à jour des contrôles AMI 227
Contrôle des réserves de capacité 231
Utiliser les Zones Locales 236
Sécurité avancée 241
Comment ça marche 243
Instances gérées 243
Identités et accès 248
Réseaux 254
Observabilité 258
Journaux des composants gérés automatiquement par EKS 258
Dépannage 260
Agent de surveillance des nœuds 262
Obtenez la sortie de console d'une instance gérée EC2 à l'aide de la CLI AWS EC2 262
Extraction des journaux des nœuds à l'aide des conteneurs de débogage et de la CLI `kubectl` 263
Afficher les ressources associées au mode automatique EKS dans la AWS console 264
Afficher les erreurs IAM dans votre AWS compte 265
Résolution des problèmes liés à l'échec de la planification des pods sur un nœud du mode automatique 265
Résolution problèmes liés au fait qu'un nœud ne rejoint pas le cluster 265
Partage des volumes entre les pods 268
Affichage des événements Karpenter dans les journaux du plan de contrôle 269
Résolution des problèmes liés aux contrôleurs inclus dans le mode automatique 270
Ressources connexes 223
Notes de mise à jour 271
2 février 2026 271
19 décembre 2025 271
19 novembre 2025 272
19 novembre 2025 272
23 octobre 2025 272
1er octobre 2025 272
30 septembre 2025 272
29 septembre 2025 272

10 septembre 2025 272
24 août 2025 273
15 août 2025 273
6 août 2025 273
30 juin 2025 273
20 juin 2025 273
13 juin 2025 273
30 avril 2025 274
18 avril 2025 274
11 avril 2025 274
4 avril 2025 274
31 mars 2025 274
21 mars 2025 275
14 mars 2025 275
Configuration des clusters 276
Création d'un cluster Auto 276
Conditions préalables 54
Créer un cluster - AWS console 278
Créer un cluster - AWS CLI 283
Étapes suivantes 102
Créer un cluster 289
Conditions préalables 54
Étape 1 : créer un rôle IAM de cluster 290
Étape 2 : créer un cluster 292
Étape 3 : mettre à jour kubeconfig 302
Étape 4 : configuration du cluster 303
Étapes suivantes 102
Création d'un plan de contrôle provisionné 304
Présentation de 100
Cas d'utilisation 157
Niveaux de dimensionnement du plan de contrôle 307
Considérations 107
Commencer à utiliser le plan de contrôle provisionné 312
Informations sur le cluster 320
Types d'informations sur le cluster 320
Considérations 320

Cas d'utilisation 320
Mise en route 322
Afficher les informations sur le cluster 322
Mises à jour de la version Kubernetes 329
Considérations relatives au mode automatique Amazon EKS 330
Résumé 331
Étape 1 : préparer la mise à niveau 331
Étape 2 : examiner les considérations relatives à la mise à niveau 332
Étape 3 : mettre à jour le plan de contrôle du cluster 333
Étape 4 : mettre à jour les composants du cluster 336
Réduire la version Kubernetes d'un cluster Amazon EKS 337
Supprimer un cluster 337
Considérations 107
Etapes prérequises 338
Supprimer le cluster (eksctl) 339
Supprimer le cluster (AWS console) 340
Supprimer le cluster (AWS CLI) 340
Protection des clusters EKS contre toute suppression accidentelle 342
Accès au point de terminaison de cluster 343
Format du point de terminaison du cluster IPv6 343
Format du point de terminaison du cluster IPv4 344
Point de terminaison privé du cluster 345
Modification de l'accès au point de terminaison de cluster 345
Accès à un serveur d'API privé uniquement 349
Configuration de l'accès au point de terminaison 350
Activer la prise en charge de Windows 353
Considérations 107
Conditions préalables 54
Activer la prise en charge de Windows 355
Déployer des pods Windows 358
Prise en charge d'une densité de pods plus élevée sur les nœuds Windows 358
Désactiver la prise en charge Windows 359
Clusters privés 359
Exigences en matière d'architecture de cluster 359
Exigences relatives aux nœuds 360
Exigences relatives aux pods 362

Autoscaling 364
Mode automatique EKS 364
Solutions supplémentaires 365
En savoir plus sur le changement de zone 366
Comprendre le flux de trafic réseau est-ouest entre les pod 366
Exigences relatives au changement de zone EKS 371
Questions fréquentes (FAQ) 380
Ressources supplémentaires 383
Activer le changement de zone 383
Considérations 384
Qu'est-ce qu'Amazon Application Recovery Controller ? 384
Qu'est-ce que le changement de zone ? 385
Qu'est-ce que le changement de zone autoshift ? 385
Que fait EKS pendant un changement automatique ? 385
Enregistrer le cluster EKS auprès d'Amazon Application Recovery Controller (ARC) (AWS console) 385
Étapes suivantes 102
Gestion de l'accès 387
Tâches courantes 387
Contexte 219
Considérations relatives au mode automatique EKS 338
Accès à l'API Kubernetes 389
Associer les identités IAM aux autorisations Kubernetes 390
Définition du mode d'authentification du cluster 391
Entrées d'accès 392
aws-auth ConfigMap 451
Association d'un fournisseur OIDC externe 461
Supprimer le lien avec le fournisseur OIDC 467
Accéder aux ressources du cluster 467
Autorisations requises 468
CloudTrail visibilité 472
Modifier les ressources du cluster 473
Autorisations requises 473
CloudTrail visibilité 472
Accès au cluster avec kubectl 474
Créer le fichier `kubeconfig` automatiquement 475

Accès des charges de travail à AWS 476
Jetons de compte de service 476
Modules complémentaires de cluster 478
Octroi d'autorisations de gestion des identités et des accès AWS aux charges de travail sur les clusters Amazon Elastic Kubernetes Service 479
Informations d'identification auprès de l'IRSA 485
Pod Identity 510
Gestion du Calcul 546
Comparaison des options de calcul 546
Groupes de nœuds gérés 554
Concepts des groupes de nœuds gérés 555
Types de capacité des groupes de nœuds gérés 557
Créer 561
Mettre à jour 573
Mise à jour des informations des comportements 579
Modèles de lancement 583
Suppression 602
Nœuds autogérés 604
Amazon Linux 605
Flacon Rocket 615
Windows 619
Ubuntu Linux 628
Méthodes de mise à jour 631
AWS Fargate 645
Considérations relatives à AWS Fargate 646
Tableau de comparaison de Fargate 649
Mise en route 651
Définir des profils 657
Supprimer les profils 663
Détails de la configuration du pod 664
Événements d'application de correctifs au système d'exploitation 667
Collecter les métriques 670
Journalisation 672
Types d'instances Amazon EC2 684
Nombre maximal de pods recommandé par Amazon EKS pour chaque type d'instance Amazon EC2 687

Comment est déterminé MaxPods 688
Considérations relatives au mode automatique EKS 338
AMI optimisées préconçues 690
AL2 Obsolète de l'AMI 691
Amazon Linux 697
Flacon Rocket 708
Ubuntu Linux 714
Windows 715
État du nœud 788
Agent de surveillance des nœuds 789
Réparation automatique de nœuds 789
Problèmes d'état du nœud 792
Afficher l'état de santé des nœuds 807
Extraction des journaux de nœuds 810
Nœuds hybrides 813
Fonctionnalités 813
Limites 814
Considérations 815
Ressources supplémentaires 815
Prérequis 815
Exécuter des nœuds hybrides 886
Configuration 909
Fonctionnement 997
Nœuds hybrides nodeadm 1041
Résolution des problèmes 1060
Stockage de données d'application 1082
Amazon EBS 1082
Considérations 1083
Conditions préalables 1083
Étape 1 : Créer un rôle IAM 1084
Étape 2 : obtenir le pilote CSI Amazon EBS 1092
Étape 3 : déployer un exemple d'application 1093
Amazon EFS 1093
Considérations 1093
Conditions préalables 1094
Étape 1 : Créer un rôle IAM 1095

Étape 2 : obtenir le pilote CSI Amazon EFS 1101
Étape 3 : Créer un système de fichiers Amazon EFS 1102
Étape 4 : déployer un exemple d'application 1102
Amazon FSx pour Lustre 1102
Déploiement du pilote 1102
Optimiser (EFA) 1111
Optimize (non-EFA) 1127
Amazon FSx pour NetApp ONTAP 1135
Amazon FSx pour OpenZFS 1136
Cache de fichiers Amazon 1136
Mountpoint pour Amazon S3 1136
Considérations 1137
Déploiement du pilote 1137
Supprimer le pilote 1148
Contrôleur d'instantané CSI 1150
Configuration de la mise en réseau 1151
Ajouter un sous-réseau VPC existant à un cluster Amazon EKS à partir de la console de gestion 1151
Exigences requises pour le VPC et les sous-réseaux 1152
Exigences et considérations requises pour le VPC 1152
Exigences et considérations requises pour les sous-réseaux 1154
Exigences et considérations requises pour les sous-réseaux partagés 1161
Création d'un VPC 1162
Conditions préalables 54
Sous-réseaux publics et privés 1163
Sous-réseaux publics uniquement 1165
Sous-réseaux privés uniquement 1167
Exigences de groupe de sécurité 1169
Groupe de sécurité par défaut du cluster 1169
Restriction du trafic de cluster 1171
Groupes de sécurité partagés 1172
Gérer les modules complémentaires de mise en réseau 1173
Modules complémentaires intégrés 1173
Modules complémentaires AWS réseau en option 1174
CNI Amazon VPC 1175
Plug-ins CNI alternatifs 1302

Multus 1304
AWS Contrôleur Load Balancer 1305
CoreDNS 1324
kube-proxy 1344
Charges de travail 1350
Exemple de déploiement (Linux) 1350
Prérequis 54
Création d'un espace de noms 1351
Créer un déploiement Kubernetes 1351
Création d'un service 1353
Vérifier les ressources créées 1354
Exécuter un shell sur un pod 1357
Étapes suivantes 1358
Exemple de déploiement (Windows) 1358
Prérequis 54
Création d'un espace de noms 1351
Créer un déploiement Kubernetes 1351
Création d'un service 1353
Vérifier les ressources créées 1362
Exécuter un shell sur un pod 1357
Étapes suivantes 1366
Vertical Pod Autoscaler (VPA) 1366
Déploiement du VPA (Vertical Pod Autoscaler) 1366
Test de l'installation de votre VPA (Vertical Pod Autoscaler) 1368
Horizontal Pod Autoscaler 1372
Exécution d'une application de test du Horizontal Pod Autoscaler 1373
Répartition de charge réseau 1376
Conditions préalables 54
Considérations 107
Créer un équilibreur de charge de réseau 1380
(Facultatif) Déployer un exemple d'application 1383
Répartition de la charge des applications 1386
Conditions préalables 54
Réutilisation ALBs avec des groupes d'entrée 1390
(Facultatif) Déployer un exemple d'application 1391
Limiter les adresses IP externes des services 1394

Copier une image dans un référentiel 1397
Affichage des registres d'images Amazon 1400
Modules complémentaires Amazon EKS 1404
Considérations 1405
Espace de noms personnalisé pour les modules complémentaires 1406
Considérations relatives au mode automatique Amazon EKS 1407
Support 1408
AWS modules complémentaires 1410
Extensions communautaires 1434
Modules complémentaires Marketplace 1440
Créer un module complémentaire 1458
Mettre à jour un module complémentaire 1472
Vérification de la compatibilité 1481
Supprimer un module complémentaire 1482
Rôles IAM 1486
Champs que vous pouvez personnaliser 1494
Vérification d'images de conteneur 1497
Fonctionnalités d'EKS 1499
Fonctionnalités disponibles 1500
AWS Contrôleurs pour Kubernetes (ACK) 1500
Argo CD 1500
kro (Kube Resource Orchestrator) 1501
Avantages des fonctionnalités d'EKS 1501
Tarification 260
Comment fonctionnent les fonctionnalités d'EKS 1502
Cas d'utilisation courants 1503
Travailler avec les capacités 1505
Ressources relatives aux fonctionnalités d'EKS 1505
Comprendre l'état des capacités 1505
Créez des capacités 1507
Fonctionnalités de la liste 1507
Décrire une fonctionnalité 1509
Mettre à jour la configuration d'une fonctionnalité 1510
Supprimer une fonctionnalité 1511
Étapes suivantes 102
ressources Kubernetes 1512

Ressources sur Argo CD 1513
ressources kro 1514
Ressources de l'ACK 1515
Limites des ressources 1516
Étapes suivantes 102
Considérations 1517
Fonctionnalité : rôles IAM et Kubernetes RBAC 1517
Modèles d'architecture multi-clusters 1518
Comparaison des fonctionnalités d'EKS avec les solutions autogérées 1520
AWS Contrôleurs pour Kubernetes 1523
Comment fonctionne ACK 1523
Avantages de l'ACK 1524
AWS Services pris en charge 1525
Intégration avec d'autres fonctionnalités gérées par EKS 1525
Débuter avec ACK 1526
Créer une fonctionnalité ACK 1526
Concepts d'ACK 1537
Configurer les autorisations 1545
Considérations 1552
Résolution des problèmes 1561
Comparaison avec l'autogestion 1565
Argo CD 1567
Comment fonctionne Argo CD 1567
Avantages d'Argo CD 1569
Intégration à AWS Identity Center 1569
Intégration avec d'autres fonctionnalités gérées par EKS 1525
Commencer à utiliser Argo CD 1570
Création d'une fonctionnalité Argo CD 1570
Concepts d'Argo CD 1581
Configurer les autorisations 1588
Travailler avec Argo CD 1602
Considérations relatives à Argo CD 1646
Résolution des problèmes 1655
Comparaison avec l'autogestion 1660
kro 1666
Comment fonctionne Kro 1666

Les avantages de Kro 1667
Intégration avec d'autres fonctionnalités gérées par EKS 1525
Débuter avec Kro 1668
Créez une capacité KRO 1669
concepts kro 1681
Configurer les autorisations 1688
Considérations 1693
Résolution des problèmes 1699
Comparaison avec l'autogestion 1706
Fonctionnalités de résolution des problèmes 1707
Approche générale du dépannage 1708
Vérifier l'état des capacités 1708
Statuts de capacité communs 1709
Vérifier l'état des ressources Kubernetes 1709
Vérifiez les autorisations IAM et l'accès au cluster 1710
Résolution des problèmes spécifiques aux capacités 1712
Problèmes courants liés à toutes les fonctionnalités 1712
Étapes suivantes 102
Gestion du cluster 1714
Suivi des coûts 1714
Afficher les coûts par Pod 1715
Installer Kubecost 1716
Accéder au tableau de bord Kubecost 1718
En savoir plus sur Kubecost 1719
Serveur de métriques 1733
Considérations 107
Déployer en tant que module complémentaire communautaire avec Amazon EKS Add-ons 1734
Déployer avec Manifeste 1734
Déployer des applications avec Helm 1735
Balisage de vos ressources 1737
Principes de base des étiquettes 1738
Balisage de vos ressources 1738
Restrictions liées aux étiquettes 1739
Balisage de vos ressources pour facturation 1740
Gestion des étiquettes à l'aide de la console 1741

Gestion des identifications à l'aide de la CLI, de l'API ou de `eksctl` 1742
Quotas de service 1744
Consultez les quotas de service EKS dans le AWS Management Console 1744
Afficher les quotas de service EKS avec la AWS CLI 1744
Service quotas Amazon EKS 1745
AWS Quotas de service Fargate 1746
Sécurité 1747
Bonnes pratiques 1748
Analyse des vulnérabilités 1748
Référence Center for Internet Security (CIS) pour Amazon EKS 1749
Versions de plateforme Amazon EKS 1749
Liste des vulnérabilités du système d'exploitation 1750
Détection de nœuds avec Amazon Inspector 1750
Détection de clusters et de nœuds avec Amazon GuardDuty 1750
Validation de la conformité 1750
Considérations relatives à EKS 1752
Sécurité de l'infrastructure 1752
Résilience 1757
Prévention du problème de l'adjoint confus entre services 1758
Considérations pour Kubernetes 1760
Signature des certificats 1760
Rôles et utilisateurs par défaut 1762
Activer le chiffrement secret 1768
AWS Secrets Manager 1773
Chiffrement d'enveloppe par défaut pour toutes les données de l'API Kubernetes 1773
Considérations relatives à EKS Auto 1781
Sécurité et authentification de l'API 1782
Sécurité du réseau 1782
Sécurité des instances EC2 gérées 1783
Gestion automatisée des ressources 1784
Bonnes pratiques de sécurité 1785
Considérations relatives aux fonctionnalités d'EKS 1785
Capacité : rôle IAM 1785
Entrées d'accès EKS 1786
Autorisations Kubernetes supplémentaires 1788
Autorisations IAM requises par fonctionnalité 1790

Bonnes pratiques de sécurité 1791
Étapes suivantes 102
Référence IAM 1794
Public ciblé 1794
Authentification par des identités 1795
Gestion des accès à l'aide de politiques 1799
Amazon EKS et IAM 1801
Politiques basées sur l'identité 1806
Rôles liés à un service 1814
Rôle IAM d'exécution de pod 1831
Rôle IAM du connecteur 1836
AWS politiques gérées 1840
Résolution des problèmes 1872
Rôle IAM du cluster 1875
Rôle IAM de nœud 1879
Rôle IAM du cluster du mode automatique 1885
Rôle IAM du mode automatique pour les nœuds 1888
Capacité : rôle IAM 1891
Grappes de moniteurs 1898
Surveillance et journalisation sur Amazon EKS 1899
Outils de surveillance et de journalisation Amazon EKS 1900
Tableau de bord d'observabilité 1904
Résumé 1904
État du cluster 1904
Surveillance du plan de contrôle 1905
Informations sur le cluster 1907
Problèmes d'état du nœud 1908
Fonctionnalités d'EKS 1908
Observabilité du réseau de conteneurs 1908
Cas d'utilisation 157
Caractéristiques 86
Mise en route 322
Prérequis et remarques importantes 1911
Utilisation de la AWS CLI, de l'API EKS et de l'API NFM 1912
Utilisation de l'infrastructure en tant que code (IaC) 1914
Fonctionnement 211

Considérations et restrictions 1924
Métriques Prometheus 1925
Étape 1 : activer les métriques Prometheus 1926
Étape 2 : utiliser les métriques Prometheus 1928
Étape 3 : gérer les scrapers Prometheus 1928
Pour déployer à l'aide de Helm 1928
Plan de contrôle 1932
Amazon CloudWatch 1940
Statistiques de base sur Amazon CloudWatch 1940
Opérateur Amazon CloudWatch Observability 1950
journaux de plan de contrôle 1950
Activer ou désactiver les journaux du plan de contrôle 1952
Affichage des journaux de plan de contrôle de cluster 1954
AWS CloudTrail 1956
Références 1956
Entrées du fichier journal 1957
Métriques du groupe Auto Scaling 1960
Opérateur ADOT 1961
Tableau de bord Amazon EKS 1961
Qu'est-ce que le tableau de bord Amazon EKS◦? 1961
Comment le tableau de bord utilise-t-il AWS les Organizations ? 1962
Activez le tableau de bord EKS à l'aide de la AWS console 1963
Affichage du tableau de bord EKS 1964
Configuration du tableau de bord 1965
Désactiver le tableau de bord EKS à l'aide de la AWS console 1966
Résolution des problèmes du tableau de bord EKS 1966
Configuration des organisations 1967
Utilisation avec d'autres services 1973
AWS Backup 1973
AWS CloudFormation 1974
Amazon EKS et modèles CloudFormation AWS 1974
En savoir plus sur CloudFormation AWS 1975
AWS CodeConnections 1975
À utiliser AWS CodeConnections avec Argo CD 1975
Considérations relatives à l'utilisation CodeConnections avec Argo CD 1976
Amazon Detective 1977

Utilisation d'Amazon Detective avec Amazon EKS 1977
Amazon GuardDuty 1978
AWS Local Zones 1979
AWS Hub de Résilience 1980
AWS Secrets Manager 1980
Utiliser AWS Secrets Manager avec Argo CD 1981
Amazon Security Lake 1983
Avantages de l'utilisation de Security Lake avec Amazon Amazon EKS 1983
Activation de Security Lake pour Amazon EKS 1984
Analyse des journaux EKS dans Security Lake 1984
Amazon VPC Lattice 1985
Connecteur Amazon EKS 1986
Considérations relatives à Amazon EKS Connector 1986
Rôles IAM requis pour Amazon EKS Connector 1987
Connecter un cluster 1987
Considérations 1988
Prérequis 1988
Étape 1 : Enregistrement du cluster 1988
Étape 2 : Installation de l'agent `eks-connector` 1991
Étapes suivantes 1993
Accorder l'accès à tous les clusters 1993
Prérequis 1994
Annuler l'enregistrement d'un cluster 1995
Désenregistrer le cluster Kubernetes 1996
Nettoyez les ressources de votre cluster Kubernetes 1997
Dépannage d'EKS Connector 1997
Dépannage de base 1998
Problème Helm : 403 Forbidden 1999
Erreur de console : le cluster est bloqué dans l'état En attente 2000
Erreur de console : l'utilisateur system:serviceaccount:eks-connector:eks-connector ne peut pas usurper l'identité des utilisateurs de ressources dans le groupe d'API à portée du cluster 2000
Erreur de console : […] est interdit : l'utilisateur […] ne peut pas répertorier la ressource […] dans le groupe API à portée du cluster 2001

Erreur de console : Amazon EKS ne peut pas communiquer avec le serveur API de votre cluster Kubernetes. Le cluster doit être dans un état ACTIF pour que la connexion soit réussie. Veuillez réessayer dans quelques minutes. 2002
Les pods du connecteur Amazon EKS sont en boucle de crash 2002
Échec de l'initialisation de eks-connector : InvalidActivation 2002
Le nœud du cluster manque de connectivité sortante 2004
Les pods du connecteur Amazon EKS sont dans l'état `ImagePullBackOff` 2004
Questions fréquentes (FAQ) 2004
Considérations sur la sécurité 2006
Responsabilités AWS 2006
Responsabilités client 2006
Amazon EKS sur AWS Outposts 2008
Quand utiliser chaque option de déploiement 2008
Comparaison des options de déploiement 2009
Exécuter des clusters locaux 2012
Régions AWS prises en charge 2012
Déployer un cluster local 2012
Versions de la plateforme EKS 2024
Créer un VPC et des sous-réseaux 2037
Préparer les déconnexions 2041
Considérations relatives à la capacité 2046
Résolution des problèmes liés à un cluster 2049
Nœuds 2061
`eksctl` 2063
AWS Management Console 2065
IA/ML sur EKS 2071
Pourquoi choisir EKS pour l'IA/ML ? 2071
Principaux cas d'utilisation 2072
Études de cas 2073
Commencez à utiliser Machine learning sur EKS 2074
Inférence en temps réel 2074
Créer un cluster 2074
Inférence LLM avec vLLM 2112
Configuration du cluster 2130
Utiliser le GPU EKS Linux AMIs 2131
Installer le plugin d'appareil pour GPUs 2136

Configurer les AMI GPU Windows 2142
Configurer des clusters d'entraînement avec EFA 2149
Configurer des clusters d'inférence avec Inferentia 2160
Gestion de capacité 2166
Réservation de GPU pour les groupes de nœuds gérés 2167
Réservation de GPU pour les nœuds autogérés 2170
Utiliser le P6e- GB2 00 avec EKS 2174
Recettes 2185
Empêcher la planification de pods sur des nœuds spécifiques 2185
Ressources 2187
Ateliers 2188
Bonnes pratiques 2188
Architectures de référence 2190
Didacticiels 2190
Outils 2193
Protocole de contexte du modèle (MCP) 2193
Présentation de 100
Exemples d'intégration 2195
Mise en route 322
Mise en route 2196
Configuration 2213
Outils 2215
Amazon Q 2226
Comment ça marche 529
Utilisation d'Amazon Q pour le dépannage 2227
Considérations 107
En savoir plus 2228
Gestion des versions 2229
Versions Kubernetes 2229
Versions disponibles bénéficiant du support standard 2230
Versions disponibles bénéficiant du support étendu 2230
Calendrier de publication Amazon EKS Kubernetes 2230
Obtenir des informations de version avec la AWS CLI 2232
Version d'Amazon EKS FAQs 2233
Support étendu d'Amazon EKS FAQs 2235
Versions bénéficiant du support standard 2238

Versions bénéficiant d'un support étendu 2245
Afficher la période de support 2248
Consulter la politique de mise à niveau 2249
Permettre un support étendu 2251
Désactiver le support étendu 2253
Versions de plateforme 2254
Version de Kubernetes 1.35 2255
Version de Kubernetes 1.34 2256
Version de Kubernetes 1.33 2258
Version de Kubernetes 1.32 2262
Version de Kubernetes 1.31 2268
Version de Kubernetes 1.30 2275
Version de Kubernetes 1.29 2282
Obtenir la version actuelle de la plateforme 2292
Changer la version de plateforme 2292
Dépannage 2293
Capacité insuffisante 2293
Les nœuds ne parviennent pas à joindre le cluster 2293
Accès non autorisé ou refusé (kubectl) 2295
hostname doesn't match 2296
getsockopt: no route to host 2297
Instances failed to join the Kubernetes cluster 2297
Codes d'erreurs liées aux groupes de nœuds gérés 2297
Not authorized for images 2302
Le nœud est dans l'état NotReady 2302
Collecteur de journaux EKS 2303
Le réseau d'exécution du conteneur n'est pas prêt 2304
Délai d'expiration de la liaison TLS 2305
InvalidClientTokenId 2306
Les groupes de nœuds doivent correspondre à la version Kubernetes avant de mettre à niveau le plan de contrôle 2307
Lors du lancement de nombreux nœuds, des erreurs Too Many Requests se produisent ... 2307
Réponse d'erreur non autorisée HTTP 401 sur les requêtes du serveur API Kubernetes 2307
La version de la plateforme Amazon EKS est inférieure de plus de deux versions à la version actuelle de la plateforme 2308
Foire aux questions sur l'intégrité du cluster et codes d'erreur avec chemins de résolution 2311

Projets liés à Amazon EKS 2318
Prise en charge des logiciels déployés sur EKS 2318
Outils de gestion 2319
eksctl 2319
AWS Contrôleurs pour Kubernetes 2319
kro (Kube Resource Orchestrator) 2320
Argo CD 2320
Flux CD 2320
CDK pour Kubernetes 2320
Réseaux 2321
Plugin CNI Amazon VPC pour Kubernetes 2321
AWS Contrôleur Load Balancer pour Kubernetes 2321
ExternalDNS 2321
Machine learning 2322
Kubeflow 2322
Auto Scaling 2322
Cluster Autoscaler 2322
Karpenter 2322
Escalator 2323
Contrôle 2323
Prometheus 2323
Intégration continue/déploiement continu 2323
Jenkins X 2323
Nouvelles fonctionnalités et feuille de route 2324
Historique du document 2325
Contribuez 2391
Modifier une seule page 2391
Prérequis 54
Procédure 110
Vue d'ensemble des extraction 2393
Modifiez des fichiers avec GitHub 2394
Conditions préalables 54
Procédure 110
Afficher les commentaires sur le style 2395
Installer Vale 2396
Installer l'extension VS Code Vale 2396

Synchroniser Vale 2397
Afficher les commentaires sur le style dans VS Code 2397
Création d'une page 2397
Création d'une page 2398
Ajout de la page à la navigation 2398
Insérer le lien 2399
Lien vers une page ou une section du Guide de l'utilisateur EKS 2399
Lien vers un autre guide dans les AWS documents 2399
Lien vers une page Web externe 2400
Création avec Amazon Q 2400
Installation d'Amazon Q avec VS Code 2401
Connexion à Amazon Q 2401
Utilisation d'Amazon Q pour créer du contenu 2402
Conseils 2402
Afficher l'aperçu de la demande d'extraction 2402
Afficher l'aperçu 2403
Limitations propres à la version préliminaire 2404
Syntaxe AsciiDoc 2404
Nouvelle page 2405
En-têtes 2405
Formatage du texte 2405
Listes 2405
Liens 2406
Exemples de code 2406
Images 2406
Tables 2406
Alertes 2407
Inclusion d'autres fichiers 2407
Attributs (similaires aux entités) 2407
Procédures 2408

Aidez à améliorer cette page

Les traductions sont fournies par des outils de traduction automatique. En cas de conflit entre le contenu d'une traduction et celui de la version originale en anglais, la version anglaise prévaudra.

Pour contribuer à ce guide de l'utilisateur, cliquez sur le GitHub lien Modifier cette page sur qui se trouve dans le volet droit de chaque page.

Les traductions sont fournies par des outils de traduction automatique. En cas de conflit entre le contenu d'une traduction et celui de la version originale en anglais, la version anglaise prévaudra.

Présentation d'Amazon EKS

> **Tip**
>
> Inscrivez-vous aux prochains ateliers Amazon EKS.

Amazon EKS : gestion simplifiée de Kubernetes

Amazon Elastic Kubernetes Service (EKS) fournit un service Kubernetes entièrement géré qui élimine la complexité de l'exploitation des clusters Kubernetes. Avec EKS, vous pouvez :

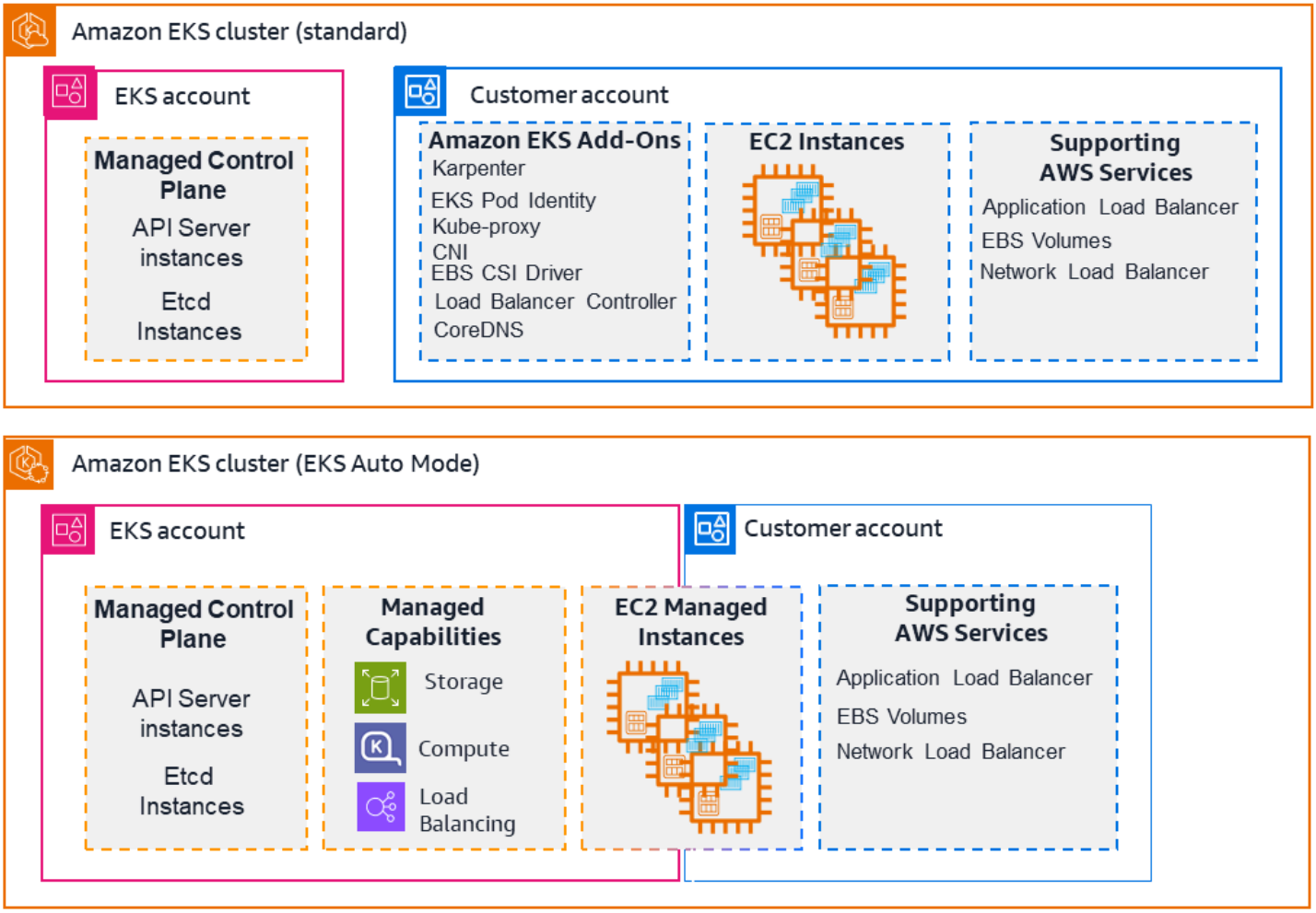
- Déployer des applications plus rapidement avec moins de frais généraux opérationnels
- Évoluer de manière transparente pour répondre aux demandes de charge de travail changeantes
- Améliorez la sécurité grâce à AWS l'intégration et aux mises à jour automatisées
- Choisir entre EKS standard ou le mode automatique EKS entièrement automatisé

Amazon Elastic Kubernetes Service (Amazon EKS) est la plateforme de référence pour l'exécution de clusters Kubernetes, à la fois dans le cloud Amazon Web Services (AWS) et dans vos propres centres de données (EKS Anywhere et Nœuds hybrides Amazon EKS).

Amazon EKS simplifie le fait de créer, de sécuriser et de maintenir des clusters Kubernetes. Il peut être plus rentable de fournir suffisamment de ressources pour répondre aux pics de demande que de maintenir vos propres centres de données. Voici deux des principales approches pour utiliser Amazon EKS :

- Norme EKS : AWS gère le plan de contrôle Kubernetes lorsque vous créez un cluster avec EKS. Les composants qui gèrent les nœuds, planifient les charges de travail, s'intègrent au AWS cloud et stockent et dimensionnent les informations du plan de contrôle pour assurer le bon fonctionnement de vos clusters sont gérés automatiquement pour vous.
- Mode automatique EKS : grâce à la fonctionnalité Mode automatique EKS, EKS étend son contrôle pour gérer également les nœuds (plan de données Kubernetes). Il simplifie la gestion de Kubernetes en provisionnant automatiquement l'infrastructure, en sélectionnant les instances de calcul optimales, en dimensionnant les ressources de manière dynamique, en optimisant continuellement les coûts, en appliquant des correctifs aux systèmes d'exploitation et en intégrant les services de sécurité. AWS

Le schéma suivant illustre la manière dont Amazon EKS intègre vos clusters Kubernetes au AWS cloud, en fonction de la méthode de création de clusters que vous choisissez :



Amazon EKS vous aide à éliminer les frictions et à accélérer le délai de production, à améliorer les performances, la disponibilité et la résilience, ainsi qu'à renforcer la sécurité du système. Pour plus d'informations, consultez Amazon Elastic Kubernetes Service.

Création et mise à l'échelle avec Kubernetes : fonctionnalités d'Amazon EKS

Amazon EKS vous aide non seulement à créer et à gérer des clusters, mais également à créer et à faire évoluer des systèmes d'applications avec Kubernetes. Les fonctionnalités Amazon EKS sont des services de cluster entièrement gérés qui étendent les fonctionnalités de votre cluster grâce à des outils mains libres natifs de Kubernetes, notamment :

- Argo CD : Argo CD fournit un déploiement continu déclaratif GitOps basé sur un déploiement continu pour vos charges de travail, vos AWS ressources et votre infrastructure cloud.
- AWS Contrôleurs pour Kubernetes (ACK) : ACK permet la création native de Kubernetes et la gestion du cycle de vie des AWS ressources, en unifiant l'orchestration des charges de travail et les flux de travail. Infrastructure-as-code
- kro (Kube Resource Orchestrator) : kro étend les fonctionnalités natives de Kubernetes pour simplifier la création, l'orchestration et les compositions de ressources personnalisées, en vous donnant les outils nécessaires pour créer vos propres éléments de base cloud personnalisés.

Les fonctionnalités EKS sont des ressources cloud qui minimisent la charge opérationnelle liée à l'installation, à la maintenance et à la mise à l'échelle de ces composants de base de la plate-forme dans vos clusters, vous permettant ainsi de vous concentrer sur le développement de logiciels plutôt que sur les opérations de plate-forme de cluster.

Pour en savoir plus, veuillez consulter la section Fonctionnalités d'EKS.

Fonctions d'Amazon EKS

Amazon EKS offre les fonctionnalités de haut niveau suivantes :

Interfaces de gestion

EKS propose plusieurs interfaces pour approvisionner, gérer et maintenir des clusters, notamment l'API Amazon EKS AWS Management Console, le CDKSDKs, la CLI AWS CloudFormation, la AWS CLI eksctl et Terraform. Pour plus d'informations, consultez Démarrer et Configuration des clusters.

Outils de contrôle d'accès

EKS s'appuie à la fois sur les fonctionnalités Kubernetes et AWS Identity and Access Management (AWS IAM) pour gérer l'accès des utilisateurs et des charges de travail. Pour plus d'informations, consultez the section called "Accès à l'API Kubernetes" et the section called "Accès des charges de travail à AWS ".

Ressources informatiques

En ce qui concerne les ressources de calcul, EKS propose la gamme complète de types d' EC2 instances Amazon et d' AWS innovations telles que Nitro et Graviton avec Amazon EKS pour vous permettre d'optimiser le calcul en fonction de vos charges de travail. Pour de plus amples informations, veuillez consulter Gestion du Calcul.

Stockage

Le mode automatique EKS crée automatiquement des classes de stockage à l'aide de volumes EBS. À l'aide des pilotes CSI (Container Storage Interface), vous pouvez également utiliser Amazon S3, Amazon EFS, Amazon FSX et Amazon File Cache pour vos besoins de stockage d'applications. Pour de plus amples informations, veuillez consulter Stockage de données d'application.

Sécurité

Le modèle de responsabilité partagée est utilisé en ce qui concerne la sécurité dans Amazon EKS. Pour plus d'informations, consultez Bonnes pratiques en matière de sécurité, Sécurité de l'infrastructure et Sécurité Kubernetes.

Outils de surveillance

Utilisez le tableau de bord d'observabilité pour surveiller les clusters Amazon EKS. Les outils de surveillance incluent Prometheus CloudWatch, Cloudtrail et ADOT Operator. Pour plus d'informations sur les tableaux de bord, les serveurs de métriques et d'autres outils, consultez les sections Coûts des clusters Amazon EKS et Serveur de métriques Kubernetes.

Fonctionnalités du cluster

EKS fournit des fonctionnalités de clusters gérés pour le déploiement continu, la gestion des ressources cloud et la composition des ressources sur la base d'innovations open source. EKS installe Kubernetes APIs dans vos clusters, mais les contrôleurs et autres composants s'exécutent dans EKS et sont entièrement gérés, fournissant des correctifs, une mise à l'échelle et une surveillance automatisés. Pour de plus amples informations, veuillez consulter Fonctionnalités d'EKS.

Compatibilité et prise en charge de Kubernetes

Amazon EKS est certifié conforme à Kubernetes, vous pouvez donc déployer des applications compatibles avec Kubernetes sans refactorisation et utiliser les outils et plug-ins de la communauté Kubernetes. EKS offre à la fois un support standard et un support étendu pour Kubernetes. Pour de plus amples informations, veuillez consulter the section called "Versions Kubernetes".

Services connexes

Services à utiliser avec Amazon EKS

Vous pouvez utiliser d'autres AWS services avec les clusters que vous déployez à l'aide d'Amazon EKS :

Amazon EC2

Bénéficiez d'une capacité de calcul évolutive à la demande avec Amazon EC2.

Amazon EBS

Associez des ressources de stockage en bloc évolutives et hautes performances avec Amazon EBS.

Amazon ECR

Stocker des images de conteneurs en toute sécurité avec Amazon ECR.

Amazon CloudWatch

Surveillez AWS les ressources et les applications en temps réel avec Amazon CloudWatch.

Amazon Prometheus

Suivez les métriques des applications conteneurisées avec le service géré Amazon pour Prometheus.

Elastic Load Balancing

Répartissez le trafic entrant entre plusieurs cibles avec Elastic Load Balancing.

Amazon GuardDuty

Détectez les menaces qui pèsent sur les clusters EKS avec Amazon GuardDuty.

AWS Centre de résilience

Évaluez la résilience des clusters EKS avec AWS Resilience Hub.

Tarification Amazon EKS

Amazon EKS propose une tarification par cluster basée sur la prise en charge de la version du cluster Kubernetes, une tarification pour le mode automatique Amazon EKS et une tarification par vCPU pour les nœuds hybrides Amazon EKS.

Lorsque vous utilisez Amazon EKS, vous payez séparément pour les AWS ressources que vous utilisez pour exécuter vos applications sur les nœuds de travail Kubernetes. Par exemple, si vous exécutez des nœuds de travail Kubernetes en tant qu'instances Amazon EC2 avec des volumes

Amazon EBS et des IPv4 adresses publiques, la capacité de l'instance vous est facturée via Amazon EC2, la capacité du volume via Amazon EBS et l'adresse via IPv4 Amazon VPC.

Consultez les pages de tarification respectives des AWS services que vous utilisez avec vos applications Kubernetes pour obtenir des informations détaillées sur les prix.

- Pour connaître la tarification du cluster Amazon EKS, du mode automatique Amazon EKS, des fonctionnalités Amazon EKS et des nœuds hybrides Amazon EKS, consultez la tarification d'Amazon EKS.
- Pour connaître les EC2 tarifs Amazon, consultez les sections Tarification Amazon EC2 On-Demand et Amazon EC2 Spot Pricing.
- Pour les tarifs AWS Fargate, consultez AWS la section Tarification Fargate.
- Vous pouvez utiliser vos plans de réduction pour les ressources de calcul utilisées dans les clusters Amazon EKS. Pour plus d'informations, consultez Tarification avec Savings Plans.

Cas d'utilisation courants dans Amazon EKS

Amazon EKS propose des services Kubernetes gérés robustes AWS, conçus pour optimiser les applications conteneurisées. Voici quelques-uns des cas d'utilisation les plus courants d'Amazon EKS, qui vous aideront à tirer parti de ses atouts pour répondre à vos besoins spécifiques.

Déploiement d'applications à haute disponibilité

: Elastic Load Balancing vous permet de vous assurer que vos applications sont hautement disponibles sur plusieurs Zones de disponibilité.

Création d'architectures de microservices

: Utilisez les fonctionnalités de découverte de services Kubernetes avec AWS Cloud Map ou Amazon VPC Lattice pour concevoir des systèmes résilients et hautement distribués.

Automatisation des processus de publication de logiciels

: Gérez les pipelines d'intégration continue et de déploiement continu (CI/CD) qui simplifient le processus de création, de test et de déploiement automatisés des applications. Pour un déploiement continu déclaratif, voir Déploiement continu avec Argo CD.

Exécution d'applications sans serveur

: Utilisez AWS Fargate avec Amazon EKS pour exécuter des applications sans serveur, sans avoir à gérer l'infrastructure sous-jacente. Cela signifie que vous pouvez vous concentrer uniquement

sur le développement d'applications, pendant qu'Amazon EKS et Fargate gèrent l'infrastructure sous-jacente.

Exécution de charges de travail liées à l'apprentissage automatique

Amazon EKS est compatible avec les cadres de machine learning les plus courants tels que TensorFlow, MXNet et PyTorch. Grâce à la prise en charge GPU, vous pouvez gérer efficacement des tâches de machine learning même complexes.

Déploiement cohérent sur site et dans le cloud

Pour simplifier l'exécution de Kubernetes dans des environnements sur site, vous pouvez utiliser les mêmes clusters, fonctionnalités et outils Amazon EKS pour exécuter des nœuds autogérés sur AWS Outposts ou utiliser les nœuds hybrides Amazon EKS avec votre propre infrastructure. Dans les environnements autonomes et isolés, vous pouvez recourir utiliser Amazon EKS Anywhere pour automatiser la gestion du cycle de vie des clusters Kubernetes sur votre propre infrastructure.

Exécution d'un traitement par lots rentable et de charges de travail liées aux mégadonnées

Utilisez les Instances Spot pour exécuter vos traitements par lots et charges de travail big data, telles que Apache Hadoop et Spark, à moindre coût, tout en conservant performance et fiabilité. Cela vous permet de profiter de la EC2 capacité Amazon inutilisée à des prix réduits.

Gestion des AWS ressources depuis Kubernetes

Utilisez AWS Controllers for Kubernetes (ACK) pour créer et gérer des AWS ressources directement à partir de votre cluster Kubernetes à l'aide de Kubernetes natif. APIs

Création d'abstractions d'ingénierie de plateforme

Créez des Kubernetes personnalisés APIs qui composent plusieurs ressources en abstractions de niveau supérieur à l'aide de kro (Kube Resource Orchestrator).

Sécurisation des applications et garantie de la conformité

Mettez en œuvre des pratiques de sécurité strictes et maintenez la conformité avec Amazon EKS, qui s'intègre à des services de AWS sécurité tels que AWS Identity and Access Management (IAM), Amazon Virtual Private Cloud (Amazon VPC) AWS et Key Management Service AWS (KMS). Cela garantit la confidentialité et la protection des données conformément aux normes du secteur.

Architecture Amazon EKS

Amazon EKS s'aligne sur l'architecture générale des clusters Kubernetes. Pour plus d'informations, consultez la section Composants Kubernetes dans la documentation Kubernetes. Les sections suivantes résument certains détails supplémentaires relatifs à l'architecture d'Amazon EKS.

Plan de contrôle

Amazon EKS garantit que chaque cluster dispose de son propre plan de contrôle Kubernetes unique. Cette conception permet de séparer l'infrastructure de chaque cluster, sans aucun chevauchement entre les clusters ou les AWS comptes. La configuration inclut :

Composants distribués

Le plan de contrôle positionne au moins deux instances de serveur API et trois instances etcd dans trois zones de AWS disponibilité au sein d'une AWS région.

Performances optimales

Amazon EKS surveille et ajuste activement les instances du plan de contrôle afin de maintenir des performances optimales.

Résilience

Si une instance de plan de contrôle échoue, Amazon EKS la remplace rapidement, en utilisant une autre zone de disponibilité si nécessaire.

Disponibilité constante

En exécutant des clusters sur plusieurs zones de disponibilité, uncontrat de niveau de service (SLA) pour la disponibilité des points de terminaison du serveur API fiable est atteint.

Amazon EKS utilise Amazon Virtual Private Cloud (Amazon VPC) pour limiter le trafic entre les composants du plan de contrôle au sein d'un même cluster. Les composants du cluster ne peuvent pas afficher ni recevoir de communications en provenance d'autres clusters ou AWS comptes, sauf lorsqu'ils sont autorisés par les politiques de contrôle d'accès basé sur les rôles (RBAC) de Kubernetes.

Calcul

Outre le plan de contrôle, un cluster Amazon EKS possède un ensemble de machines de travail appelées nœuds. La sélection du type de nœud de cluster Amazon EKS approprié est essentielle

pour répondre à vos besoins spécifiques et optimiser l'utilisation des ressources. Amazon EKS propose les types de nœuds primaires suivants :

Mode automatique EKS

Le mode automatique EKS étend la AWS gestion au-delà du plan de contrôle pour inclure le plan de données, automatisant ainsi la gestion de l'infrastructure du cluster. Il intègre les fonctionnalités essentielles de Kubernetes en tant que composants intégrés, notamment l'autoscaling informatique, la mise en réseau, l'équilibrage de charge, le DNS, le stockage et la prise en charge des GPU. Le mode automatique EKS gère les nœuds de manière dynamique en fonction des demandes de charge de travail, en utilisant AMIs des fonctionnalités de sécurité immuables et améliorées. Il automatise les mises à jour et les mises à niveau tout en respectant les budgets de perturbation des pods, et comprend des composants gérés qui, autrement, nécessiteraient une gestion supplémentaire. Cette option est idéale pour les utilisateurs qui souhaitent tirer parti de AWS leur expertise day-to-day opérationnelle, minimiser les frais opérationnels et se concentrer sur le développement d'applications plutôt que sur la gestion de l'infrastructure.

Fargate AWS

Fargateest un moteur de calcul sans serveur pour les conteneurs qui élimine le besoin de gérer les instances sous-jacentes. Avec Fargate, vous spécifiez les besoins en ressources de votre application et provisionnez, AWS dimensionnez et entretenez automatiquement l'infrastructure. Cette option est idéale pour les utilisateurs qui privilégient ease-of-use et souhaitent se concentrer sur le développement et le déploiement d'applications plutôt que sur la gestion de l'infrastructure.

Charpentier

Karpenter est un outil Cluster Autoscaler de Kubernetes flexible et performant qui permet d'améliorer la disponibilité des applications et l'efficacité du cluster. Karpenter lance des ressources informatiques adaptées à la charge variable des applications. Cette option permet de fournir des ressources de just-in-time calcul répondant aux exigences de votre charge de travail.

Groupes de nœuds gérés

Les groupes de nœuds gérés sont un mélange d'automatisation et de personnalisation permettant de gérer un ensemble d' EC2 instances Amazon au sein d'un cluster Amazon EKS. AWS prend en charge des tâches telles que l'application de correctifs, la mise à jour et le dimensionnement des nœuds, simplifiant ainsi les aspects opérationnels. En parallèle, les arguments `kubelet` personnalisés sont pris en charge, ce qui ouvre la voie à des politiques avancées de gestion du processeur et de la mémoire. En outre, ils renforcent la sécurité grâce

aux rôles AWS Identity and Access Management (IAM) pour les comptes de service, tout en réduisant le besoin d'autorisations distinctes par cluster.

Nœuds autogérés

Les nœuds autogérés offrent un contrôle total sur vos EC2 instances Amazon au sein d'un cluster Amazon EKS. Vous êtes chargé de gérer, de mettre à l'échelle et de maintenir les nœuds, ce qui vous donne un contrôle total sur l'infrastructure sous-jacente. Cette option convient aux utilisateurs qui ont besoin d'un contrôle granulaire et d'une personnalisation de leurs nœuds et qui sont prêts à investir du temps dans la gestion et la maintenance de leur infrastructure.

Nœuds hybrides Amazon EKS

Avec les nœuds hybrides Amazon EKS, vous pouvez utiliser votre infrastructure sur site et périphérique comme nœuds dans les clusters Amazon EKS. Amazon EKS Hybrid Nodes unifie la gestion de Kubernetes dans tous les environnements et délègue la gestion du plan de contrôle Kubernetes à vos applications sur site et de périphérie. AWS

Fonctionnalités d'EKS

Amazon EKS fournit des fonctionnalités de cluster entièrement gérées, en installant et en gérant Kubernetes APIs (avec des définitions de ressources personnalisées Kubernetes) dans votre cluster, tandis que les contrôleurs d'exploitation et les autres composants de l'infrastructure AWS détenue sont séparés de votre cluster. EKS fournit des correctifs, une mise à l'échelle et une surveillance automatisés de ces fonctionnalités, en gérant entièrement leur cycle de vie afin de réduire la charge de travail liée à l'exploitation des services en cluster pour l'orchestration de la charge de travail, la gestion des AWS ressources, etc.

EKS propose les types de fonctionnalités suivants :

AWS Contrôleurs pour Kubernetes (ACK)

AWS Controllers for Kubernetes (ACK) vous permet de gérer les AWS ressources à l'aide de Kubernetes APIs, ce qui vous permet de définir des compartiments S3, des bases de données RDS, des rôles IAM et d'autres ressources en tant que ressources personnalisées Kubernetes. AWS Vous pouvez gérer les AWS ressources parallèlement à vos charges de travail Kubernetes à l'aide des mêmes outils et flux de travail, avec la prise en charge de plus de 50 AWS services, notamment S3, RDS, DynamoDB et Lambda.

Argo CD

Argo CD implémente un déploiement continu GitOps basé sur les charges de travail de vos applications, les AWS ressources et la configuration de votre cluster, en utilisant les référentiels Git comme source de vérité. Argo CD synchronise automatiquement vos clusters avec vos référentiels Git et détecte les dérives, en se réconciliant continuellement pour garantir que les applications et les ressources déployées correspondent à l'état souhaité en matière de contrôle de version. Vous pouvez utiliser Argo CD pour gérer des applications sur un cluster donné, ou déployer et gérer des applications sur plusieurs clusters à partir d'une seule ressource Argo CD, avec un déploiement automatique à partir des référentiels Git chaque fois que des modifications sont validées.

kro (Kube Resource Orchestrator)

kro (Kube Resource Orchestrator) vous permet de créer des Kubernetes personnalisés APIs qui composent plusieurs ressources en abstractions de haut niveau, permettant aux équipes de la plateforme de définir des modèles réutilisables pour des combinaisons de ressources communes. Cela permet aux équipes de la plateforme de fournir des fonctionnalités de libre-service avec des garde-fous appropriés, permettant aux développeurs de fournir une infrastructure complexe à l'aide d'une infrastructure simple et spécialement conçue APIs tout en respectant les normes organisationnelles et les meilleures pratiques.

Concepts Kubernetes

Amazon Elastic Kubernetes Service (Amazon EKS) est un service géré AWS basé sur le projet open source Kubernetes. Bien qu'il y ait certaines choses à savoir sur l'intégration du service Amazon EKS avec Cloud AWS (en particulier lorsque vous créez un cluster Amazon EKS pour la première fois), une fois qu'il est opérationnel, vous utilisez votre cluster Amazon EKS de la même manière que n'importe quel autre cluster Kubernetes. Pour commencer à gérer des clusters Kubernetes et à déployer des charges de travail, vous devez donc avoir au moins une compréhension de base des concepts Kubernetes.

Cette page divise les concepts Kubernetes en trois sections : the section called "Pourquoi choisir Kubernetes ?", the section called "Clusters", et the section called "Charges de travail". La première section décrit l'intérêt d'utiliser un service Kubernetes, en particulier sous la forme d'un service géré tel qu'Amazon EKS. La section « Charges de travail » traite de la manière dont les applications Kubernetes sont créées, stockées, exécutées et gérées. La section Clusters présente les différents

composants qui constituent les clusters Kubernetes et vos responsabilités en matière de création et de maintenance des clusters Kubernetes.

Rubriques

- Pourquoi choisir Kubernetes ?
- Clusters
- Charges de travail
- Étapes suivantes

Au fur et à mesure que vous parcourez ce contenu, des liens vous mèneront vers des descriptions plus détaillées des concepts Kubernetes dans la documentation Amazon EKS et Kubernetes, au cas où vous souhaiteriez approfondir l'un des sujets abordés ici. Pour plus d'informations sur la manière dont Amazon EKS implémente le plan de contrôle et les fonctionnalités de calcul Kubernetes, consultez the section called "Architecture".

Pourquoi choisir Kubernetes ?

Kubernetes a été conçu pour améliorer la disponibilité et la capacité de mise à l'échelle lors de l'exécution d'applications conteneurisées critiques et de qualité production. Plutôt que de simplement exécuter Kubernetes sur une seule machine (bien que cela soit possible), Kubernetes atteint ces objectifs en vous permettant d'exécuter des applications sur des ensembles d'ordinateurs qui peuvent s'étendre ou se réduire pour répondre à la demande. Kubernetes comprend des fonctionnalités qui vous facilitent la tâche pour :

- Déployer des applications sur plusieurs machines (à l'aide de conteneurs déployés dans des pods)
- Surveiller l'état des conteneurs et redémarrer ceux qui ont échoué
- Augmenter ou réduire la taille des conteneurs en fonction de la charge
- Mettre à jour les conteneurs avec les nouvelles versions
- Répartir les ressources entre les conteneurs
- Équilibrer le trafic entre les machines

Le fait que Kubernetes automatise ce type de tâches complexes permet aux développeurs d'applications de se concentrer sur la création et l'amélioration de leurs charges de travail applicatives, plutôt que de se soucier de l'infrastructure. Le développeur crée généralement des fichiers de configuration, au format YAML, qui décrivent l'état souhaité de l'application. Cela

peut inclure les conteneurs à exécuter, les limites de ressources, le nombre de réplicas de pods, l'allocation de CPU/mémoire, les règles d'affinité, etc.

Attributs de Kubernetes

Pour atteindre ses objectifs, Kubernetes présente les caractéristiques suivantes :

- Conteneurisé — Kubernetes est un outil d'orchestration de conteneurs. Pour utiliser Kubernetes, vous devez d'abord conteneuriser vos applications. Selon le type d'application, il peut s'agir d'un ensemble de microservices, de tâches par lots ou d'autres formes. Vos applications peuvent alors tirer parti d'un workflow Kubernetes qui englobe un vaste écosystème d'outils, où les conteneurs peuvent être stockés sous forme d'images dans un registre de conteneurs, déployés sur un cluster Kubernetes et exécutés sur un nœud disponible. Vous pouvez créer et tester des conteneurs individuels sur votre ordinateur local avec Docker ou une autre exécution de conteneurs, avant de les déployer sur votre cluster Kubernetes.
- Évolutif : si la demande pour vos applications dépasse la capacité des instances en cours d'exécution de ces applications, Kubernetes est en mesure d'augmenter verticalement. Au besoin, Kubernetes peut déterminer si les applications ont besoin de plus de CPU ou de mémoire et réagir en augmentant automatiquement la capacité disponible ou en utilisant davantage la capacité existante. La mise à l'échelle peut être effectuée au niveau du pod, s'il y a suffisamment de puissance de calcul disponible pour exécuter davantage d'instances de l'application (mise à l'échelle automatique horizontale des pods), ou au niveau du nœud, si davantage de nœuds doivent être activés pour gérer l'augmentation de capacité (Cluster Autoscaler ou Karpenter). Comme la capacité n'est plus nécessaire, ces services peuvent supprimer les pods inutiles et arrêter les nœuds superflus.
- Disponible : si une application ou un nœud devient défaillant ou indisponible, Kubernetes peut déplacer les charges de travail en cours vers un autre nœud disponible. Vous pouvez forcer la question en supprimant simplement une instance en cours d'exécution d'une charge de travail ou d'un nœud qui exécute vos charges de travail. En résumé, cela signifie que les charges de travail peuvent être transférées vers d'autres emplacements si elles ne peuvent plus être exécutées là où elles se trouvent.
- Déclaratif — Kubernetes utilise la réconciliation active pour vérifier en permanence que l'état que vous déclarez pour votre cluster correspond à l'état réel. En appliquant des objets Kubernetes à un cluster, généralement via des fichiers de configuration au format YAML, vous pouvez, par exemple, demander le démarrage des charges de travail que vous souhaitez exécuter sur votre cluster. Vous pouvez modifier ultérieurement les configurations pour, par exemple, utiliser une version plus récente d'un conteneur ou allouer davantage de mémoire. Kubernetes fera le nécessaire pour

établir l'état souhaité. Cela peut inclure la mise en service ou la mise hors service de nœuds, l'arrêt et le redémarrage de charges de travail, ou le retrait de conteneurs mis à jour.

- Composable : Étant donné qu'une application se compose généralement de plusieurs composants, vous souhaitez pouvoir gérer ensemble un ensemble de ces composants (souvent représentés par plusieurs conteneurs). Alors que Docker Compose offre un moyen de le faire directement avec Docker, la commande Kubernetes Kompose peut vous aider à le faire avec Kubernetes. Consultez Traduire un fichier Docker Compose vers Kubernetes Resources pour obtenir un exemple de la manière de procéder.
- Extensible : Contrairement aux logiciels propriétaires, le projet open source Kubernetes est conçu pour vous permettre d'étendre Kubernetes comme vous le souhaitez afin de répondre à vos besoins. Les API et les fichiers de configuration sont ouverts aux modifications directes. Les tiers sont encouragés à écrire leurs propres contrôleurs afin d'étendre à la fois l'infrastructure et les fonctionnalités Kubernetes destinées aux utilisateurs finaux. Les webhooks vous permettent de configurer des règles de cluster afin d'appliquer des politiques et de vous adapter à des conditions changeantes. Pour plus d'idées sur la manière d'étendre les clusters Kubernetes, consultez la section Étendre Kubernetes.
- Portable : De nombreuses organisations ont standardisé leurs opérations sur Kubernetes, car cela leur permet de gérer tous leurs besoins applicatifs de la même manière. Les développeurs peuvent utiliser les mêmes pipelines pour créer et stocker des applications conteneurisées. Ces applications peuvent ensuite être déployées sur des clusters Kubernetes fonctionnant sur site, dans le cloud, sur des terminaux de point de vente dans les restaurants ou sur des appareils IoT dispersés sur les sites distants de l'entreprise. Sa nature open source permet aux utilisateurs de développer ces distributions Kubernetes spécialisées, ainsi que les outils nécessaires à leur gestion.

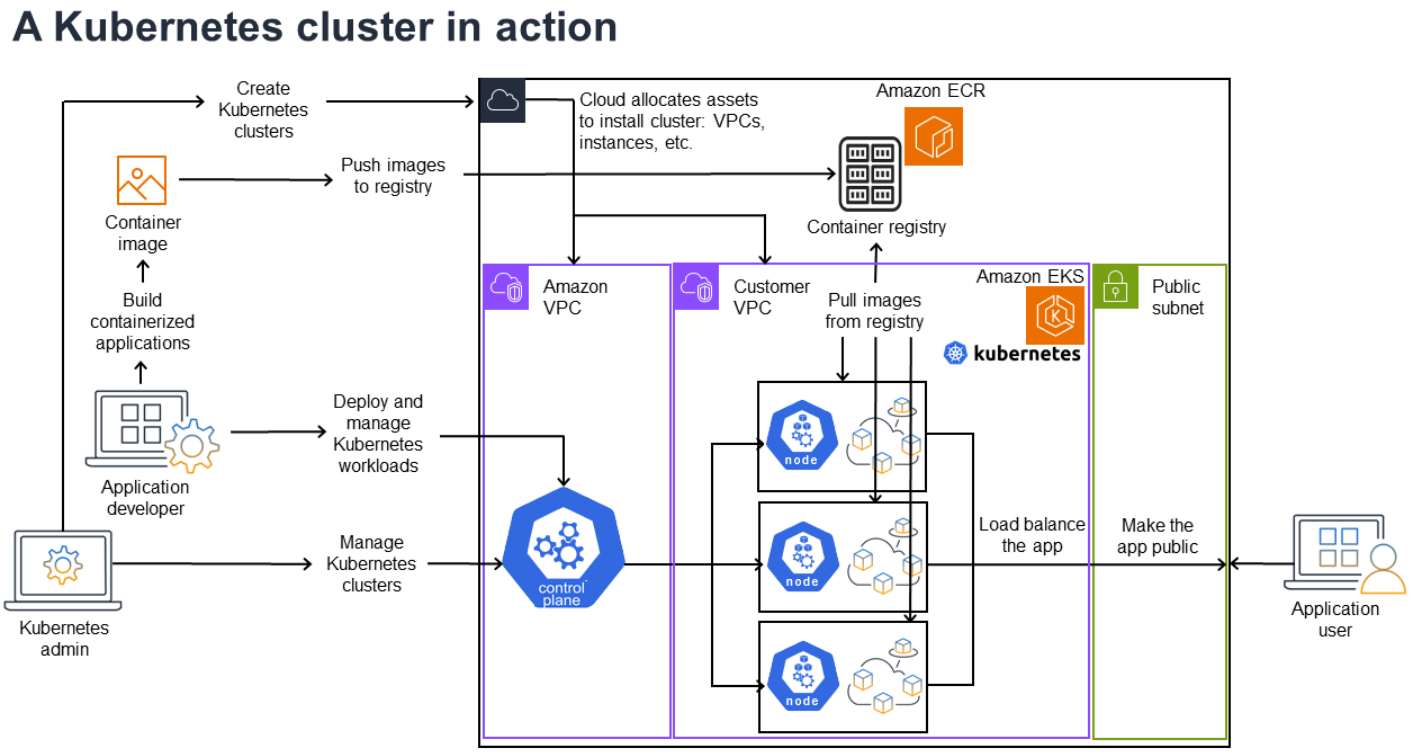
Gestion Kubernetes

Le code source de Kubernetes est disponible gratuitement. Vous pouvez donc installer et gérer Kubernetes vous-même avec votre propre équipement. Cependant, la gestion autonome de Kubernetes nécessite une expertise opérationnelle approfondie et demande du temps et des efforts pour être maintenue. Pour ces raisons, la plupart des personnes qui déploient des charges de travail de production choisissent un fournisseur de cloud (tel qu'Amazon EKS) ou un fournisseur sur site (tel qu'Amazon EKS Anywhere) disposant de sa propre distribution Kubernetes testée et bénéficiant du soutien d'experts Kubernetes. Cela vous permet de vous décharger d'une grande partie des tâches fastidieuses et répétitives nécessaires à la maintenance de vos clusters, notamment :

- Matériel : Si vous ne disposez pas du matériel nécessaire pour exécuter Kubernetes conformément à vos exigences, un fournisseur de services cloud tel que AWS Amazon EKS peut vous permettre de réduire vos coûts initiaux. Avec Amazon EKS, cela signifie que vous pouvez utiliser les meilleures ressources cloud proposées par AWS, notamment des instances informatiques (Amazon Elastic Compute Cloud), votre propre environnement privé (Amazon VPC), la gestion centralisée des identités et des autorisations (IAM) et le stockage (Amazon EBS). AWS gère les ordinateurs, les réseaux, les centres de données et tous les autres composants physiques nécessaires au fonctionnement de Kubernetes.c De même, vous n'avez pas besoin de planifier votre centre de données pour qu'il puisse gérer la capacité maximale lors des jours où la demande est la plus forte. Pour Amazon EKS Anywhere ou d'autres clusters Kubernetes sur site, vous êtes responsable de la gestion de l'infrastructure utilisée dans vos déploiements Kubernetes, mais vous pouvez toujours compter sur AWS pour vous aider à maintenir Kubernetes à jour.
- Gestion du plan de contrôle : Amazon EKS gère la sécurité et la disponibilité du plan de contrôle Kubernetes hébergé par AWS, qui est chargé de planifier les conteneurs, de gérer la disponibilité des applications et d'autres tâches clés, afin que vous puissiez vous concentrer sur vos charges de travail applicatives. Si votre cluster tombe en panne, AWS doit disposer des moyens nécessaires pour le restaurer et le remettre en état de fonctionnement. Pour Amazon EKS Anywhere, vous devez gérer vous-même le plan de contrôle.
- Mises à niveau testées : Lorsque vous mettez à niveau vos clusters, vous pouvez compter sur Amazon EKS ou Amazon EKS Anywhere pour fournir des versions testées de leurs distributions Kubernetes.
- Extensions : Il existe des centaines de projets conçus pour étendre et fonctionner avec Kubernetes que vous pouvez ajouter à l'infrastructure de votre cluster ou utiliser pour faciliter l'exécution de vos charges de travail. Au lieu de créer et de gérer vous-même ces modules complémentaires, AWS fournit the section called "Modules complémentaires Amazon EKS" que vous pouvez utiliser avec vos clusters. Amazon EKS Anywhere propose des packages sélectionnés qui incluent des versions de nombreux projets open source populaires. Vous n'avez donc pas à créer vous-même le logiciel ni à gérer les correctifs de sécurité critiques, les corrections de bogues ou les mises à niveau. De même, si les valeurs par défaut répondent à vos besoins, il est courant que très peu de configuration de ces modules complémentaires soient nécessaires. Consultez the section called "Extension des clusters" pour plus de détails sur l'extension de votre cluster avec des modules complémentaires.

Kubernetes en action

Le diagramme suivant présente les principales activités que vous devrez réaliser en tant qu'administrateur Kubernetes ou développeur d'applications pour créer et utiliser un cluster Kubernetes. Ce processus illustre comment les composants Kubernetes interagissent entre eux, en utilisant le cloud AWS comme exemple de fournisseur de cloud sous-jacent.



Un administrateur Kubernetes crée le cluster Kubernetes à l'aide d'un outil spécifique au type de fournisseur sur lequel le cluster sera construit. Cet exemple utilise le cloud AWS comme fournisseur, qui propose le service Kubernetes géré appelé Amazon EKS. Le service géré alloue automatiquement les ressources nécessaires à la création du cluster, notamment la création de deux nouveaux clouds privés virtuels (Amazon VPC) pour le cluster, la configuration du réseau et le mappage des autorisations Kubernetes directement dans les nouveaux VPC pour la gestion des actifs cloud. Le service géré veille également à ce que les services du plan de contrôle disposent d'emplacements pour s'exécuter et alloue zéro ou plusieurs instances Amazon EC2 en tant que nœuds Kubernetes pour l'exécution des charges de travail. AWS gère lui-même un Amazon VPC pour le plan de contrôle, tandis que l'autre Amazon VPC contient les nœuds clients qui exécutent les charges de travail.

À l'avenir, la plupart des tâches de l'administrateur Kubernetes seront effectuées à l'aide d'outils Kubernetes tels que `kubectl`. Cet outil envoie les demandes de services directement au plan de contrôle du cluster. La manière dont les requêtes et les modifications sont apportées au cluster est alors très similaire à celle que vous utiliseriez sur n'importe quel cluster Kubernetes.

Un développeur d'applications souhaitant déployer des charges de travail sur ce cluster peut effectuer plusieurs tâches. Le développeur doit intégrer l'application dans une ou plusieurs images de conteneur, puis transférer ces images vers un registre de conteneurs accessible au cluster Kubernetes. AWS propose Amazon Elastic Container Registry (Amazon ECR) à cette fin.

Pour exécuter l'application, le développeur peut créer des fichiers de configuration au format YAML qui indiquent au cluster comment exécuter l'application, notamment quels conteneurs extraire du registre et comment les intégrer dans des pods. Le plan de contrôle (planificateur) planifie les conteneurs sur un ou plusieurs nœuds et le moteur d'exécution des conteneurs sur chaque nœud extrait et exécute les conteneurs nécessaires. Le développeur peut également configurer un Application Load Balancer afin d'équilibrer le trafic vers les conteneurs disponibles s'exécutant sur chaque nœud et d'exposer l'application afin qu'elle soit accessible sur un réseau public depuis l'extérieur. Une fois tout cela fait, une personne souhaitant utiliser l'application peut se connecter au point de terminaison de l'application pour y accéder.

Les sections suivantes présentent en détail chacune de ces fonctionnalités, du point de vue des clusters et des charges de travail Kubernetes.

Clusters

Si votre travail consiste à démarrer et à gérer des clusters Kubernetes, vous devez savoir comment ces clusters sont créés, améliorés, gérés et supprimés. Vous devez également connaître les composants qui constituent un cluster et ce que vous devez faire pour les entretenir.

Les outils de gestion des clusters gèrent le chevauchement entre les services Kubernetes et le fournisseur de matériel sous-jacent. Pour cette raison, l'automatisation de ces tâches est généralement effectuée par le fournisseur Kubernetes (tel qu'Amazon EKS ou Amazon EKS Anywhere) à l'aide d'outils spécifiques au fournisseur. Par exemple, pour démarrer un cluster Amazon EKS, vous pouvez utiliser `eksctl create cluster`, tandis que pour Amazon EKS Anywhere, vous pouvez utiliser `eksctl anywhere create cluster`. Notez que, bien que ces commandes créent un cluster Kubernetes, elles sont spécifiques au fournisseur et ne font pas partie du projet Kubernetes lui-même.

Outils de création et de gestion de clusters

Le projet Kubernetes propose des outils permettant de créer manuellement un cluster Kubernetes. Si vous souhaitez installer Kubernetes sur une seule machine ou exécuter le plan de contrôle sur une machine et ajouter des nœuds manuellement, vous pouvez utiliser des outils CLI tels que kind, minikube ou kubeadm, répertoriés sous Outils d'installation de Kubernetes. Pour simplifier et automatiser le cycle de vie complet de la création et de la gestion des clusters, il est beaucoup plus facile d'utiliser des outils pris en charge par un fournisseur Kubernetes établi, tel qu'Amazon EKS ou Amazon EKS Anywhere.

Dans AWS Cloud, vous pouvez créer des clusters Amazon EKS à l'aide d'outils CLI, tels que eksctl, ou d'outils plus déclaratifs, tels que Terraform (voir Amazon EKS Blueprints pour Terraform). Vous pouvez également créer un cluster à partir du fichier AWS Management Console. Consultez les fonctionnalités Amazon EKS pour obtenir la liste de ce que vous offre Amazon EKS. Les responsabilités Kubernetes prises en charge par Amazon EKS pour vous comprennent :

- Plan de contrôle géré : AWS garantit la disponibilité et l'évolutivité du cluster Amazon EKS, car il gère le plan de contrôle à votre place et le rend disponible dans toutes les zones de disponibilité AWS.
- Gestion des nœuds : Au lieu d'ajouter manuellement des nœuds, vous pouvez demander à Amazon EKS de créer automatiquement des nœuds selon vos besoins, à l'aide de groupes de nœuds gérés (voir the section called "Groupes de nœuds gérés") ou de Karpenter. Les groupes de nœuds gérés sont intégrés à Kubernetes Cluster Autoscaling. Grâce aux outils de gestion des nœuds, vous pouvez réaliser des économies, notamment grâce aux instances Spot et à la consolidation des nœuds, et bénéficier d'une disponibilité optimale, en utilisant les fonctionnalités Planification pour définir le déploiement des charges de travail et la sélection des nœuds.
- Réseau de cluster : À l'aide de modèles CloudFormation, `eksctl` configure le réseau entre les composants du plan de contrôle et du plan de données (nœud) dans le cluster Kubernetes. Il met également en place des points d'accès permettant les communications internes et externes. Pour plus d'informations, consultez la section Démystifier la mise en réseau des clusters pour les composants master Amazon EKS. Les communications entre les pods dans Amazon EKS s'effectuent à l'aide des identités de pod Amazon EKS (voir the section called "Pod Identity"), qui permettent aux pods d'exploiter les méthodes AWS cloud de gestion des informations d'identification et des autorisations.
- Modules complémentaires : Amazon EKS vous évite d'avoir à créer et à ajouter des composants logiciels couramment utilisés pour prendre en charge les clusters Kubernetes. Par exemple, lorsque vous créez un cluster Amazon EKS à partir du AWS Management Console, il ajoute

automatiquement les modules complémentaires Amazon EKS kube-proxy (the section called "kube-proxy"), Amazon VPC CNI plug-in for Kubernetes (the section called "CNI Amazon VPC") et CoreDNS (the section called "CoreDNS"). Consultez the section called "Modules complémentaires Amazon EKS" pour en savoir plus sur ces modules complémentaires, y compris la liste des modules disponibles.

Pour exécuter vos clusters sur vos propres ordinateurs et réseaux locaux, Amazon propose Amazon EKS Anywhere. Au lieu d'utiliser le cloud AWS comme fournisseur, vous avez la possibilité d'exécuter Amazon EKS Anywhere sur VMWare vSphere, matériel nu (fournisseur Tinkerbell), Snow, CloudStack ou Nutanix en utilisant votre propre équipement.

Amazon EKS Anywhere est basé sur le même logiciel Amazon EKS Distro que celui utilisé par Amazon EKS. Cependant, Amazon EKS Anywhere s'appuie sur différentes implémentations de l'interface Kubernetes Cluster API (CAPI) pour gérer le cycle de vie complet des machines dans un cluster Amazon EKS Anywhere (telles que CAPV pour vSphere et CAPC pour CloudStack). Étant donné que l'ensemble du cluster fonctionne sur votre équipement, vous assumez la responsabilité supplémentaire de gérer le plan de contrôle et de sauvegarder ses données (voir `etcd` plus loin dans ce document).

Composants d'un cluster

Les composants du cluster Kubernetes sont divisés en deux grandes catégories : le plan de contrôle et les composants master. Les composants du plan de contrôle gèrent le cluster et fournissent un accès à ses API. Les composants master (parfois simplement appelés Nœuds) fournissent les emplacements où les charges de travail réelles sont exécutées. Les composants de nœud sont constitués de services qui s'exécutent sur chaque nœud pour communiquer avec le plan de contrôle et exécuter des conteneurs. L'ensemble des composants master de votre cluster est appelé Plan de données.

Plan de contrôle

Le plan de contrôle est constitué d'un ensemble de services qui gèrent le cluster. Ces services peuvent tous fonctionner sur un seul ordinateur ou être répartis sur plusieurs ordinateurs. En interne, elles sont appelées instances du plan de contrôle (CPI). Le fonctionnement des CPI dépend de la taille du cluster et des exigences en matière de haute disponibilité. À mesure que la demande augmente dans le cluster, un service de plan de contrôle peut se mettre à l'échelle pour fournir davantage d'instances de ce service, les requêtes étant réparties entre les instances.

Les tâches effectuées par les composants du plan de contrôle Kubernetes sont les suivantes :

- Communication avec les composants du cluster (serveur API) : Le serveur API (kube-apiserver) expose l'API Kubernetes afin que les requêtes puissent être envoyées au cluster depuis l'intérieur et l'extérieur de celui-ci. En d'autres termes, les demandes d'ajout ou de modification des objets d'un cluster (Pods, Services, Nœuds, etc.) peuvent provenir de commandes externes, telles que les demandes d'exécution `kubectl` pour exécuter un Pod. De même, des requêtes peuvent être envoyées depuis le serveur API vers des composants au sein du cluster, comme une requête au service `kubelet` pour connaître l'état d'un pod.
- Stocker les données relatives au cluster (stockage de valeurs clés **`etcd`**) : Le service `etcd` joue un rôle essentiel dans le suivi de l'état actuel du cluster. Si le service `etcd` devenait inaccessible, vous ne pourriez plus mettre à jour ou consulter l'état du cluster, mais les charges de travail continueraient à s'exécuter pendant un certain temps. Pour cette raison, les clusters critiques disposent généralement de plusieurs instances du service `etcd` fonctionnant simultanément avec équilibrage de charge et effectuent des sauvegardes périodiques du magasin de valeurs clés `etcd` en cas de perte ou de corruption des données. N'oubliez pas que, dans Amazon EKS, tout cela est géré automatiquement par défaut. Amazon EKS Anywhere fournit des instructions pour la sauvegarde et la restauration d'etcd. Consultez le modèle de données etcd pour savoir comment `etcd` gère les données.
- Planifier des pods vers des nœuds (planificateur): Les demandes de démarrage ou d'arrêt d'un pod dans Kubernetes sont dirigées vers le planificateur Kubernetes (kube-scheduler). Étant donné qu'un cluster peut comporter plusieurs nœuds capables d'exécuter le pod, c'est au planificateur qu'il revient de choisir le ou les nœuds (dans le cas de réplicas) sur lesquels le pod doit s'exécuter. Si la capacité disponible est insuffisante pour exécuter le pod demandé sur un nœud existant, la demande échouera, sauf si vous avez pris d'autres dispositions. Ces dispositions pourraient inclure des services d'activation tels que Managed Node Groups (the section called "Groupes de nœuds gérés") ou Karpenter, capables de démarrer automatiquement de nouveaux nœuds pour gérer les charges de travail.
- Maintenir les composants dans l'état souhaité (Controller Manager) : Le Kubernetes Controller Manager s'exécute en tant que processus démon (kube-controller-manager) afin de surveiller l'état du cluster et d'apporter des modifications au cluster pour rétablir les états attendus. En particulier, plusieurs contrôleurs surveillent différents objets Kubernetes, notamment `statefulset-controller`, `endpoint-controller`, `cronjob-controller`, `node-controller`, et d'autres.
- Gérer les ressources cloud (Cloud Controller Manager) : Les interactions entre Kubernetes et le fournisseur de cloud qui traite les demandes relatives aux ressources du centre de données sous-jacent sont gérées par le Cloud Controller Manager (cloud-controller-manager). Les contrôleurs gérés par Cloud Controller Manager peuvent inclure un contrôleur de routage (pour configurer les

routes du réseau cloud), un contrôleur de service (pour utiliser les services d'équilibrage de charge cloud) et un contrôleur de cycle de vie des nœuds (pour synchroniser les nœuds avec Kubernetes tout au long de leur cycle de vie).

Composant master (plan de données)

Pour un cluster Kubernetes à nœud unique, les charges de travail s'exécutent sur la même machine que le plan de contrôle. Cependant, une configuration plus standard consiste à disposer d'un ou plusieurs systèmes informatiques distincts (nœuds) dédiés à l'exécution des charges de travail Kubernetes.

Lorsque vous créez un cluster Kubernetes pour la première fois, certains outils de création de cluster vous permettent de configurer un certain nombre de nœuds à ajouter au cluster (soit en identifiant les systèmes informatiques existants, soit en demandant au fournisseur d'en créer de nouveaux). Avant d'ajouter des charges de travail à ces systèmes, des services sont ajoutés à chaque nœud afin de mettre en œuvre les fonctionnalités suivantes :

- Gérer chaque nœud (**`kubelet`**) : Le serveur API communique avec le service kubelet exécuté sur chaque nœud afin de s'assurer que le nœud est correctement enregistré et que les pods demandés par le planificateur sont en cours d'exécution. Le kubelet peut lire les manifestes des pods et configurer les volumes de stockage ou d'autres fonctionnalités nécessaires aux pods sur le système local. Il peut également vérifier l'état des conteneurs gérés localement.
- Exécuter des conteneurs sur un nœud (environnement d'exécution de conteneurs) : l'exécution de conteneurs sur chaque nœud gère les conteneurs demandés pour chaque pod attribué au nœud. Cela signifie qu'il peut extraire des images de conteneur du registre approprié, exécuter le conteneur, l'arrêter et répondre aux requêtes concernant le conteneur. L'exécution de conteneur par défaut est containerd. À partir de Kubernetes 1.24, l'intégration spéciale de Docker (`dockershim`) qui pouvait être utilisée comme exécution de conteneur a été supprimée de Kubernetes. Bien que vous puissiez toujours utiliser Docker pour tester et exécuter des conteneurs sur votre système local, pour utiliser Docker avec Kubernetes, vous devez désormais installer Docker Engine sur chaque nœud afin de pouvoir l'utiliser avec Kubernetes.
- Gérer la mise en réseau entre les conteneurs (**`kube-proxy`**) : Afin de prendre en charge la communication entre les pods, Kubernetes utilise une fonctionnalité appelée Service pour configurer des réseaux de pods qui suivent les adresses IP et les ports associés à ces pods. Le service kube-proxy s'exécute sur chaque nœud pour permettre la communication entre les pods.

Extension des clusters

Il existe certains services que vous pouvez ajouter à Kubernetes pour prendre en charge le cluster, mais qui ne sont pas exécutés dans le plan de contrôle. Ces services s'exécutent souvent directement sur des nœuds dans l'espace de noms kube-system ou dans leur propre espace de noms (comme c'est souvent le cas avec les fournisseurs de services tiers). Un exemple courant est le service CoreDNS, qui fournit des services DNS au cluster. Reportez-vous à la section Découverte des services intégrés pour savoir comment voir quels services de cluster sont en cours d'exécution dans kube-system sur votre cluster.

Il existe différents types de modules complémentaires que vous pouvez envisager d'ajouter à vos clusters. Pour maintenir vos clusters en bon état, vous pouvez ajouter des fonctionnalités d'observabilité (voir Grappes de moniteurs) qui vous permettent d'effectuer des tâches telles que la journalisation, l'audit et la mesure des métriques. Grâce à ces informations, vous pouvez résoudre les problèmes qui surviennent, souvent par le biais des mêmes interfaces d'observabilité. Parmi les exemples de ce type de services, on peut citer Amazon GuardDuty, CloudWatch (voir the section called "Amazon CloudWatch"), Distro for OpenTelemetry AWS, Amazon VPC CNI plug-in for Kubernetes (voir the section called "CNI Amazon VPC") et Grafana Kubernetes Monitoring. Pour le stockage (voir Stockage de données d'application), les modules complémentaires à Amazon EKS comprennent Amazon Elastic Block Store CSI Driver (voir the section called "Amazon EBS"), Amazon Elastic File System CSI Driver (voir the section called "Amazon EFS") et plusieurs modules complémentaires de stockage tiers tels que le pilote CSI Amazon FSx pour NetApp ONTAP the section called "Amazon FSx pour NetApp ONTAP").

Pour une liste plus complète des modules complémentaires Amazon EKS disponibles, consultez the section called "Modules complémentaires Amazon EKS".

Charges de travail

Kubernetes définit une charge de travail comme « une application s'exécutant sur Kubernetes ». Cette application peut être constituée d'un ensemble de microservices exécutés sous forme de conteneurs dans des pods, ou peut être exécutée sous forme de tâche batch ou d'autres types d'applications. Le travail de Kubernetes est de s'assurer que les demandes que vous faites pour configurer ou déployer ces objets sont exécutées. En tant que personne chargée du déploiement d'applications, vous devez vous familiariser avec la manière dont les conteneurs sont construits, comment les pods sont définis et quelles méthodes vous pouvez utiliser pour les déployer.

Conteneurs

L'élément le plus basique d'une charge de travail d'application que vous déployez et gérez dans Kubernetes est un pod . Un Pod représente un moyen de contenir les composants d'une application et de définir des spécifications décrivant les attributs du Pod. Comparez cela à un paquet RPM ou Deb, qui regroupe des logiciels pour un système Linux, mais qui ne fonctionne pas en tant qu'entité à part entière.

Le Pod étant la plus petite unité déployable, il contient généralement un seul conteneur. Cependant, plusieurs conteneurs peuvent se trouver dans un pod dans les cas où les conteneurs sont étroitement couplés. Par exemple, un conteneur de serveur Web peut être intégré dans un pod avec un conteneur de type sidecar qui peut fournir des services de journalisation, de surveillance ou d'autres services étroitement liés au conteneur de serveur Web. Dans ce cas, le fait d'être dans le même pod garantit que, pour chaque instance en cours d'exécution du pod, les deux conteneurs s'exécutent toujours sur le même nœud. De même, tous les conteneurs d'un pod partagent le même environnement, les conteneurs d'un pod s'exécutant comme s'ils se trouvaient sur le même hôte isolé. Cela a pour effet que les conteneurs partagent une seule adresse IP qui donne accès au pod et que les conteneurs peuvent communiquer entre eux comme s'ils fonctionnaient sur leur propre hôte local.

Les spécifications du pod (PodSpec) définissent l'état souhaité du pod. Vous pouvez déployer un ou plusieurs pods individuels à l'aide des ressources de charge de travail pour gérer les modèles de pod. Les ressources de charge de travail comprennent Déploiements (pour gérer plusieurs réplicas de pods), les StatefulSets (pour déployer des pods qui doivent être uniques, tels que les pods de base de données) et les DaemonSets (lorsqu'un pod doit s'exécuter en continu sur chaque nœud). Plus d'informations à ce sujet plus tard.

Alors qu'un Pod est la plus petite unité que vous déployez, un conteneur est la plus petite unité que vous construisez et gérez.

Containers de construction

Le Pod n'est en réalité qu'une structure autour d'un ou plusieurs conteneurs, chaque conteneur contenant lui-même le système de fichiers, les exécutables, les fichiers de configuration, les bibliothèques et autres composants nécessaires au fonctionnement de l'application. Comme c'est une société appelée Docker Inc. qui a popularisé les conteneurs, certaines personnes les appellent « conteneurs Docker ». Cependant, la Open Container Initiative a depuis défini les exécutions, les images et les méthodes de distribution des conteneurs pour l'industrie. Ajoutez à cela le fait que

les conteneurs ont été créés à partir de nombreuses fonctionnalités Linux existantes, d'autres les appellent souvent conteneurs OCI, conteneurs Linux ou simplement conteneurs.

Lorsque vous créez un conteneur, vous commencez généralement par un fichier Dockerfile (qui porte littéralement ce nom). Dans ce Dockerfile, vous identifiez :

- Une image de base : Une image de conteneur de base est un conteneur généralement construit à partir d'une version minimale du système de fichiers d'un système d'exploitation (tel que Red Hat Enterprise Linux ou Ubuntu) ou d'un système minimal amélioré pour fournir des logiciels permettant d'exécuter des types d'applications spécifiques (telles que les applications nodejs ou python).
- Logiciel d'application : Vous pouvez ajouter votre logiciel d'application à votre conteneur de la même manière que vous l'ajouteriez à un système Linux. Par exemple, dans votre fichier Dockerfile, vous pouvez exécuter `npm` et `yarn` pour installer une application Java ou `yum` et `dnf` pour installer des paquets RPM. En d'autres termes, en utilisant une commande RUN dans un fichier Dockerfile, vous pouvez exécuter n'importe quelle commande disponible dans le système de fichiers de votre image de base pour installer ou configurer des logiciels à l'intérieur de l'image de conteneur résultante.
- Instructions : La référence Dockerfile décrit les instructions que vous pouvez ajouter à un Dockerfile lorsque vous le configurez. Il s'agit notamment des instructions utilisées pour créer le contenu du conteneur lui-même (ou les fichiers `ADD` ou `COPY` du système local), identifier les commandes à exécuter lorsque le conteneur est lancé (`CMD` ou `ENTRYPOINT`), et connecter le conteneur au système sur lequel il s'exécute (en identifiant le `USER` sous lequel il doit s'exécuter, un local `VOLUME` à monter ou les ports à `EXPOSE`).

Alors que la commande `docker` et le service ont traditionnellement été utilisés pour créer des conteneurs (`docker build`), d'autres outils sont disponibles pour créer des images de conteneurs, notamment podman et nerdctl. Consultez les sections Créer de meilleures images de conteneur ou Présentation de Docker Build pour en savoir plus sur la création de conteneurs.

Conteneurs de stockage

Une fois votre image de conteneur créée, vous pouvez la stocker dans un registre de distribution de conteneurs sur votre poste de travail ou dans un registre de conteneurs public. L'exécution d'un registre de conteneurs privé sur votre poste de travail vous permet de stocker des images de conteneurs localement, ce qui les rend facilement accessibles.

Pour stocker les images de conteneur de manière plus publique, vous pouvez les envoyer vers un registre de conteneurs public. Les registres publics de conteneurs fournissent un emplacement

centralisé pour le stockage et la distribution des images de conteneurs. Parmi les exemples de registres de conteneurs publics, on peut citer le registre Amazon Elastic Container Registry, Red Hat Quay et Docker Hub.

Lorsque vous exécutez des charges de travail conteneurisées sur Amazon Elastic Kubernetes Service (Amazon EKS), nous vous recommandons de récupérer des copies des images officielles Docker stockées dans Amazon Elastic Container Registry. Amazon ECR stocke ces images depuis 2021. Vous pouvez rechercher des images de conteneurs populaires dans la galerie publique Amazon ECR, et plus particulièrement pour les images Docker Hub, vous pouvez effectuer une recherche dans la galerie Amazon ECR Docker.

Containers de course

Les conteneurs étant construits dans un format standard, ils peuvent fonctionner sur n'importe quelle machine capable d'exécuter une exécution de conteneur (tel que Docker) et dont le contenu correspond à l'architecture de la machine locale (telle que `x86_64` ou `arm`). Pour tester un conteneur ou simplement l'exécuter sur votre bureau local, vous pouvez utiliser les commandes `docker run` ou `podman run` pour démarrer un conteneur sur l'hôte local. Pour Kubernetes, cependant, chaque composant master dispose d'une exécution de conteneur déployée et c'est à Kubernetes qu'il appartient de demander à un nœud d'exécuter un conteneur.

Une fois qu'un conteneur a été affecté à un nœud, celui-ci vérifie si la version demandée de l'image du conteneur existe déjà sur le nœud. Si ce n'est pas le cas, Kubernetes demande à l'exécution de conteneur de récupérer ce conteneur à partir du registre de conteneurs approprié, puis d'exécuter ce conteneur localement. Gardez à l'esprit qu'une image de conteneur fait référence au progiciel qui est transféré entre votre ordinateur portable, le registre de conteneurs et les nœuds Kubernetes. Un conteneur désigne une instance en cours d'exécution de cette image.

Pods

Une fois vos conteneurs prêts, travailler avec les pods implique de les configurer, de les déployer et de les rendre accessibles.

Configuration des pods

Lorsque vous définissez un Pod, vous lui attribuez un ensemble d'attributs. Ces attributs doivent inclure au moins le nom du pod et l'image du conteneur à exécuter. Cependant, il existe de nombreux autres éléments que vous souhaitez également configurer avec vos définitions de pod (consultez la page PodSpec pour plus de détails sur ce qui peut être inclus dans un pod). Il s'agit des licences suivantes :

- Stockage : lorsqu'un conteneur en cours d'exécution est arrêté et supprimé, les stockages de données dans ce conteneur disparaissent, sauf si vous configurez un stockage plus permanent. Kubernetes prend en charge de nombreux types de stockage différents et les regroupe sous l'appellation Volumes. Les types de stockage incluent CephFS, NFS,iSCSI et autres. Vous pouvez même utiliser un périphérique de stockage en mode bloc à partir de l'ordinateur local. Avec l'un de ces types de stockage disponibles dans votre cluster, vous pouvez monter le volume de stockage sur un point de montage sélectionné dans le système de fichiers de votre conteneur. Un volume persistant est un volume qui continue d'exister après la suppression du pod, tandis qu'un volume éphémère est supprimé lorsque le pod est supprimé. Si votre administrateur de cluster a créé différentes classes de stockage pour votre cluster, vous pouvez choisir les attributs du stockage que vous utilisez, par exemple si le volume est supprimé ou récupéré après utilisation, s'il s'étend si davantage d'espace est nécessaire, et même s'il répond à certaines exigences de performances.
- Secrets : En rendant les secrets disponibles pour les conteneurs dans les spécifications Pod, vous pouvez fournir les autorisations dont ces conteneurs ont besoin pour accéder aux systèmes de fichiers, bases de données ou autres ressources protégées. Les clés, les mots de passe et les jetons font partie des éléments qui peuvent être stockés en tant que secrets. L'utilisation de secrets vous évite d'avoir à stocker ces informations dans des images de conteneur, mais vous devez simplement mettre les secrets à la disposition des conteneurs en cours d'exécution. ConfigMaps est similaire à Secrets. Une `ConfigMap` tendance à contenir moins d'informations critiques, telles que les paires clé-valeur pour la configuration d'un service.
- Ressources de conteneur : Les objets permettant de configurer davantage les conteneurs peuvent prendre la forme d'une configuration de ressources. Pour chaque conteneur, vous pouvez demander la quantité de mémoire et de CPU qu'il peut utiliser, ainsi que fixer des limites quant à la quantité totale de ces ressources que le conteneur peut utiliser. Voir Gestion des ressources pour les pods et les conteneurs pour des exemples.
- Interruptions : Les pods peuvent être perturbés involontairement (un nœud tombe en panne) ou volontairement (une mise à niveau est souhaitée). En configurant un budget de perturbation des pods, vous pouvez exercer un certain contrôle sur la disponibilité de votre application en cas de perturbation. Consultez la section Spécification d'un budget de perturbation pour votre application pour obtenir des exemples.
- Espaces de noms : Kubernetes propose différentes méthodes pour isoler les composants et les charges de travail Kubernetes les uns des autres. L'exécution de tous les pods d'une application particulière dans le même espace de noms est une méthode courante pour sécuriser et gérer ces pods ensemble. Vous pouvez créer vos propres espaces de noms à utiliser ou choisir de ne pas indiquer d'espace de noms (ce qui oblige Kubernetes à utiliser l'espace de noms `default`). Les

composants du plan de contrôle Kubernetes s'exécutent généralement dans l'espace de noms kube-system.

La configuration qui vient d'être décrite est généralement regroupée dans un fichier YAML à appliquer au cluster Kubernetes. Pour les clusters Kubernetes personnels, vous pouvez simplement stocker ces fichiers YAML sur votre système local. Cependant, avec des clusters et des charges de travail plus critiques, GitOps est un moyen populaire d'automatiser le stockage et les mises à jour des ressources de charge de travail et de l'infrastructure Kubernetes.

Les objets utilisés pour rassembler et déployer les informations Pod sont définis par l'une des méthodes de déploiement suivantes.

Déploiement de pods

La méthode que vous choisirez pour déployer des pods dépend du type d'application que vous prévoyez d'exécuter avec ces pods. Voici certains de vos choix :

- Applications sans état : Une application sans état n'enregistre pas les données de session d'un client, de sorte qu'une autre session n'a pas besoin de se référer à ce qui s'est passé lors d'une session précédente. Cela permet de remplacer plus facilement les pods par de nouveaux s'ils deviennent insalubres ou de les déplacer sans enregistrer leur état. Si vous exécutez une application sans état (telle qu'un serveur Web), vous pouvez utiliser un Déploiement pour déployer des Pods et des ReplicaSets. Un ReplicaSet définit le nombre d'instances d'un Pod que vous souhaitez exécuter simultanément. Bien que vous puissiez exécuter un ReplicaSet directement, il est courant d'exécuter des réplicas directement dans un déploiement, afin de définir le nombre de réplicas d'un pod qui doivent être exécutés simultanément.
- Applications dynamiques : Une application avec état est une application dans laquelle l'identité du pod et l'ordre dans lequel les pods sont lancés sont importants. Ces applications ont besoin d'un stockage persistant stable qui doit être déployé et mis à l'échelle de manière cohérente. Pour déployer une application dynamique dans Kubernetes, vous pouvez utiliser StatefulSets. Une base de données est un exemple d'application généralement exécutée en tant que StatefulSet. Dans un StatefulSet, vous pouvez définir des réplicas, le pod et ses conteneurs, les volumes de stockage à monter et les emplacements dans le conteneur où les données sont stockées. Consultez Exécuter une application avec état répliquée pour obtenir un exemple de base de données déployée en tant que ReplicaSet.
- Applications par nœud : Il arrive parfois que vous souhaitiez exécuter une application sur chaque nœud de votre cluster Kubernetes. Par exemple, votre centre de données peut exiger que chaque

ordinateur exécute une application de surveillance ou un service d'accès à distance particulier. Pour Kubernetes, vous pouvez utiliser un DaemonSet pour vous assurer que l'application sélectionnée s'exécute sur tous les nœuds de votre cluster.

- Les applications s'exécutent jusqu'à leur achèvement : Certaines applications doivent être exécutées pour accomplir une tâche particulière. Cela peut inclure un système qui publie des rapports d'état mensuels ou nettoie les anciennes données. Un objet Job peut être utilisé pour configurer une application afin qu'elle démarre et s'exécute, puis se ferme une fois la tâche terminée. Un objet CronJob vous permet de configurer une application pour qu'elle s'exécute à une heure, une minute, un jour du mois, un mois ou un jour de la semaine spécifiques, à l'aide d'une structure définie par le format crontab de Linux.

Rendre les applications accessibles depuis le réseau

Les applications étant souvent déployées sous forme d'un ensemble de microservices déplacés à différents endroits, Kubernetes avait besoin d'un moyen permettant à ces microservices de se trouver les uns les autres. De plus, pour permettre à d'autres personnes d'accéder à une application en dehors du cluster Kubernetes, Kubernetes avait besoin d'un moyen d'exposer cette application sur des adresses et des ports externes. Ces fonctionnalités liées au réseau sont réalisées respectivement à l'aide d'objets Service et Ingress :

- Services :Étant donné qu'un pod peut se déplacer vers différents nœuds et adresses, un autre pod qui doit communiquer avec le premier pod peut avoir des difficultés à le localiser. Pour résoudre ce problème, Kubernetes vous permet de représenter une application en tant que Service. Avec un service, vous pouvez identifier un pod ou un ensemble de pods avec un nom particulier, puis indiquer quel port expose le service de cette application à partir du pod et quels ports une autre application pourrait utiliser pour contacter ce service. Un autre pod au sein d'un cluster peut simplement demander un service par son nom, et Kubernetes dirigera cette demande vers le port approprié pour une instance du pod exécutant ce service.
- Ingress : Ingress permet de mettre les applications représentées par les services Kubernetes à la disposition des clients situés en dehors du cluster. Les fonctionnalités de base d'Ingress comprennent un équilibreur de charge (géré par Ingress), le contrôleur Ingress et des règles pour acheminer les requêtes du contrôleur vers le service. Il existe plusieurs contrôleurs Ingress parmi lesquels vous pouvez choisir avec Kubernetes.

Étapes suivantes

Comprendre les concepts de base de Kubernetes et leur relation avec Amazon EKS vous aidera à naviguer dans la documentation Amazon EKS et Kubernetes afin de trouver les informations dont vous avez besoin pour gérer les clusters Amazon EKS et déployer des charges de travail sur ces clusters. Pour commencer à utiliser Amazon EKS, choisissez l'une des options suivantes :

- the section called "Créer un cluster (eksctl)"
- the section called "Créer un cluster"
- the section called "Exemple de déploiement (Linux)"
- Gestion du cluster

Déploiement de clusters Amazon EKS dans des environnements cloud et sur site

Comprendre les options de déploiement Amazon EKS

Amazon Elastic Kubernetes Service (Amazon EKS) est un service Kubernetes entièrement géré qui vous permet d'exécuter Kubernetes de manière transparente dans le cloud et dans vos environnements sur site.

Dans le cloud, Amazon EKS automatise la gestion de l'infrastructure des clusters Kubernetes pour le plan de contrôle et les nœuds Kubernetes. Ceci est essentiel pour planifier les conteneurs, gérer la disponibilité des applications, dimensionner dynamiquement les ressources, optimiser le calcul, stocker les données du cluster et exécuter d'autres fonctions critiques. Avec Amazon EKS, vous bénéficiez de performances robustes, d'une capacité de mise à l'échelle, d'une fiabilité et d'une disponibilité optimales de l'infrastructure AWS, ainsi que d'intégrations natives avec des services de mise en réseau AWS, de sécurité, de stockage et d'observabilité.

Pour simplifier l'exécution de Kubernetes dans vos environnements sur site, vous pouvez utiliser les mêmes clusters, fonctionnalités et outils poru the section called "Nœuds" ou les nœuds hybrides Amazon EKS sur votre propre infrastructure, ou vous pouvez utiliser Amazon EKS Anywhere pour les environnements autonomes isolés.

Amazon EKS dans le cloud

Vous pouvez utiliser Amazon EKS avec des ressources de calcul dans les régions AWS, les zones locales AWS et les zones Wavelength AWS. Avec Amazon EKS dans le cloud, la sécurité, la capacité de mise à l'échelle et la disponibilité du plan de contrôle Kubernetes sont entièrement gérées par AWS dans la région AWS. Lorsque vous exécutez des applications avec des ressources de calcul dans les régions AWS, vous bénéficiez de toutes les fonctionnalités d'Amazon EKS et d'AWS, y compris le mode automatique Amazon EKS, qui automatise entièrement la gestion de l'infrastructure du cluster Kubernetes pour le calcul, le stockage et la mise en réseau sur AWS en un seul clic. Lorsque vous exécutez des applications avec des ressources de calcul dans les zones locales AWS et les zones Wavelength AWS, vous pouvez utiliser les nœuds autogérés Amazon EKS pour connecter des instances Amazon EC2 pour le calcul de votre cluster et utiliser les autres services AWS disponibles dans les zones locales AWS et les zones Wavelength AWS. Pour plus d'informations, consultez les fonctionnalités des zones locales AWS et les fonctionnalités des zones Wavelength AWS.

	Amazon EKS dans les régions AWS	Amazon EKS dans les zones locales/Wavelength
Gestion du plan de contrôle Kuberenetes	Géré par AWS	Géré par AWS
Emplacement du plan de contrôle Kubernetes	Régions AWS	Régions AWS
Plan de données Kubernetes	• Amazon EKS Auto Mode • Groupes de nœuds gérés par Amazon EKS • Nœuds autogérés Amazon EC2 • AWS Fargate	• Groupes de nœuds gérés Amazon EKS (zones locales uniquement) • Nœuds autogérés Amazon EC2
Emplacement du plan de données Kubernetes	Régions AWS	Zones locales AWS ou Wavelength

Amazon EKS dans votre centre de données ou vos environnements périphériques

Si vous devez exécuter des applications dans vos propres centres de données ou environnements périphériques, vous pouvez utiliser Amazon EKS sur AWS Outposts ou les nœuds hybrides Amazon EKS. Vous pouvez utiliser des nœuds autogérés avec des instances Amazon EC2 sur AWS Outposts pour le calcul de votre cluster, ou vous pouvez utiliser des nœuds hybrides Amazon EKS avec votre propre infrastructure sur site ou en périphérie pour le calcul de votre cluster. AWS Outposts est une infrastructure gérée par AWS que vous exécutez dans vos centres de données ou vos installations de colocation, tandis que les nœuds hybrides Amazon EKS s'exécutent sur vos machines physiques ou virtuelles que vous gérez dans vos environnements sur site ou périphériques. Amazon EKS sur AWS Outposts et les nœuds hybrides Amazon EKS nécessitent une connexion fiable entre vos environnements sur site et une région AWS. Vous pouvez utiliser les mêmes clusters, fonctionnalités et outils Amazon EKS que ceux que vous utilisez pour exécuter des applications dans le cloud. Lorsque vous utilisez AWS Outposts, vous pouvez également déployer l'ensemble du cluster Kubernetes sur AWS Outposts avec des clusters locaux Amazon EKS sur AWS Outposts.

	Nœuds hybrides Amazon EKS	Amazon EKS sur AWS Outposts
Gestion du plan de contrôle Kuberenetes	Géré par AWS	Géré par AWS
Emplacement du plan de contrôle Kubernetes	Régions AWS	Régions AWS ou AWS Outposts
Plan de données Kubernetes	Machines physiques ou virtuelles gérées par le client	Nœuds autogérés Amazon EC2
Emplacement du plan de données Kubernetes	Centre de données client ou environnement périphérique	Centre de données client ou environnement périphérique

Amazon EKS Anywhere pour les environnements isolés

Amazon EKS Anywhere simplifie la gestion des clusters Kubernetes grâce à l'automatisation des tâches lourdes et indifférenciées telles que la configuration de l'infrastructure et les opérations du cycle de vie des clusters Kubernetes dans les environnements sur site et en périphérie.

Contrairement à Amazon EKS, Amazon EKS Anywhere est un produit géré par le client. Les clients sont responsables des opérations liées au cycle de vie des clusters et de la maintenance des clusters Amazon EKS Anywhere. Amazon EKS Anywhere s'appuie sur le sous-projet Kubernetes Cluster API (CAPI) et prend en charge toute une gamme d'infrastructures, notamment VMware vSphere, matériel nu, Nutanix, Apache CloudStack et AWS Snow. Amazon EKS Anywhere peut être exécuté dans des environnements isolés et offre des intégrations optionnelles avec des services régionaux AWS pour la gestion de l'observabilité et des identités. Pour bénéficier d'une assistance pour Amazon EKS Anywhere et accéder aux modules complémentaires Kubernetes fournis par AWS, vous pouvez acheter des abonnements Amazon EKS Anywhere Enterprise.

	Amazon EKS Anywhere
Gestion du plan de contrôle Kuberenetes	Gérée par le client
Emplacement du plan de contrôle Kubernetes	Centre de données client ou environnement périphérique
Plan de données Kubernetes	Machines physiques ou virtuelles gérées par le client
Emplacement du plan de données Kubernetes	Centre de données client ou environnement périphérique

Amazon EKS Tooling

Vous pouvez utiliser Amazon EKS Connector pour enregistrer et connecter n'importe quel cluster Kubernetes conforme sur AWS, puis l'afficher dans la console Amazon EKS. Une fois qu'un cluster est connecté, vous pouvez voir le statut, la configuration et les charges de travail de ce cluster dans la console Amazon EKS. Vous pouvez utiliser cette fonctionnalité pour afficher les clusters connectés dans la console Amazon EKS, mais Amazon EKS Connector ne permet pas de gérer ni de modifier vos clusters connectés via la console Amazon EKS.

Amazon EKS Distro est la distribution AWS des composants Kubernetes sous-jacents qui alimentent toutes les offres Amazon EKS. Il comprend les composants essentiels nécessaires au fonctionnement d'un cluster Kubernetes, tels que les composants du plan de contrôle Kubernetes (etcd, kube-apiserver, kube-scheduler, kube-controller-manager) et les composants réseau (CoreDNS, kube-proxy, plugins CNI). Amazon EKS Distro peut être utilisé pour gérer vous-même des

clusters Kubernetes avec les outils de votre choix. Les déploiements Amazon EKS Distro ne sont pas couverts par les plans de support AWS.

Configuration pour utiliser Amazon EKS

Afin de préparer la gestion en ligne de commande de vos clusters Amazon EKS, il est nécessaire d'installer plusieurs outils. Utilisez les éléments suivants pour configurer les informations d'identification, créer et modifier des clusters, et travailler avec les clusters une fois qu'ils sont en cours d'exécution :

- Configurer l'AWS CLI : procurez-vous AWS CLI pour configurer et gérer les services dont vous avez besoin pour travailler avec les clusters Amazon EKS. Vous avez notamment besoin de l'interface AWS CLI pour configurer les informations d'identification, mais également pour d'autres services AWS.
- Configurer kubectl et eksctl : l'interface CLI `eksctl` interagit avec AWS pour créer, modifier et supprimer des clusters Amazon EKS. Une fois le cluster opérationnel, utilisez la commande open source `kubectl` pour gérer les objets Kubernetes au sein de vos clusters Amazon EKS.
- Configurez un environnement de développement (facultatif) : envisagez d'ajouter les outils suivants :
  - Outil de déploiement local : si vous débutez avec Kubernetes, envisagez d'installer un outil de déploiement local tel que minikube ou kind. Ces outils vous permettent de disposer d'un cluster Amazon EKS sur votre machine locale pour tester des applications.
  - Gestionnaire de paquets : helm est un gestionnaire de paquets populaire pour Kubernetes qui simplifie l'installation et la gestion de paquets complexes. Avec Helm, il est plus facile d'installer et de gérer des paquets tels que AWS Load Balancer Controller sur votre cluster Amazon EKS.

Étapes suivantes

- Configurer AWS CLI
- Configurez kubectl et eksctl
- Démarrage rapide : déploiement d'une application Web et stockage des données

Configuration de l'interface de ligne de commande AWS

La CLI AWS est un outil en ligne de commande permettant de travailler avec des services AWS, notamment Amazon EKS. Il sert également à authentifier les utilisateurs ou les rôles IAM afin d'accéder au cluster Amazon EKS et à d'autres ressources AWS depuis votre ordinateur local. Pour

mettre en service des ressources dans AWS depuis la ligne de commande, vous devez obtenir un ID de clé d'accès AWS et une clé secrète à utiliser dans la ligne de commande. Vous devez ensuite configurer ces informations d'identification dans l'interface CLI AWS. Si vous n'avez pas encore installé l'AWS CLI, consultez la section Installer ou mettre à jour la dernière version de l'AWS CLI du Guide de l'utilisateur de l'interface de ligne de commande AWS.

Pour créer une clé d'accès

1. Connectez-vous au AWS Management Console.
2. Pour les comptes mono-utilisateur ou multi-utilisateurs :
   - Compte utilisateur unique –:: En haut à droite, sélectionnez votre nom d'utilisateur AWS pour ouvrir le menu de navigation. Par exemple, sélectionnez **`webadmin`** .
   - Compte multi-utilisateurs —:: Choisissez IAM dans la liste des services. Dans le tableau de bord IAM, sélectionnez Utilisateurs, puis choisissez le nom de l'utilisateur.
3. Choisissez Informations d'identification de sécurité.
4. Sous Clés d'accès, choisissez Créer une clé d'accès.
5. Choisissez Interface de ligne de commande (CLI), puis Suivant.
6. Choisissez Create access key (Créer une clé d'accès).
7. Choisissez Télécharger le fichier .csv.

Pour configurer l'interface de ligne de commande AWS

Après avoir installé l'interface AWS CLI, procédez comme suit pour la configurer. Pour plus d'informations, consultez la section Configurer l'AWS CLI du Guide de l'utilisateur de l'interface de la ligne de commande AWS.

1. Dans une fenêtre de terminal, entrez la commande suivante :

   ```
   aws configure
   ```

   Vous pouvez éventuellement configurer un profil nommé, tel que `--profile cluster-admin`. Si vous configurez un profil nommé dans l'interface CLI AWS, vous devez toujours passer ce drapeau dans les commandes suivantes.
2. Saisissez vos informations d'identification AWS. Par exemple :

```
Access Key ID [None]: AKIAIOSFODNN7EXAMPLE
Secret Access Key [None]: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
Default region name [None]: region-code
Default output format [None]: json
```

Pour obtenir un jeton de sécurité

Si nécessaire, exécutez la commande suivante pour obtenir un nouveau jeton de sécurité pour l'interface CLI AWS. Pour plus d'informations, consultez get-session-token dans la référence des commandes CLI AWS.

Par défaut, le jeton est valide pendant 15 minutes. Pour modifier le délai d'expiration de session par défaut, transmettez l'indicateur `--duration-seconds`. Par exemple :

```
aws sts get-session-token --duration-seconds 3600
```

Cette commande renvoie les informations d'identification de sécurité temporaires pour une session CLI AWS. Le résultat de la réponse devrait être le suivant :

```
{
    "Credentials": {
        "AccessKeyId": "ASIA5FTRU3LOEXAMPLE",
        "SecretAccessKey": "JnKgvwfqUD9mNsPoi9IbxAYEXAMPLE",
        "SessionToken": "VERYLONGSESSIONTOKENSTRING",
        "Expiration": "2023-02-17T03:14:24+00:00"
    }
}
```

Pour vérifier l'identité de l'utilisateur

Si nécessaire, exécutez la commande suivante pour vérifier les informations d'identification AWS de votre identité utilisateur IAM (telle que *`ClusterAdmin`*) pour la session de terminal.

```
aws sts get-caller-identity
```

Cette commande renvoie l'Amazon Resource Name (ARN) de l'entité IAM configurée pour l'AWS CLI. Vous devriez voir l'exemple de réponse suivant :

```
{
    "UserId": "AKIAIOSFODNN7EXAMPLE",
    "Account": "01234567890",
    "Arn": "arn:aws:iam::01234567890:user/ClusterAdmin"
}
```

Étapes suivantes

- Configurer kubectl et eksctl
- Démarrage rapide : déploiement d'une application Web et stockage des données

Configuration de **kubectl** et **eksctl**

> **Tip**
>
> Inscrivez-vous aux prochains ateliers Amazon EKS.

Une fois la AWS CLI installée, vous devez installer deux autres outils pour créer et gérer vos clusters Kubernetes :

- `kubectl` : L'outil en ligne de commande `kubectl` est le principal outil que vous utiliserez pour gérer les ressources au sein de votre cluster Kubernetes. Cette page décrit comment télécharger et configurer le binaire `kubectl` correspondant à la version de votre cluster Kubernetes. Voir Installer ou mettre à jour kubectl.
- `eksctl`: L'outil de ligne de `eksctl` commande est conçu pour créer des clusters EKS dans le AWS cloud ou sur site (avec EKS Anywhere), ainsi que pour modifier et supprimer ces clusters. Voir Installer eksctl.

Installer ou mettre à jour **kubectl**

Cette rubrique vous aide à télécharger et installer ou mettre à jour le fichier binaire `kubectl` sur votre appareil. Le fichier binaire est identique aux versions communautaires en amont. Le binaire n'est pas propre à Amazon EKS ou AWS. Suivez les étapes ci-dessous pour obtenir la version spécifique de `kubectl` dont vous avez besoin, bien que de nombreux constructeurs se contentent d'exécuter `brew install kubectl` pour l'installer.

> **Note**
>
> Vous devez utiliser une version `kubectl` qui se situe à une différence de version mineure près du plan de contrôle de votre cluster Amazon EKS. Par exemple, un client `kubectl` `1.32` doit utiliser des clusters `1.31`, `1.32` et `1.33` Kubernetes.

Étape 1 : vérifier si **`kubectl`** est installé

Déterminez si vous avez déjà `kubectl` installé sur votre appareil.

```
kubectl version --client
```

Si `kubectl` est installé dans le chemin d'accès de votre appareil, l'exemple de sortie comprend des informations similaires à celles qui suivent. Si vous souhaitez mettre à jour la version que vous avez actuellement installée avec une version plus récente, passez à l'étape suivante en vous assurant d'installer la nouvelle version au même emplacement que votre version actuelle.

```
Client Version: v1.31.X-eks-1234567
```

Si vous n'obtenez aucun résultat, cela signifie que vous n'avez pas installé `kubectl` ou qu'il n'est pas installé à un emplacement accessible depuis le chemin d'accès de votre appareil.

Étape 2 : installer ou mettre à niveau **`kubectl`**

Installer ou mettre à niveau `kubectl` sur l'un des systèmes d'exploitation suivants :

- the section called "macOS"
- the section called "Linux (amd64)"
- the section called "Linux (arm64)"
- the section called "Windows"

> **Note**
>
> Si les téléchargements vers votre AWS région sont lents à partir des AWS régions utilisées dans cette section, pensez CloudFront à configurer la mise en avant du contenu. Pour plus d'informations, consultez Commencer avec une CloudFront distribution de base.

macOS

Suivez les étapes ci-dessous pour installer `kubectl` sur macOS. Les étapes sont les suivantes :

- Choisissez et téléchargez le fichier binaire correspondant à la version Kubernetes souhaitée.
- Vérification facultative de la somme de contrôle du fichier binaire.
- Ajouter l'exécution aux autorisations du fichier binaire.
- Copiez le fichier binaire dans un dossier de votre PATH.
- Ajouter éventuellement le répertoire du binaire à votre PATH.

Procédure :

1. Téléchargez le fichier binaire correspondant à la version Kubernetes de votre cluster depuis Amazon S3.
   - Kubernetes `1.34`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.34.2/2025-11-13/bin/darwin/amd64/kubectl
     ```

   - Kubernetes `1.33`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.33.5/2025-11-13/bin/darwin/amd64/kubectl
     ```

   - Kubernetes `1.32`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.32.9/2025-11-13/bin/darwin/amd64/kubectl
     ```

   - Kubernetes `1.31`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.31.13/2025-11-13/bin/darwin/amd64/kubectl
     ```

   - Kubernetes `1.30`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.14/2025-11-13/bin/darwin/amd64/kubectl
     ```

- Kubernetes `1.29`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.15/2025-11-13/bin/darwin/amd64/kubectl
```

- Kubernetes `1.28`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.28.15/2025-11-13/bin/darwin/amd64/kubectl
```

- Kubernetes `1.27`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.16/2025-01-10/bin/darwin/amd64/kubectl
```

- Kubernetes `1.26`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.26.15/2024-12-12/bin/darwin/amd64/kubectl
```

2. (Facultatif) Vérifiez le fichier binaire téléchargé avec le total de contrôle `SHA-256` de votre fichier binaire.

   a. Téléchargez la somme de contrôle `SHA-256` pour la version Kubernetes de votre cluster.

   - Kubernetes `1.34`

   ```
   curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.34.2/2025-11-13/bin/darwin/amd64/kubectl.sha256
   ```

   - Kubernetes `1.33`

   ```
   curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.33.5/2025-11-13/bin/darwin/amd64/kubectl.sha256
   ```

   - Kubernetes `1.32`

   ```
   curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.32.9/2025-11-13/bin/darwin/amd64/kubectl.sha256
   ```

   - Kubernetes `1.31`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.31.13/2025-11-13/bin/
darwin/amd64/kubectl.sha256
```

- Kubernetes `1.30`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.14/2025-11-13/bin/
darwin/amd64/kubectl.sha256
```

- Kubernetes `1.29`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.15/2025-11-13/bin/
darwin/amd64/kubectl.sha256
```

- Kubernetes `1.28`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.28.15/2025-11-13/bin/
darwin/amd64/kubectl.sha256
```

- Kubernetes `1.27`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.16/2025-01-10/bin/
darwin/amd64/kubectl.sha256
```

- Kubernetes `1.26`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.26.15/2024-12-12/bin/
darwin/amd64/kubectl.sha256
```

b. Vérifiez le total de contrôle SHA-256 du fichier binaire téléchargé.

```
openssl sha1 -sha256 kubectl
```

c. Assurez-vous que le total de contrôle généré dans la sortie correspond à celui du fichier `kubectl.sha256` téléchargé.

3. Appliquez les autorisations d'exécution au fichier binaire.

```
chmod +x ./kubectl
```

4. Copiez le fichier binaire vers un dossier de votre PATH. Si vous avez déjà installé une version de `kubectl`, nous vous recommandons de créer un fichier `$HOME/bin/kubectl` et de vérifier que le fichier `$HOME/bin` est en premier dans votre `$PATH`.

```
mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export PATH=$HOME/bin:$PATH
```

5. (Facultatif) Ajoutez le chemin `$HOME/bin` à votre fichier d'initialisation du shell de façon à ce qu'il soit configuré lorsque vous ouvrez un shell.

```
echo 'export PATH=$HOME/bin:$PATH' >> ~/.bash_profile
```

Linux (amd64)

Suivez les étapes ci-dessous pour installer `kubectl` sous Linux (amd64). Les étapes sont les suivantes :

- Choisissez et téléchargez le fichier binaire correspondant à la version Kubernetes souhaitée.
- Vérification facultative de la somme de contrôle du fichier binaire.
- Ajouter l'exécution aux autorisations du fichier binaire.
- Copiez le fichier binaire dans un dossier de votre PATH.
- Ajouter éventuellement le répertoire du binaire à votre PATH.

Procédure :

1. Téléchargez le fichier binaire `kubectl` correspondant à la version Kubernetes de votre cluster depuis Amazon S3.
   - Kubernetes `1.34`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.34.2/2025-11-13/bin/linux/amd64/kubectl
     ```

   - Kubernetes `1.33`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.33.5/2025-11-13/bin/linux/amd64/kubectl
     ```

   - Kubernetes `1.32`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.32.9/2025-11-13/bin/linux/amd64/kubectl
```

- Kubernetes `1.31`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.31.13/2025-11-13/bin/linux/amd64/kubectl
```

- Kubernetes `1.30`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.14/2025-11-13/bin/linux/amd64/kubectl
```

- Kubernetes `1.29`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.15/2025-11-13/bin/linux/amd64/kubectl
```

- Kubernetes `1.28`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.28.15/2025-11-13/bin/linux/amd64/kubectl
```

- Kubernetes `1.27`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.16/2024-12-12/bin/linux/amd64/kubectl
```

- Kubernetes `1.26`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.26.15/2024-12-12/bin/linux/amd64/kubectl
```

2. (Facultatif) Vérifiez le fichier binaire téléchargé avec le total de contrôle `SHA-256` de votre fichier binaire.

   a. Téléchargez la somme de contrôle `SHA-256` pour la version Kubernetes de votre cluster depuis Amazon S3 à l'aide de la commande correspondant à la plate-forme matérielle de votre appareil.

      - Kubernetes `1.34`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.34.2/2025-11-13/bin/
linux/amd64/kubectl.sha256
```

- Kubernetes `1.33`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.33.5/2025-11-13/bin/
linux/amd64/kubectl.sha256
```

- Kubernetes `1.32`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.32.9/2025-11-13/bin/
linux/amd64/kubectl.sha256
```

- Kubernetes `1.31`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.31.13/2025-11-13/bin/
linux/amd64/kubectl.sha256
```

- Kubernetes `1.30`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.14/2025-11-13/bin/
linux/amd64/kubectl.sha256
```

- Kubernetes `1.29`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.15/2025-11-13/bin/
linux/amd64/kubectl.sha256
```

- Kubernetes `1.28`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.28.15/2025-11-13/bin/
linux/amd64/kubectl.sha256
```

- Kubernetes `1.27`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.16/2024-12-12/bin/
linux/amd64/kubectl.sha256
```

- Kubernetes `1.26`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.26.15/2024-12-12/bin/
linux/amd64/kubectl.sha256
```

b. Vérifiez le total de contrôle SHA-256 du fichier binaire que vous avez téléchargé à l'aide de l'une des commandes suivantes.

```
sha256sum -c kubectl.sha256
```

or

```
openssl sha1 -sha256 kubectl
```

c. Pour le premier, vous devriez voir `kubectl: OK`, pour le second, vous pouvez vérifier que la somme de contrôle générée dans la sortie correspond à celle du fichier `kubectl.sha256` téléchargé.

3. Appliquez les autorisations d'exécution au fichier binaire.

```
chmod +x ./kubectl
```

4. Copiez le fichier binaire vers un dossier de votre PATH. Si vous avez déjà installé une version de `kubectl`, nous vous recommandons de créer un fichier `$HOME/bin/kubectl` et de vérifier que le fichier `$HOME/bin` est en premier dans votre `$PATH`.

```
mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export PATH=$HOME/bin:$PATH
```

5. (Facultatif) Ajoutez le chemin `$HOME/bin` à votre fichier d'initialisation du shell de façon à ce qu'il soit configuré lorsque vous ouvrez un shell.

> **Note**
>
> Cette étape suppose que vous utilisiez le shell Bash ; si vous utilisez un autre shell, modifiez la commande pour utiliser votre propre fichier d'initialisation shell.

```
echo 'export PATH=$HOME/bin:$PATH' >> ~/.bashrc
```

Linux (arm64)

Suivez les étapes ci-dessous pour installer `kubectl` pour Linux (arm64). Les étapes sont les suivantes :

- Choisissez et téléchargez le fichier binaire correspondant à la version Kubernetes souhaitée.
- Vérification facultative de la somme de contrôle du fichier binaire.
- Ajouter l'exécution aux autorisations du fichier binaire.
- Copiez le fichier binaire dans un dossier de votre PATH.
- Ajouter éventuellement le répertoire du binaire à votre PATH.

Procédure :

1. Téléchargez le fichier binaire `kubectl` correspondant à la version Kubernetes de votre cluster depuis Amazon S3.
   - Kubernetes `1.34`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.34.2/2025-11-13/bin/linux/arm64/kubectl
     ```

   - Kubernetes `1.33`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.33.5/2025-11-13/bin/linux/arm64/kubectl
     ```

   - Kubernetes `1.32`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.32.9/2025-11-13/bin/linux/arm64/kubectl
     ```

   - Kubernetes `1.31`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.31.13/2025-11-13/bin/linux/arm64/kubectl
     ```

   - Kubernetes `1.30`

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.14/2025-11-13/bin/linux/arm64/kubectl
     ```

- Kubernetes `1.29`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.15/2025-11-13/bin/linux/arm64/kubectl
```

- Kubernetes `1.28`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.28.15/2025-11-13/bin/linux/arm64/kubectl
```

- Kubernetes `1.27`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.16/2024-12-12/bin/linux/arm64/kubectl
```

- Kubernetes `1.26`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.26.15/2024-12-12/bin/linux/arm64/kubectl
```

2. (Facultatif) Vérifiez le fichier binaire téléchargé avec le total de contrôle `SHA-256` de votre fichier binaire.

   a. Téléchargez la somme de contrôle `SHA-256` pour la version Kubernetes de votre cluster depuis Amazon S3 à l'aide de la commande correspondant à la plate-forme matérielle de votre appareil.

   - Kubernetes `1.34`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.34.2/2025-11-13/bin/linux/arm64/kubectl.sha256
```

   - Kubernetes `1.33`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.33.5/2025-11-13/bin/linux/arm64/kubectl.sha256
```

   - Kubernetes `1.32`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.32.9/2025-11-13/bin/linux/arm64/kubectl.sha256
```

- Kubernetes `1.31`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.31.13/2025-11-13/bin/linux/arm64/kubectl.sha256
```

- Kubernetes `1.30`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.14/2025-11-13/bin/linux/arm64/kubectl.sha256
```

- Kubernetes `1.29`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.15/2025-11-13/bin/linux/arm64/kubectl.sha256
```

- Kubernetes `1.28`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.28.15/2025-11-13/bin/linux/arm64/kubectl.sha256
```

- Kubernetes `1.27`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.16/2024-12-12/bin/linux/arm64/kubectl.sha256
```

- Kubernetes `1.26`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.26.15/2024-12-12/bin/linux/arm64/kubectl.sha256
```

b. Vérifiez le total de contrôle SHA-256 du fichier binaire que vous avez téléchargé à l'aide de l'une des commandes suivantes.

```
sha256sum -c kubectl.sha256
```

or

```
openssl sha1 -sha256 kubectl
```

c. Pour le premier, vous devriez voir `kubectl: OK`, pour le second, vous pouvez vérifier que la somme de contrôle générée dans la sortie correspond à celle du fichier `kubectl.sha256` téléchargé.

3. Appliquez les autorisations d'exécution au fichier binaire.

```
chmod +x ./kubectl
```

4. Copiez le fichier binaire vers un dossier de votre `PATH`. Si vous avez déjà installé une version de `kubectl`, nous vous recommandons de créer un fichier `$HOME/bin/kubectl` et de vérifier que le fichier `$HOME/bin` est en premier dans votre `$PATH`.

```
mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export PATH=$HOME/bin:$PATH
```

5. (Facultatif) Ajoutez le chemin `$HOME/bin` à votre fichier d'initialisation du shell de façon à ce qu'il soit configuré lorsque vous ouvrez un shell.

> **Note**
>
> Cette étape suppose que vous utilisiez le shell Bash ; si vous utilisez un autre shell, modifiez la commande pour utiliser votre propre fichier d'initialisation shell.

```
echo 'export PATH=$HOME/bin:$PATH' >> ~/.bashrc
```

Windows

Suivez les étapes ci-dessous pour installer `kubectl` sous Windows. Les étapes sont les suivantes :

- Choisissez et téléchargez le fichier binaire correspondant à la version Kubernetes souhaitée.
- Vérification facultative de la somme de contrôle du fichier binaire.
- Copiez le fichier binaire dans un dossier de votre PATH.
- Ajouter éventuellement le répertoire du binaire à votre PATH.

Procédure :

1. Ouvrez un PowerShell terminal.

2. Téléchargez le fichier binaire `kubectl` correspondant à la version Kubernetes de votre cluster depuis Amazon S3.

   - Kubernetes `1.34`

     ```
     curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.34.2/2025-11-13/bin/windows/amd64/kubectl.exe
     ```

   - Kubernetes `1.33`

     ```
     curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.33.5/2025-11-13/bin/windows/amd64/kubectl.exe
     ```

   - Kubernetes `1.32`

     ```
     curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.32.9/2025-11-13/bin/windows/amd64/kubectl.exe
     ```

   - Kubernetes `1.31`

     ```
     curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.31.13/2025-11-13/bin/windows/amd64/kubectl.exe
     ```

   - Kubernetes `1.30`

     ```
     curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.14/2025-11-13/bin/windows/amd64/kubectl.exe
     ```

   - Kubernetes `1.29`

     ```
     curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.15/2025-11-13/bin/windows/amd64/kubectl.exe
     ```

   - Kubernetes `1.28`

     ```
     curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.28.15/2025-11-13/bin/windows/amd64/kubectl.exe
     ```

   - Kubernetes `1.27`

     ```
     curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.16/2024-12-12/bin/windows/amd64/kubectl.exe
     ```

- Kubernetes `1.26`

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.26.15/2024-12-12/bin/windows/amd64/kubectl.exe
```

3. (Facultatif) Vérifiez le fichier binaire téléchargé avec le total de contrôle SHA-256 de votre fichier binaire.

   a. Téléchargez la somme SHA-256 pour la version Kubernetes de votre cluster pour Windows.

   - Kubernetes `1.34`

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.34.2/2025-11-13/bin/windows/amd64/kubectl.exe.sha256
```

   - Kubernetes `1.33`

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.33.5/2025-11-13/bin/windows/amd64/kubectl.exe.sha256
```

   - Kubernetes `1.32`

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.32.9/2025-11-13/bin/windows/amd64/kubectl.exe.sha256
```

   - Kubernetes `1.31`

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.31.13/2025-11-13/bin/windows/amd64/kubectl.exe.sha256
```

   - Kubernetes `1.30`

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.30.14/2025-11-13/bin/windows/amd64/kubectl.exe.sha256
```

   - Kubernetes `1.29`

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.15/2025-11-13/bin/windows/amd64/kubectl.exe.sha256
```

   - Kubernetes `1.28`

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.28.15/2025-11-13/bin/windows/amd64/kubectl.exe.sha256
```

- Kubernetes `1.27`

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.16/2024-12-12/bin/windows/amd64/kubectl.exe.sha256
```

- Kubernetes `1.26`

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.26.15/2024-12-12/bin/windows/amd64/kubectl.exe.sha256
```

b. Vérifiez le total de contrôle `SHA-256` du fichier binaire téléchargé.

```
Get-FileHash kubectl.exe
```

c. Assurez-vous que le total de contrôle généré dans la sortie correspond à celui du fichier `kubectl.sha256` téléchargé. La PowerShell sortie doit être une chaîne de caractères équivalente en majuscules.

4. Copiez le fichier binaire vers un dossier de votre `PATH`. Si vous disposez déjà d'un répertoire dans votre `PATH` que vous utilisez pour les utilitaires de ligne de commande, copiez le fichier binaire dans ce répertoire. Sinon, effectuez les étapes suivantes.
   a. Créez un répertoire pour vos fichiers binaires de ligne de commande, par exemple `C:\bin`.
   b. Copiez le fichier binaire `kubectl.exe` vers votre nouveau répertoire.
   c. Modifiez la variable d'environnement `PATH` de votre utilisateur ou système pour ajouter le nouveau répertoire à votre variable `PATH`.
   d. Fermez votre PowerShell terminal et ouvrez-en un nouveau pour récupérer la nouvelle `PATH` variable.
5. Après avoir installé `kubectl`, vous pouvez vérifier sa version.

```
kubectl version --client
```

6. Lors de la première installation de `kubectl`, il n'est pas encore configuré pour communiquer avec un serveur. Nous aborderons cette configuration au besoin dans d'autres procédures. Si vous devez mettre à jour la configuration pour communiquer avec un cluster en particulier, vous pouvez

exécuter la commande suivante. Remplacez *region-code* par la AWS région dans laquelle se trouve votre cluster. Remplacez *my-cluster* par le nom de votre cluster.

```
aws eks update-kubeconfig --region region-code --name my-cluster
```

7. Envisagez de configurer la saisie automatique, qui vous permet d'utiliser la touche de tabulation pour exécuter les sous-commandes `kubectl` après avoir saisi les premières lettres. Voir Kubectl autocomplete dans la documentation de Kubernetes pour plus de détails.

Installer **`eksctl`**

La CLI `eksctl` est utilisée pour travailler avec des clusters EKS. Il automatise de nombreuses tâches individuelles. Consultez la section Installation dans la documentation `eksctl` pour obtenir des instructions sur l'installation `eksctl`. Pour Linux, utilisez les instructions UNIX.

Lorsque vous utilisez `eksctl` le principal de sécurité IAM que vous utilisez, vous devez disposer des autorisations nécessaires pour utiliser les rôles Amazon EKS IAM, les rôles liés à un service AWS CloudFormation, un VPC et les ressources associées. Pour plus d'informations, consultez les sections Actions et Utilisation des rôles liés aux services dans le Guide de l'utilisateur IAM. Vous devez effectuer toutes les étapes de ce guide avec le même utilisateur. Exécutez la commande suivante pour vérifier l'utilisateur actuel :

```
aws sts get-caller-identity
```

Étapes suivantes

- Démarrage rapide : déploiement d'une application Web et stockage des données

Démarrage rapide : déploiement d'une application Web et stockage des données

Ce didacticiel de démarrage rapide vous explique les étapes à suivre pour déployer l'exemple d'application de jeu 2048 et conserver ses données sur un cluster Amazon EKS Auto Mode à l'aide d'eksctl.

Le mode automatique d'Amazon EKS simplifie la gestion des clusters en automatisant les tâches de routine telles que le stockage par blocs, la mise en réseau, l'équilibrage de charge et le dimensionnement automatique du calcul. Lors de l'installation, il gère la création de nœuds avec des instances EC2 gérées, des équilibreurs de charge d'applications et des volumes EBS.

En résumé, vous allez déployer un exemple de charge de travail avec les annotations personnalisées nécessaires pour une intégration fluide aux AWS services.

Dans ce tutoriel

À l'aide du modèle de cluster `eksctl` ci-dessous, vous créez un cluster avec le mode automatique EKS pour le provisionnement automatisé des nœuds.

- Configuration VPC : Lorsque vous utilisez le modèle de cluster eksctl ci-dessous, eksctl crée automatiquement un Virtual Private Cloud (IPv4 VPC) pour le cluster. Par défaut, eksctl configure un VPC qui répond à toutes les exigences de mise en réseau, en plus de créer des points de terminaison publics et privés.
- Gestion des instances : le mode automatique EKS ajoute ou supprime dynamiquement des nœuds dans votre cluster EKS en fonction des demandes de vos applications Kubernetes.
- Persistance des données : utilisez la capacité de stockage par blocs du mode automatique d'EKS pour garantir la persistance des données d'application, même dans les scénarios impliquant le redémarrage ou la défaillance du pod.
- Accès aux applications externes : utilisez la fonctionnalité d'équilibrage de charge du mode automatique d'EKS pour provisionner dynamiquement un Application Load Balancer (ALB).

Conditions préalables

Avant de commencer, assurez-vous d'avoir effectué les tâches suivantes :

- Configuration de votre environnement pour Amazon EKS
- Installation de la dernière version d'eksctl

Configurer le cluster

Dans cette section, vous allez créer un cluster à l'aide du mode automatique EKS pour le provisionnement dynamique des nœuds.

Créez un fichier `cluster-config.yaml` et collez-y le contenu suivant. Remplacez `region-code` par une région valide (par exemple,`us-east-1`).

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: web-quickstart
  region: <region-code>

autoModeConfig:
  enabled: true
```

Nous sommes maintenant prêts à créer le cluster.

Créez le cluster EKS à l'aide de `cluster-config.yaml`` :

```
eksctl create cluster -f cluster-config.yaml
```

> **Important**
>
> Si vous n'utilisez pas eksctl pour créer le cluster, vous devez baliser manuellement les sous-réseaux VPC.

Créez IngressClass

Créez une `IngressClass` Kubernetes pour le mode automatique EKS. IngressClass définit la manière dont le mode automatique EKS gère les ressources d'entrée. Cette étape configure la capacité d'équilibrage de charge du mode automatique EKS. Lorsque vous créez des ressources

d'entrée pour vos applications, le mode automatique d'EKS les utilise pour provisionner et gérer automatiquement les équilibreurs de charge, en intégrant vos applications Kubernetes aux services d'équilibrage de charge. IngressClass AWS

Enregistrez le fichier yaml suivant sous le nom `ingressclass.yaml` :

```
apiVersion: networking.k8s.io/v1
kind: IngressClass
metadata:
  name: alb
  annotations:
    ingressclass.kubernetes.io/is-default-class: "true"
spec:
  controller: eks.amazonaws.com/alb
```

Appliquez le IngressClass à votre cluster :

```
kubectl apply -f ingressclass.yaml
```

Déployer l'application exemple de jeu 2048

Dans cette section, nous vous guidons à travers les étapes de déploiement du célèbre « jeu 2048 » en tant qu'application exemple au sein du cluster. Le manifeste fourni comprend des annotations personnalisées pour l'Application Load Balancer (ALB). Ces annotations s'intègrent à EKS et indiquent à EKS de traiter le trafic HTTP entrant comme « orienté vers Internet » et de l'acheminer vers le service approprié dans l'`game-2048`espace de noms en utilisant le type de cible « ip ».

> **Note**
>
> L'image `docker-2048` de l'exemple est une image de conteneur `x86_64` et ne fonctionnera pas sur d'autres architectures.

1. Créez un espace de noms Kubernetes appelé `game-2048` avec l'indicateur `--save-config`.

   ```
   kubectl create namespace game-2048 --save-config
   ```

   Le résultat de la réponse devrait être le suivant :

```
namespace/game-2048 created
```

2. Déployez l'application exemple de jeu 2048.

```
kubectl apply -n game-2048 -f https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.8.0/docs/examples/2048/2048_full.yaml
```

   Ce manifeste configure un déploiement, un service et un ingress Kubernetes pour l'espace de noms `game-2048`, créant ainsi les ressources nécessaires pour déployer et exposer l'application `game-2048` au sein du cluster. Il inclut la création d'un service nommé `service-2048` qui expose le déploiement sur le port `80`, et une ressource Ingress nommée `ingress-2048` qui définit les règles de routage pour le trafic HTTP entrant et les annotations pour un Application Load Balancer (ALB) connecté à Internet. Le résultat de la réponse devrait être le suivant :

```
namespace/game-2048 configured
deployment.apps/deployment-2048 created
service/service-2048 created
ingress.networking.k8s.io/ingress-2048 created
```

3. Exécutez la commande suivante pour obtenir la ressource Ingress pour l'espace de noms `game-2048`.

```
kubectl get ingress -n game-2048
```

   Le résultat de la réponse devrait être le suivant :

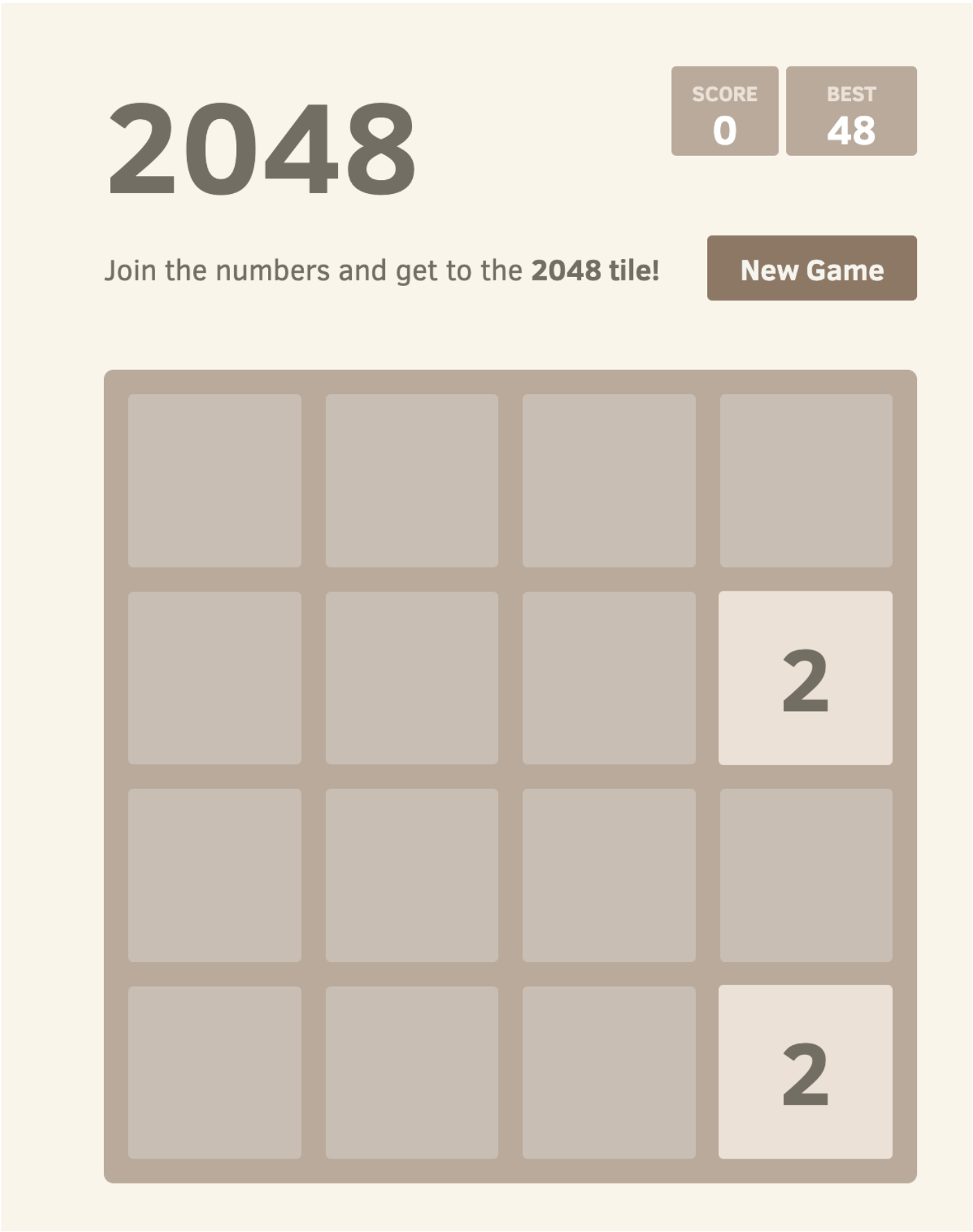
```
NAME           CLASS   HOSTS   ADDRESS
                      PORTS   AGE
ingress-2048   alb     *       k8s-game2048-ingress2-eb379a0f83-378466616.region-
code.elb.amazonaws.com   80      31s
```

   Vous devrez patienter quelques minutes pour que l'Application Load Balancer (ALB) soit provisionné avant de passer aux étapes suivantes.

4. Ouvrez un navigateur Web et saisissez l'`ADDRESS` de l'étape précédente pour accéder à l'application Web. Par exemple :

```
k8s-game2048-ingress2-eb379a0f83-378466616.region-code.elb.amazonaws.com
```

Le jeu 2048 devrait s’afficher dans votre navigateur. Lire !


2048
SCORE
0
BEST
48
Join the numbers and get to the 2048 tile!
New Game
2
2

Conserver les données à l'aide du mode automatique Amazon EKS

Maintenant que le jeu 2048 est opérationnel sur votre cluster Amazon EKS, il est temps de vous assurer que les données de votre jeu sont conservées en toute sécurité à l'aide de la fonctionnalité de stockage en bloc du mode automatique Amazon EKS.

1. Créez un fichier nommé `storage-class.yaml`:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: auto-ebs-sc
  annotations:
    storageclass.kubernetes.io/is-default-class: "true"
provisioner: ebs.csi.eks.amazonaws.com
volumeBindingMode: WaitForFirstConsumer
parameters:
  type: gp3
  encrypted: "true"
```

2. Appliquez le `StorageClass` :

```
kubectl apply -f storage-class.yaml
```

3. Créez une demande de volume persistant (PVC) pour demander du stockage pour les données de votre jeu. Créez un fichier nommé `ebs-pvc.yaml` et ajoutez-y le contenu suivant :

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: game-data-pvc
  namespace: game-2048
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
  storageClassName: auto-ebs-sc
```

4. Appliquez le PVC à votre cluster :

```
kubectl apply -f ebs-pvc.yaml
```

Le résultat de la réponse devrait être le suivant :

```
persistentvolumeclaim/game-data-pvc created
```

5. Vous devez maintenant mettre à jour le déploiement de votre jeu 2048 afin d'utiliser ce PVC pour stocker les données. Le déploiement suivant est configuré pour utiliser le PVC pour stocker les données du jeu. Créez un fichier nommé `ebs-deployment.yaml` et ajoutez-y le contenu suivant :

```
apiVersion: apps/v1
kind: Deployment
metadata:
  namespace: game-2048
  name: deployment-2048
spec:
  replicas: 3  # Adjust the number of replicas as needed
  selector:
    matchLabels:
      app.kubernetes.io/name: app-2048
  template:
    metadata:
      labels:
        app.kubernetes.io/name: app-2048
    spec:
      containers:
        - name: app-2048
          image: public.ecr.aws/l6m2t8p7/docker-2048:latest
          imagePullPolicy: Always
          ports:
            - containerPort: 80
          volumeMounts:
            - name: game-data
              mountPath: /var/lib/2048
      volumes:
        - name: game-data
          persistentVolumeClaim:
            claimName: game-data-pvc
```

6. Appliquez le déploiement mis à jour :

```
kubectl apply -f ebs-deployment.yaml
```

Le résultat de la réponse devrait être le suivant :

```
deployment.apps/deployment-2048 configured
```

Grâce à ces étapes, votre jeu 2048 sur le cluster est désormais configuré pour conserver les données à l'aide de la fonctionnalité de stockage en blocs du mode automatique Amazon EKS. Cela garantit la sécurité de la progression et des données de votre jeu, même en cas de défaillance d'un pod ou d'un nœud.

Si vous avez apprécié ce tutoriel, veuillez nous en faire part afin que nous puissions vous proposer d'autres tutoriels de démarrage rapide spécifiques à des cas d'utilisation comme celui-ci.

Nettoyage

Pour éviter d'encourir de futurs frais, vous devez supprimer manuellement la CloudFormation pile associée pour supprimer toutes les ressources créées au cours de ce guide, y compris le réseau VPC.

Supprimez la CloudFormation pile :

```
eksctl delete cluster -f ./cluster-config.yaml
```

Découvrez Amazon EKS par l'exemple

Présentation

Ce Guide de l'utilisateur Amazon EKS contient des procédures générales pour créer votre premier cluster EKS à partir de la ligne de commande ou AWS Management Console une référence solide pour tous les principaux composants Amazon EKS. Cependant, en tant qu'administrateur ou développeur de cluster Amazon EKS, vous pouvez approfondir vos connaissances sur Amazon EKS en suivant les parcours d'apprentissage proposés sur des sites externes à ce guide. Ces sites sont là pour vous aider à :

- Configurez des types spécifiques de clusters. Les types de clusters spécifiques peuvent être basés sur vos types de charge de travail ou vos exigences en matière de sécurité. Par exemple, vous souhaiterez peut-être régler un cluster pour exécuter des tâches par lots, de machine learning ou des charges de travail gourmandes en ressources de calcul.
- Améliorez vos clusters. Vous pouvez ajouter des fonctionnalités avancées à votre cluster afin de bénéficier d'observabilité, d'un stockage flexible, d'une mise à l'échelle automatique ou d'une mise en réseau spécialisée du cluster.
- Automatisez les mises à jour. Grâce à des fonctionnalités telles que GitOps, vous pouvez configurer le provisionnement automatique de l'infrastructure du cluster et des charges de travail, en fonction des modifications apportées à ces composants dans vos référentiels Git.
- Utilisez des outils de configuration de cluster avancés. Bien que `eksctl` permette de créer rapidement un cluster, il existe d'autres outils qui facilitent la configuration et la mise à niveau de clusters plus complexes. Il s'agit notamment d'outils tels que Terraform et CloudFormation.

Pour commencer votre apprentissage d'Amazon EKS, je vous recommande de consulter certains des sites décrits sur cette page. Si vous rencontrez des problèmes en cours de route, il existe également des ressources pour vous aider à les surmonter. Par exemple, le centre de connaissances Re:post vous permet d'effectuer des recherches dans la base de données d'assistance pour trouver des problèmes liés à Amazon EKS. Le guide des meilleures pratiques Amazon EKS propose également des conseils sur les meilleures façons de configurer vos clusters de production.

Atelier Amazon EKS

Partant d'une compréhension de base de Kubernetes et des conteneurs, l'atelier Amazon EKS est une plateforme d'apprentissage qui guide les administrateurs de clusters à travers les fonctionnalités importantes d'Amazon EKS. Voici comment vous pouvez participer à l'atelier Amazon EKS :

- Notions de base sur Amazon EKS : Regardez la vidéo sur la page Introduction pour découvrir comment Amazon EKS implémente les fonctionnalités Kubernetes dans le cloud AWS. Si vous avez besoin d'une compréhension encore plus élémentaire de Kubernetes, regardez la vidéo Qu'est-ce que Kubernetes ?
- Configuration d'Amazon EKS : Si vous avez un compte AWS, la section Configuration vous aide à configurer un environnement CloudShell pour créer un cluster. Il propose le choix entre eksctl (une simple ligne de commande de création de cluster) et Terraform (une approche davantage axée sur l'infrastructure en tant que code pour créer un cluster) pour créer votre cluster Amazon EKS.
- Amazon EKS Pour démarrer : essayez une boutique en ligne simple dans la section Exemple d'application. Vous pouvez l'utiliser dans les autres exercices. Dans cette section, vous pouvez également découvrir comment empaqueter des images de conteneur et comment les microservices sont gérés à l'aide des pods, déploiements, services, StatefulSets et espaces de noms Kubernetes. Utilisez ensuite Kustomize pour déployer les modifications dans les manifestes Kubernetes.
- Principes fondamentaux d'Amazon EKS : à l'aide de fonctionnalités AWS telles qu'AWS Load Balancer Controller, cet atelier vous montre comment exposer vos applications au monde extérieur. En matière de stockage, l'atelier présente comment utiliser Amazon EBS pour le stockage en blocs, Amazon EFS pour le stockage de systèmes de fichiers et Amazon FSx pour NetApp ONTAP afin de gérer les systèmes de fichiers ONTAP dans AWS. Pour la gestion des nœuds, l'atelier vous aide à configurer des groupes de nœuds gérés.
- Fonctionnalités avancées d'Amazon EKS : Les fonctionnalités avancées proposées dans le cadre de l'atelier Amazon EKS comprennent des laboratoires pour la configuration :
  - Autoscaling : Cela inclut le dimensionnement automatique des nœuds (avec Cluster Autoscaler ou Karpenter) et le dimensionnement automatique de la charge de travail (avec Horizontal Pod Autoscaler et Cluster Proportional Autoscaler).
  - Observabilité : découvrez la journalisation, OpenSearch, Container Insights sur Amazon EKS et la visibilité des coûts avec Kubecost dans un ensemble de laboratoires d'observabilité.
  - Sécurité : Cet ensemble de laboratoires de sécurité vous permet d'explorer la gestion des secrets, Amazon GuardDuty, les normes de sécurité des Pod et la gestion des politiques Kyverno.

- Réseautage : Découvrez les fonctionnalités de mise en réseau pour Amazon EKS grâce aux laboratoires de mise en réseau qui incluent Amazon VPC CNI (prenant en charge les plug-ins réseau) et Amazon VPC Lattice (pour configurer des clusters sur plusieurs comptes VC et utilisateurs).
- Automatisation : les laboratoires sur l'automatisation vous guident à travers les méthodes GitOps de gestion de vos clusters et projets, tels que AWS Controllers for Kubernetes et Crossplane pour la gestion des plans de contrôle Amazon EKS.

Tutoriels pratiques de configuration de clusters Amazon EKS

Une série de tutoriels sur la configuration des clusters Amazon EKS disponibles sur le site communautaire AWS peut vous aider à créer des clusters Amazon EKS à usage spécifique et à les améliorer de différentes manières. Les tutoriels sont divisés en trois types différents :

Création de clusters

Ces tutoriels vous aident à créer des clusters qui peuvent être utilisés à des fins spécifiques. Ces objectifs spéciaux incluent la possibilité d'exécuter :

- Applications évolutives à l'échelle mondiale basées sur IPv6
- Tâches par lots asynchrones
- Microservices à fort trafic
- Autoscaling avec Karpenter sur Fargate
- Charges de travail financières
- Groupes de nœuds gérés Windows

Améliorer les clusters

Une fois que vous disposez d'un cluster existant, vous pouvez l'étendre et l'améliorer de manière à lui permettre d'exécuter des charges de travail spécialisées et à améliorer les clusters. Ces tutoriels proposent des méthodes pour :

- Fournir des solutions de stockage avec EFS CSI
- Fournir un stockage de base de données dynamique avec EBS CSI
- Exposer des applications sur des clusters IPv4 à l'aide d'AWS Load Balancer Controller

- Exposer des applications sur des clusters IPv6 à l'aide d'AWS Load Balancer Controller

Optimisation des services AWS

À l'aide de ces tutoriels, vous pouvez mieux intégrer vos clusters aux services AWS. Ces tutoriels incluent ceux qui vous aideront à :

- Gérer les enregistrements DNS pour les microservices avec ExternalDNS
- Surveiller les applications avec CloudWatch
- Gérer les tâches asynchrones avec le stockage SQS et EFS
- Consommer les secrets AWS Secrets Manager provenant des charges de travail
- Configurer des MTL avec Fargate, NGINX et ACM PCA

Echantillons Amazon EKS

Le référentiel Amazon EKS Samples stocke les manifestes à utiliser avec Amazon EKS. Ces manifestes vous permettent d'essayer différents types d'applications dans Amazon EKS ou de créer des types spécifiques de clusters Amazon EKS. Les exemples incluent des manifestes destinés à :

- Créer un cluster AWS Fargate Amazon EKS
- Créer un cluster avec un rôle IAM existant
- Ajouter un groupe de nœuds gérés par Ubuntu à un cluster
- Sauvegarder et restaurer le stockage du pod avec des instantanés de volume
- Restaurer des volumes EBS montés sous forme de PVC avec plusieurs comptes
- Activer le protocole proxy pour le contrôleur d'entrée NGINX avec les Classic Load Balancers
- Configurer la journalisation sur Fargate dans OpenSearch AWS
- Exécuter le kit SDK Python avec un fournisseur d'identité fédéré Web
- Déployer un exemple d'application sur un contrôleur NFS CSI
- Utiliser des instantanés de volume pour les StatefulSets
- Déployer des pods sur des nœuds situés dans différentes zones de disponibilité

Gardez à l'esprit que ces exemples sont uniquement destinés à des fins d'apprentissage et de test et ne sont pas destinés à être utilisés en production.

Didacticiels AWS

Le site AWS Tutorials publie quelques tutoriels Amazon EKS, mais propose également un outil de recherche permettant de trouver d'autres tutoriels publiés sur d'autres sites AWS (tels que le site AWS Community). Les tutoriels Amazon EKS publiés directement sur ce site comprennent :

- Déployer une application Web de conteneur sur Amazon EKS
- Exécuter des clusters Kubernetes à moindre coût (instances Amazon EKS et Spot)
- Comment optimiser les coûts des tâches Jenkins sur Kubernetes

Ateliers pour développeurs

Si vous êtes développeur de logiciels et que vous souhaitez créer ou refactoriser des applications pour les exécuter sur Amazon EKS, l'atelier Amazon EKS Developers est un bon point de départ. L'atelier vous aide non seulement à créer des applications conteneurisées, mais aussi à déployer ces conteneurs dans un registre de conteneurs (ECR), puis dans un cluster Amazon EKS.

Commencez par l'atelier Amazon EKS Python pour découvrir le processus de refactorisation d'une application Python, puis configurez votre environnement de développement afin de préparer le déploiement de l'application. Parcourez les sections consacrées aux conteneurs, à Kubernetes et à Amazon EKS afin de vous préparer à exécuter vos applications conteneurisées dans ces environnements.

Ateliers Terraform

Bien que `eksctl` soit un outil simple pour créer un cluster, Terraform est un outil populaire de création et de gestion de clusters Amazon EKS pour les déploiements Amazon EKS plus complexes de type « infrastructure-as-code ». L'atelier Terraform Amazon EKS vous apprend à utiliser Terraform pour créer un VPC AWS, créer des clusters Amazon EKS et ajouter des améliorations facultatives à votre cluster. Il existe notamment une section dédiée à la création d'un cluster Amazon EKS privé.

AWS Formation Amazon EKS

AWS propose une formation officielle pour en savoir plus sur Amazon EKS. Une formation de trois jours intitulée Running Containers on Amazon Elastic Kubernetes Service (Exécuter des conteneurs sur Amazon Elastic Kubernetes Service) enseigne :

- Principes fondamentaux de Kubernetes et Amazon EKS
- Comment créer des clusters Amazon EKS
- Sécurisation d'Amazon EKS avec l'autorisation AWS IAM et Kubernetes RBAC
- Outils d'automatisation GitOps
- Outils de surveillance
- Techniques pour améliorer les coûts, l'efficacité et la résilience

Mise en route avec Amazon EKS

Assurez-vous que votre configuration vous permet d'utiliser Amazon EKS avant de consulter les guides de démarrage. Pour de plus amples informations, consultez Configuration.

Il existe deux guides de démarrage pour créer un nouveau cluster Kubernetes avec les nœuds dans Amazon EKS :

- Démarrage avec Amazon EKS – eksctl : ce guide de démarrage vous permet d'installer toutes les ressources nécessaires pour commencer à utiliser Amazon EKS avec `eksctl`, un utilitaire de ligne de commande simple pour créer et gérer les clusters Kubernetes sur Amazon EKS. À la fin de ce didacticiel, vous disposerez d'un cluster Amazon EKS en cours d'exécution sur lequel vous pouvez déployer des applications. Il s'agit de la solution la plus simple et la plus rapide pour commencer à utiliser Amazon EKS.
- Démarrage avec Amazon EKS – AWS Management Console et AWS CLI : ce guide de démarrage vous permet de créer toutes les ressources nécessaires pour commencer à utiliser Amazon EKS à l'aide de AWS Management Console et CLI AWS. À la fin de ce didacticiel, vous disposerez d'un cluster Amazon EKS en cours d'exécution sur lequel vous pouvez déployer des applications. Dans ce guide, vous créez manuellement chaque ressource requise pour un cluster Amazon EKS. Les procédures vous donnent une visibilité sur la façon dont chaque ressource est créée et la manière dont elles interagissent les unes avec les autres.

Nous proposons également les références suivantes :

- Pour obtenir une série de tutoriels pratiques, consultez la rubrique Configuration d'un cluster EKS sur la Communauté AWS.
- Pour des exemples de code, consultez Exemples de code pour Amazon EKS utilisant des SDKAWS.

Démarrage avec Amazon EKS – Mode automatique EKS

Comme pour les autres expériences de démarrage EKS, la création de votre premier cluster avec le mode automatique EKS délègue la gestion du cluster lui-même à AWS. Cependant, le mode automatique EKS étend l'automatisation EKS en prenant en charge de nombreux services essentiels nécessaires à la mise en place de l'infrastructure de charge de travail (nœuds, réseaux et

divers services), ce qui facilite la gestion des nœuds et l'augmentation verticale pour répondre aux demandes de charge de travail.

Choisissez l'une des méthodes suivantes pour créer un cluster avec le mode automatique EKS :

- the section called " AWS CLI" : utilisez l'interface de ligne de commande `aws` pour créer un cluster.
- the section called "Console de gestion" : utilisez la AWS Management Console pour créer un cluster.
- the section called "CLI eksctl" : utilisez l'interface de ligne de commande `eksctl` pour créer un cluster.

Si vous comparez différentes approches pour créer votre premier cluster EKS, il est important de savoir que le mode automatique EKS confie à AWS des responsabilités supplémentaires en matière de gestion du cluster, notamment la configuration des composants pour :

- Démarrer et mettre à l'échelle les nœuds à mesure que la charge de travail augmente ou diminue.
- Mettre régulièrement à niveau le cluster lui-même (plan de contrôle), les systèmes d'exploitation des nœuds et les services exécutés sur les nœuds.
- Sélectionner les paramètres par défaut qui déterminent des éléments tels que la taille et la vitesse du stockage des nœuds et la configuration réseau des pods.

Pour plus d'informations sur ce que vous obtenez avec les clusters du mode automatique EKS, consultez Mode automatique EKS.

Démarrage avec Amazon EKS – **`eksctl`**

> **Note**
>
> Cette rubrique traite du démarrage sans le mode automatique EKS.
> Le mode automatique EKS automatise les tâches courantes liées au calcul, au stockage et à la mise en réseau des clusters. Découvrez comment démarrer avec le mode automatique Amazon EKS.

Ce guide vous permet de créer toutes les ressources nécessaires pour commencer à utiliser Amazon Elastic Kubernetes Service (Amazon EKS) avec `eksctl`, un utilitaire de ligne de commande

simple pour créer et gérer les clusters Kubernetes sur Amazon EKS. À la fin de ce didacticiel, vous disposerez d'un cluster Amazon EKS en cours d'exécution sur lequel vous pouvez déployer des applications.

Les procédures de ce guide créent automatiquement plusieurs ressources que vous devez créer manuellement lorsque vous créez votre cluster à l'aide de l'outil AWS Management Console. Si vous préférez créer manuellement la plupart des ressources pour mieux comprendre comment elles interagissent les unes avec les autres, utilisez le AWS Management Console pour créer votre cluster et effectuer vos calculs. Pour de plus amples informations, consultez the section called “Création d'un cluster (console et CLI)”.

Prérequis

Avant de commencer ce tutoriel, vous devez installer et configurer les outils CLI AWS, kubectl et eksctl comme décrit dans la section Configuration pour utiliser Amazon EKS.

Étape 1 : créer votre cluster Amazon EKS et votre nœud

> **Important**
>
> Pour démarrer le plus simplement et le plus rapidement possible, cette rubrique inclut les étapes pour créer un cluster et des nœuds avec des paramètres par défaut. Avant de créer un cluster et des nœuds pour une utilisation en production, nous vous recommandons de vous familiariser avec tous les paramètres et de déployer un cluster et des nœuds avec les paramètres qui répondent à vos besoins. Pour plus d'informations, consultez the section called “Créer un cluster” et Gestion du Calcul. Certains paramètres ne peuvent être activés que lors de la création de votre cluster et de vos nœuds.

Vous pouvez créer un cluster à l'aide de l'un des types de nœuds suivants. Pour en savoir plus sur chaque type, consultez Gestion du Calcul. Une fois votre cluster déployé, vous pouvez ajouter d'autres types de nœuds.

- Fargate – Linux : sélectionnez ce type de nœud si vous voulez exécuter des applications Linux sur the section called “ AWS Fargate”. Fargate est un moteur de calcul sans serveur qui vous permet de déployer des pods Kubernetes sans gérer les instances Amazon EC2.

- Nœuds gérés – Linux : sélectionnez ce type de nœud si vous souhaitez exécuter des applications Amazon Linux sur des instances Amazon EC2. Bien que cela ne soit pas inclus dans ce guide, vous pouvez également ajouter les nœuds autogérés par Windows et Bottlerocket à votre cluster.

Créez votre cluster Amazon EKS à l'aide de la commande suivante. Vous pouvez remplacer *my-cluster* par votre propre valeur. Un nom ne peut contenir que des caractères alphanumériques (sensibles à la casse) et des traits d'union. Il doit commencer par un caractère alphanumérique et ne peut pas dépasser 100 caractères. Le nom doit être unique dans la région AWS et le compte AWS dans lesquels vous créez le cluster. Remplacez *region-code* par toute région AWS prise en charge par Amazon EKS. Pour obtenir la liste des régions AWS, consultez la section Points de terminaison et quotas Amazon EKS dans le guide de référence générale AWS.

Example

Fargate - Linux

```
eksctl create cluster --name my-cluster --region region-code --fargate
```

Managed nodes - Linux

```
eksctl create cluster --name my-cluster --region region-code
```

La création du cluster prend plusieurs minutes. Pendant la création, vous verrez plusieurs lignes de sortie. La dernière ligne de sortie est similaire à celle de l'exemple suivant.

```
[...]
[#]  EKS cluster "my-cluster" in "region-code" region is ready
```

`eksctl` a créé un fichier de configuration `kubectl` dans `~/.kube/config` ou ajouté la configuration du nouveau cluster dans un fichier de configuration dans `~/.kube/config` sur votre ordinateur.

Une fois la création du cluster terminée, affichez la pile CloudFormation AWS nommée `eksctl-`*my-cluster*`-cluster` dans la console CloudFormation AWS pour voir toutes les ressources qui ont été créées.

Étape 2 : Afficher les ressources Kubernetes

1. Affichez vos nœuds de cluster.

```
kubectl get nodes -o wide
```

L'exemple qui suit illustre un résultat.

Example

Fargate - Linux

```
NAME                                                  STATUS   ROLES    AGE
 VERSION              INTERNAL-IP   EXTERNAL-IP   OS-IMAGE         KERNEL-VERSION
                CONTAINER-RUNTIME
fargate-ip-192-0-2-0.region-code.compute.internal   Ready    <none>
 8m3s    v1.2.3-eks-1234567   192.0.2.0     <none>        Amazon Linux 2
 1.23.456-789.012.amzn2.x86_64   containerd://1.2.3
fargate-ip-192-0-2-1.region-code.compute.internal   Ready    <none>
 7m30s   v1.2.3-eks-1234567   192-0-2-1     <none>        Amazon Linux 2
 1.23.456-789.012.amzn2.x86_64   containerd://1.2.3
```

Managed nodes - Linux

```
NAME                                        STATUS   ROLES    AGE    VERSION
        INTERNAL-IP   EXTERNAL-IP   OS-IMAGE         KERNEL-VERSION
   CONTAINER-RUNTIME
ip-192-0-2-0.region-code.compute.internal   Ready    <none>   6m7s
  v1.2.3-eks-1234567   192.0.2.0     192.0.2.2     Amazon Linux 2
 1.23.456-789.012.amzn2.x86_64   containerd://1.2.3
ip-192-0-2-1.region-code.compute.internal   Ready    <none>   6m4s
  v1.2.3-eks-1234567   192.0.2.1     192.0.2.3     Amazon Linux 2
 1.23.456-789.012.amzn2.x86_64   containerd://1.2.3
```

Pour plus d'informations sur les éléments que vous voyez en sortie, consultez the section called "Accéder aux ressources du cluster".

2. Affichez les applications en cours d'exécution sur votre cluster.

```
kubectl get pods -A -o wide
```

L'exemple qui suit illustre un résultat.

Example

Fargate - Linux

```
NAMESPACE     NAME                       READY   STATUS    RESTARTS   AGE   IP
          NODE                                                NOMINATED NODE
 READINESS GATES
kube-system   coredns-1234567890-abcde   1/1     Running   0          18m
 192.0.2.0   fargate-ip-192-0-2-0.region-code.compute.internal   <none>
 <none>
kube-system   coredns-1234567890-12345   1/1     Running   0          18m
 192.0.2.1   fargate-ip-192-0-2-1.region-code.compute.internal   <none>
 <none>
```

Managed nodes - Linux

```
NAMESPACE     NAME                       READY   STATUS    RESTARTS   AGE     IP
          NODE                                          NOMINATED NODE   READINESS
 GATES
kube-system   aws-node-12345             1/1     Running   0          7m43s
 192.0.2.1   ip-192-0-2-1.region-code.compute.internal   <none>           <none>
kube-system   aws-node-67890             1/1     Running   0          7m46s
 192.0.2.0   ip-192-0-2-0.region-code.compute.internal   <none>           <none>
kube-system   coredns-1234567890-abcde   1/1     Running   0          14m
 192.0.2.3   ip-192-0-2-3.region-code.compute.internal   <none>           <none>
kube-system   coredns-1234567890-12345   1/1     Running   0          14m
 192.0.2.4   ip-192-0-2-4.region-code.compute.internal   <none>           <none>
kube-system   kube-proxy-12345           1/1     Running   0          7m46s
 192.0.2.0   ip-192-0-2-0.region-code.compute.internal   <none>           <none>
kube-system   kube-proxy-67890           1/1     Running   0          7m43s
 192.0.2.1   ip-192-0-2-1.region-code.compute.internal   <none>           <none>
```

Pour plus d'informations sur les éléments que vous voyez en sortie, consultez the section called "Accéder aux ressources du cluster".

Étape 3 : supprimer votre cluster et vos nœuds

Une fois que vous avez fini d'utiliser le cluster et les nœuds que vous avez créés pour ce tutoriel, vous devez effectuer un nettoyage en supprimant le cluster et les nœuds avec la commande suivante. Si vous souhaitez exécuter d'autres opérations avec ce cluster avant le nettoyage, consultez the section called "Étapes suivantes".

```
eksctl delete cluster --name my-cluster --region region-code
```

Étapes suivantes

Les rubriques suivantes de la documentation vous aideront à étendre les fonctionnalités de votre cluster.

- Déployez un exemple d'application sur votre cluster.
- Le principal IAM qui a créé le cluster est le seul principal qui peut effectuer des appels vers le serveur API Kubernetes avec `kubectl` ou le AWS Management Console. Si vous souhaitez que d'autres principaux IAM aient accès à votre cluster, vous devez les ajouter. Pour plus d'informations, consultez the section called "Accès à l'API Kubernetes" et the section called "Autorisations requises".
- Avant de déployer un cluster pour une utilisation en production, nous vous recommandons de vous familiariser avec tous les paramètres des clusters et des nœuds. Certains paramètres, tels que l'activation de l'accès SSH aux nœuds Amazon EC2, doivent être effectués lors de la création du cluster.
- Pour renforcer la sécurité de votre cluster, configurez le plugin Amazon VPC Container Networking Interface pour utiliser les rôles IAM pour les comptes de service.

Démarrez avec Amazon EKS AWS Management Console et la AWS CLI

> **Note**
>
> Cette rubrique traite du démarrage sans le mode automatique EKS. Il utilise des groupes de nœuds gérés pour déployer des nœuds.

> Le mode automatique EKS automatise les tâches courantes liées au calcul, au stockage et à la mise en réseau des clusters. Découvrez comment démarrer avec le mode automatique Amazon EKS. Le mode automatique EKS est la méthode préférée pour déployer des nœuds.

Ce guide vous aide à créer toutes les ressources nécessaires pour démarrer avec Amazon Elastic Kubernetes Service (Amazon EKS) à l'aide AWS Management Console de la CLI et de la CLI. AWS Dans ce guide, vous créez manuellement chaque ressource. À la fin de ce didacticiel, vous disposerez d'un cluster Amazon EKS en cours d'exécution sur lequel vous pouvez déployer des applications.

Les procédures de ce guide vous donnent une visibilité complète sur la façon dont chaque ressource est créée et dont les ressources interagissent les unes avec les autres. Si vous préférez créer automatiquement la plupart des ressources pour vous, utilisez la CLI `eksctl` pour créer votre cluster et vos nœuds. Pour de plus amples informations, veuillez consulter the section called “Créer un cluster (eksctl)”.

Conditions préalables

Avant de démarrer ce didacticiel, vous devez installer et configurer les outils et les ressources suivants dont vous avez besoin pour créer et gérer un cluster Amazon EKS.

- AWS CLI : outil de ligne de commande permettant de travailler avec AWS des services, notamment Amazon EKS. Pour plus d'informations, consultez la section Installation dans le guide de l'utilisateur de l'interface de ligne de AWS commande. Après avoir installé la AWS CLI, nous vous recommandons de la configurer également. Pour plus d'informations, consultez la section Configuration rapide avec aws configure dans le Guide de l'utilisateur de l'interface de ligne de AWS commande. Notez que la AWS CLI v2 est requise pour utiliser l'option update-kubeconfig présentée sur cette page.
- **`kubectl`** : outil de ligne de commande pour travailler avec des clusters Kubernetes. Pour de plus amples informations, veuillez consulter the section called “Configurer kubectl et eksctl”.
- Autorisations IAM requises : le principal de sécurité IAM que vous utilisez doit être autorisé à utiliser les rôles IAM Amazon EKS, les rôles liés à un service, AWS CloudFormation un VPC et les ressources associées. Pour plus d'informations, consultez les sections Actions et Utilisation des rôles liés aux services dans le Guide de l'utilisateur IAM. Vous devez effectuer toutes les étapes de ce guide avec le même utilisateur. Exécutez la commande suivante pour vérifier l'utilisateur actuel :

```
aws sts get-caller-identity
```

Nous vous recommandons de terminer les étapes de cette rubrique dans un shell Bash. Si vous n'utilisez pas de shell Bash, certaines commandes de script telles que les caractères de continuation de ligne et la façon dont les variables sont définies et utilisées nécessitent un ajustement pour votre shell. En outre, les règles de votre shell en matière de guillemets peuvent être différentes. Pour plus d'informations, consultez la section Utilisation de guillemets avec des chaînes dans la AWS CLI du Guide de l'utilisateur de l'interface de ligne de AWS commande.

Étape 1 : créer votre cluster Amazon EKS

> **Important**
>
> Pour démarrer le plus simplement et le plus rapidement possible, cette rubrique inclut les étapes pour créer un cluster avec des paramètres par défaut. Avant de créer un cluster pour une utilisation en production, nous vous recommandons de vous familiariser avec tous les paramètres et de déployer un cluster avec les paramètres qui répondent à vos besoins. Pour de plus amples informations, veuillez consulter the section called "Créer un cluster". Certains paramètres ne peuvent être activés que lors de la création de votre cluster.

1. Créez un Amazon VPC avec des sous-réseaux publics et privés qui répondent aux exigences Amazon EKS. Remplacez *region-code* par n'importe quelle AWS région prise en charge par Amazon EKS. Pour obtenir la liste des AWS régions, consultez la section Points de terminaison et quotas Amazon EKS dans le guide de référence AWS général. Vous pouvez remplacer *my-eks-vpc-stack* par n'importe quel nom que vous choisissez.

   ```
   aws cloudformation create-stack \
     --region region-code \
     --stack-name my-eks-vpc-stack \
     --template-url https://s3.us-west-2.amazonaws.com/amazon-eks/
   cloudformation/2020-10-29/amazon-eks-vpc-private-subnets.yaml
   ```

> **Tip**
>
> Pour obtenir la liste de toutes les ressources créées par la commande précédente, ouvrez la AWS CloudFormation console à l'adresse https://console.aws.amazon.com/cloudformation/. Choisissez la pile *my-eks-vpc-stack*, puis choisissez l'onglet Ressources.

2. Créez un rôle IAM de cluster et attachez-y la politique gérée par Amazon EKS IAM requise. Les clusters Kubernetes gérés par Amazon EKS appellent d'autres AWS services en votre nom afin de gérer les ressources que vous utilisez avec le service.

   a. Copiez le contenu suivant dans un fichier nommé *eks-cluster-role-trust-policy.json*.

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

   b. Créez le rôle.

   ```
   aws iam create-role \
     --role-name myAmazonEKSClusterRole \
     --assume-role-policy-document file://"eks-cluster-role-trust-policy.json"
   ```

   c. Attachez la politique IAM gérée par Amazon EKS au rôle.

   ```
   aws iam attach-role-policy \
     --policy-arn arn:aws: iam::aws:policy/AmazonEKSClusterPolicy \
     --role-name myAmazonEKSClusterRole
   ```

3. Ouvrez la console Amazon EKS à l'adresse https://console.aws.amazon.com/eks/home#/clusters.

Assurez-vous que la AWS région affichée dans le coin supérieur droit de votre console est la AWS région dans laquelle vous souhaitez créer votre cluster. Si ce n'est pas le cas, choisissez le menu déroulant à côté du nom de la AWS région et choisissez la AWS région que vous souhaitez utiliser.

4. Choisissez Créer un cluster. Si cette option ne s'affiche pas, sélectionnez d'abord Clusters dans le panneau de navigation gauche.
5. Sur la page Configure cluster (Configurer le cluster), procédez de la façon suivante :
   a. Sélectionnez Configuration personnalisée et désactivez Utiliser le mode automatique EKS. (Si vous préférez un cluster du mode automatique EKS, reportez-vous plutôt à the section called "Console de gestion".)
   b. Saisissez un Name (Nom) pour votre cluster, tel que *my-cluster*. Un nom ne peut contenir que des caractères alphanumériques (sensibles à la casse) et des traits d'union. Il doit commencer par un caractère alphanumérique et ne peut pas dépasser 100 caractères. Le nom doit être unique dans la AWS région et le AWS compte dans lesquels vous créez le cluster.
   c. Pour Rôle de service de cluster, choisissez*myAmazonEKSClusterRole*.
   d. Laissez les autres paramètres à leurs valeurs par défaut et choisissez Next (Suivant).
6. Sur la page Specify networking (Spécifier les réseaux), procédez comme suit :
   a. Sélectionnez l'ID du VPC que vous avez créé à l'étape précédente dans la liste déroulante des VPC. C'est quelque chose comme** | my-eks-vpc-stack-VPC*.
   b. Sélectionnez les sous-réseaux créés lors d'une étape précédente dans la liste déroulante Sous-réseaux. Les sous-réseaux seront quelque chose comme** | my-eks-vpc-stack-**.
   c. Sélectionnez le groupe de sécurité créé à l'étape précédente dans la liste déroulante des groupes de sécurité supplémentaires. C'est quelque chose comme** | my-eks-vpc-stack-ControlPlaneSecurityGroup-**.
   d. Laissez les autres paramètres à leurs valeurs par défaut et choisissez Next (Suivant).
7. Sur la page Configurer l'observabilité, cliquez sur Suivant.
8. Sur la page Sélectionner des modules complémentaires, choisissez Suivant.

   Pour plus d'informations sur les modules complémentaires, consultez the section called "Modules complémentaires Amazon EKS".
9. Sur la page Configurer les paramètres des modules complémentaires, choisissez Suivant.
10. Sur la page Review and create (Vérifier et créer), choisissez Create (Créer).

À droite du nom du cluster, le statut du cluster est Création pendant plusieurs minutes jusqu'à ce que le processus de provisionnement du cluster soit terminé. Ne passez à l'étape suivante que lorsque le cluster est Actif.

> **Note**
>
> Il est possible que vous receviez un message d'erreur indiquant que l'une des zones de disponibilité de votre demande ne dispose pas d'une capacité suffisante pour créer un cluster Amazon EKS. Si cela se produit, la sortie de l'erreur contient les zones de disponibilité qui peuvent prendre en charge un nouveau cluster. Essayez à nouveau de créer votre cluster avec au moins deux sous-réseaux situés dans les zones de disponibilité prises en charge pour votre compte. Pour de plus amples informations, veuillez consulter the section called "Capacité insuffisante".

Étape 2 : configurer votre ordinateur de façon à ce qu'il communique avec votre cluster

Dans cette section, vous créez un fichier `kubeconfig` pour votre cluster. Les paramètres de ce fichier activent l'option `kubectl` CLI pour communiquer avec votre cluster.

Avant de continuer, assurez-vous que la création de votre cluster s'est bien terminée à l'étape 1.

1. Créez ou mettez à jour un fichier `kubeconfig` pour votre cluster. Remplacez *`region-code`* par la AWS région dans laquelle vous avez créé votre cluster. Remplacez *`my-cluster`* par le nom de votre cluster.

   ```
   aws eks update-kubeconfig --region region-code --name my-cluster
   ```

   Par défaut, le fichier `config` est créé dans `~/.kube` ou la configuration du nouveau cluster est ajoutée à un fichier `config` existant dans `~/.kube`.

2. Testez votre configuration.

   ```
   kubectl get svc
   ```

> **Note**
>
> Si vous recevez d'autres erreurs concernant les types d'autorisations ou de ressources, consultez the section called "Accès non autorisé ou refusé (`kubectl`)" dans la rubrique relative à la résolution des problèmes.

L'exemple qui suit illustre un résultat.

```
NAME             TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
svc/kubernetes   ClusterIP   10.100.0.1   <none>        443/TCP   1m
```

Étape 3 : créer des nœuds

> **Important**
>
> Pour démarrer aussi simplement et rapidement que possible, cette rubrique comprend les étapes à suivre pour créer des nœuds avec des paramètres par défaut pour la plupart. Avant de créer des nœuds pour une utilisation en production, nous vous recommandons de vous familiariser avec tous les paramètres et de déployer des nœuds avec les paramètres qui répondent à vos besoins. Pour de plus amples informations, veuillez consulter Gestion du Calcul. Certains paramètres ne peuvent être activés que lors de la création de vos nœuds.

Cette procédure configure votre cluster pour utiliser des groupes de nœuds gérés afin de créer des nœuds, en spécifiant les sous-réseaux et le rôle IAM des nœuds que vous avez créés lors des étapes précédentes. Il vous permet d'exécuter des applications Amazon Linux sur des EC2 instances Amazon.

Pour en savoir plus sur les différentes manières de configurer les nœuds dans EKS, consultez Gestion du Calcul. Une fois votre cluster déployé, vous pouvez ajouter d'autres types de nœuds. Bien que cela ne soit pas inclus dans ce guide, vous pouvez également ajouter les nœuds autogérés par Windows et Bottlerocket à votre cluster.

Pour créer votre groupe de nœuds gérés par EC2 Linux

1. Créez un rôle IAM de nœud et attachez-y la politique gérée par Amazon EKS IAM requise. Le `kubelet` démon du nœud Amazon EKS effectue des appels AWS APIs en votre nom. Les nœuds reçoivent l'autorisation pour ces appels d'API via un profil d'instance IAM et les politiques associées.

   a. Copiez le contenu suivant dans un fichier nommé `node-role-trust-policy.json`.

   ```
   {
     "Version":"2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
         "Principal": {
           "Service": "ec2.amazonaws.com"
         },
         "Action": "sts:AssumeRole"
       }
     ]
   }
   ```

   b. Créez le rôle IAM du nœud.

   ```
   aws iam create-role \
     --role-name myAmazonEKSNodeRole \
     --assume-role-policy-document file://"node-role-trust-policy.json"
   ```

   c. Attachez les politiques IAM gérées requises au rôle.

   ```
   aws iam attach-role-policy \
     --policy-arn arn:aws: iam::aws:policy/AmazonEKSWorkerNodePolicy \
     --role-name myAmazonEKSNodeRole
   aws iam attach-role-policy \
     --policy-arn arn:aws: iam::aws:policy/AmazonEC2ContainerRegistryReadOnly \
     --role-name myAmazonEKSNodeRole
   aws iam attach-role-policy \
     --policy-arn arn:aws: iam::aws:policy/AmazonEKS_CNI_Policy \
     --role-name myAmazonEKSNodeRole
   ```

   d. Ouvrez la console Amazon EKS à l'adresse https://console.aws.amazon.com/eks/home#/clusters.

   e. Choisissez le nom du cluster que vous avez créé à l'étape 1 : Créez votre cluster Amazon EKS, tel que *`my-cluster`*.

f. Sur la ***my-cluster***page, effectuez les opérations suivantes :

g. Choisissez l'onglet Compute (Calcul).

h. Choisissez Add Node Group (Ajouter un groupe de nœuds).

2. Sur la page Configure Node Group (Configurer le groupe de nœuds), procédez comme suit :

   a. Pour Name (Nom), saisissez un nom unique pour votre groupe de nœuds gérés, tel que *my-nodegroup*. Le nom du groupe de nœuds ne peut pas dépasser 63 caractères. Il doit commencer par une lettre ou un chiffre, mais peut également inclure des tirets et des traits de soulignement pour les autres caractères.

   b. Pour le nom du rôle Node IAM, choisissez *myAmazonEKSNodeRole* le rôle que vous avez créé à l'étape précédente. Nous recommandons que chaque groupe de nœuds utilise son propre rôle IAM unique.

   c. Choisissez Suivant.

3. Sur la page Set compute and scaling configuration (Définir la configuration de calcul et de mise à l'échelle), acceptez les valeurs par défaut et choisissez Next (Suivant).

4. Sur la page Specify networking (Spécifier les réseaux), acceptez les valeurs par défaut et choisissez Next (Suivant).

5. Sur la page Review and create (Vérifier et créer), vérifiez la configuration de votre groupe de nœuds gérés et choisissez Create (Créer).

6. Après quelques minutes, le Status (Statut) dans la Node Group configuration (Configuration du groupe de nœuds) passera de Creating (En cours de création) à Active (Actif). Ne passez à l'étape suivante que lorsque le cluster est Actif.

Étape 4 : afficher les ressources

Vous pouvez afficher vos nœuds et vos applications Kubernetes.

1. Dans le volet de navigation de gauche, choisissez Clusters. Dans la liste des clusters, choisissez le nom du cluster que vous avez créé, par exemple*my-cluster*.

2. Sur la ***my-cluster***page, sélectionnez les options suivantes :

   a. Onglet Compute (Calcul) – La liste Nodes (Nœuds) qui ont été déployés pour le cluster s'affiche. Vous pouvez choisir le nom d'un nœud pour voir plus d'informations à son sujet.

   b. Onglet Ressources : toutes les ressources Kubernetes qui sont déployées par défaut sur un cluster Amazon EKS s'affichent. Sélectionnez un type de ressources dans la console pour en savoir plus.

Étape 5 : Suppression des ressources

Une fois que vous avez terminé avec le cluster et les nœuds que vous avez créés pour ce tutoriel, vous devez supprimer les ressources que vous avez créées. Si vous souhaitez exécuter d'autres opérations avec ce cluster de supprimer les ressources, consultez the section called “Étapes suivantes”.

1. Supprimez tous les profils de groupes de nœuds que vous avez créés.
   a. Ouvrez la console Amazon EKS à l'adresse https://console.aws.amazon.com/eks/home#/clusters.
   b. Dans le volet de navigation de gauche, choisissez Clusters. Dans la liste des clusters, choisissez *my-cluster*.
   c. Choisissez l'onglet Calcul.
   d. Si vous avez créé un groupe de *my-nodegroup* nœuds, choisissez-le, puis choisissez Supprimer. Entrez *my-nodegroup*, puis choisissez Delete (Supprimer).
   e. Ne continuez pas tant que les profils du groupe de nœuds n'ont pas été supprimés.
2. Supprimez le cluster.
   a. Dans le volet de navigation de gauche, choisissez Clusters. Dans la liste des clusters, choisissez *my-cluster*.
   b. Choisissez Supprimer le cluster.
   c. Saisissez *my-cluster* (supprimer), puis sélectionnez Delete (Supprimer). Ne continuez pas tant que le cluster n'est pas supprimé.
3. Supprimez la AWS CloudFormation pile VPC que vous avez créée.
   a. Ouvrez la CloudFormation console à l'adresse https://console.aws.amazon.com/cloudformation/.
   b. Choisissez la pile *my-eks-vpc-stack*, puis choisissez Supprimer.
   c. Dans la boîte *my-eks-vpc-stack* de dialogue de confirmation de suppression, choisissez Supprimer la pile.
4. Supprimez le rôle IAM que vous avez créé.
   a. Ouvrez la console IAM à https://console.aws.amazon.com/iam/ l'adresse.
   b. Dans le volet de navigation de gauche, choisissez Rôles.

c. Sélectionnez chaque rôle que vous avez créé dans la liste (***myAmazonEKSClusterRole***, ainsi que*myAmazonEKSNodeRole*). Choisissez Delete (Supprimer), saisissez le texte de confirmation demandé, puis choisissez Delete (Supprimer).

Étapes suivantes

Les rubriques suivantes de la documentation vous aideront à étendre les fonctionnalités de votre cluster.

- Le principal IAM qui a créé le cluster est le seul principal qui peut effectuer des appels vers le serveur API Kubernetes avec `kubectl` ou le AWS Management Console. Si vous souhaitez que d'autres principaux IAM aient accès à votre cluster, vous devez les ajouter. Pour plus d'informations, consultez the section called "Accès à l'API Kubernetes" et the section called "Autorisations requises".
- Déployez un exemple d'application sur votre cluster.
- Avant de déployer un cluster pour une utilisation en production, nous vous recommandons de vous familiariser avec tous les paramètres des clusters et des nœuds. Certains paramètres (tels que l'activation de l'accès SSH aux EC2 nœuds Amazon) doivent être définis lors de la création du cluster.
- Pour renforcer la sécurité de votre cluster, configurez le plugin Amazon VPC Container Networking Interface pour utiliser les rôles IAM pour les comptes de service.

Automatisation de l'infrastructure du cluster avec le mode automatique EKS

> **Tip**
>
> Inscrivez-vous aux prochains ateliers Amazon EKS Auto Mode.

Le mode automatique d'EKS étend la AWS gestion des clusters Kubernetes au-delà du cluster lui-même, afin de AWS permettre également de configurer et de gérer l'infrastructure qui permet le bon fonctionnement de vos charges de travail. Vous pouvez déléguer les décisions clés en matière d'infrastructure et tirer parti de l'expertise de AWS for day-to-day operations. L'infrastructure de cluster gérée par AWS inclut de nombreuses fonctionnalités Kubernetes en tant que composants principaux, par opposition aux modules complémentaires, tels que le dimensionnement automatique du calcul, la mise en réseau des pods et des services, l'équilibrage de charge des applications, le DNS du cluster, le stockage par blocs et le support du GPU.

Pour commencer, vous pouvez déployer un nouveau cluster du mode automatique EKS ou activer le mode automatique EKS sur un cluster existant. Vous pouvez déployer, mettre à niveau ou modifier vos clusters en mode automatique EKS à l'aide d'eksctl, de la AWS AWS Management Console CLI, d'EKS APIs ou de vos outils préférés infrastructure-as-code.

Avec le mode automatique EKS, vous pouvez continuer d'utiliser vos outils compatibles avec Kubernetes préférés. Le mode automatique d'EKS s'intègre à AWS des services tels qu'Amazon EC2, Amazon EBS et ELB, en tirant parti des ressources AWS cloud conformes aux meilleures pratiques. Ces ressources sont mises à l'échelle automatiquement, optimisées en termes de coûts et mises à jour régulièrement afin de réduire les coûts d'exploitation et les charges opérationnelles.

Caractéristiques

Le mode automatique EKS offre les fonctionnalités de haut niveau suivantes :

Simplification de la gestion des clusters Kubernetes : le mode automatique EKS permet de gérer EKS de manière simplifiée, grâce à des clusters prêts pour la production, avec un minimum de gestion opérationnelle. Avec le mode automatique EKS, vous pouvez exécuter des charges de travail dynamiques et exigeantes en toute confiance, sans avoir besoin d'une expertise EKS approfondie.

Disponibilité des applications : le mode automatique EKS ajoute ou supprime dynamiquement des nœuds dans votre cluster en fonction des besoins de vos applications Kubernetes. Cela réduit la planification manuelle de la capacité et contribue à assurer la disponibilité des applications.

Efficacité : le mode automatique EKS est conçu pour optimiser les coûts de calcul tout en respectant la flexibilité définie par vos exigences NodePool et celles de votre charge de travail. Il termine les instances inutilisées et consolide les charges de travail sur d'autres nœuds pour améliorer l'efficacité et réduire les coûts.

Sécurité : le mode automatique EKS utilise AMIs des éléments considérés comme immuables pour vos nœuds. Ils AMIs renforcent le verrouillage des logiciels, activent les contrôles d'accès SELinux obligatoires et fournissent des systèmes de fichiers racine en lecture seule. De plus, les nœuds lancés par le mode automatique EKS ont une durée de vie maximale de 21 jours (réductible), après laquelle ils sont automatiquement remplacés par de nouveaux nœuds. Cette approche améliore votre niveau de sécurité en renouvelant régulièrement les nœuds, conformément aux bonnes pratiques déjà adoptées par de nombreux clients.

Mises à niveau automatisées : le mode automatique d'EKS maintient votre cluster Kubernetes, vos nœuds et les composants associés à jour avec les derniers correctifs, tout en respectant les budgets d'interruption des modules (PDBs) et les budgets d'interruption () que vous NodePool avez configurés. NDBs Jusqu'à la durée de vie maximale de 21 jours, une intervention peut être nécessaire si le blocage PDBs ou d'autres configurations empêchent les mises à jour.

Composants gérés : le mode automatique d'EKS inclut Kubernetes et les fonctionnalités AWS cloud en tant que composants de base qui devraient autrement être gérés en tant que modules complémentaires. Cela inclut la prise en charge intégrée des attributions d'adresses IP des pods, des politiques réseau des pods, des services DNS locaux, des plug-ins GPU, des vérificateurs d'état et du stockage EBS CSI.

Personnalisable NodePools et NodeClasses : si votre charge de travail nécessite des modifications des configurations de stockage, de calcul ou de mise en réseau, vous pouvez créer des configurations personnalisées NodePools à NodeClasses l'aide du mode automatique EKS. Bien que vous ne deviez pas modifier NodePools les configurations par défaut NodeClasses, vous pouvez ajouter de nouvelles configurations personnalisées NodePools ou en NodeClasses complément des configurations par défaut pour répondre à vos besoins spécifiques.

Composants automatisés

Le mode automatique EKS simplifie l'exploitation de vos clusters Amazon EKS en automatisant les composants d'infrastructure essentiels. Activer le mode automatique EKS réduit encore davantage les tâches nécessaires à la gestion de vos clusters EKS.

Voici une liste des composants du plan de données automatisés :

- Calcul : pour de nombreuses charges de travail, avec le mode automatique EKS, vous pouvez oublier la plupart des aspects liés au calcul dans vos clusters EKS. Il s'agit des licences suivantes :
  - Nœuds : les nœuds du mode automatique EKS sont conçus pour être gérés comme des appareils. Le mode automatique EKS effectue les opérations suivantes :
    - Choisit une AMI appropriée, configurée avec les services nécessaires pour exécuter vos charges de travail sans intervention.
    - Verrouille l'accès aux fichiers de l'AMI à l'aide du mode SELinux coercitif et d'un système de fichiers racine en lecture seule.
    - Empêche l'accès direct aux nœuds en interdisant l'accès SSH ou SSM.
    - Inclut la prise en charge du GPU, avec des pilotes de noyau et des plug-ins distincts pour NVIDIA et Neuron GPUs, permettant des charges de travail de haute performance.
    - Gère automatiquement les avis d'interruption de l'instance Spot EC2 et les événements liés à l'état de l'instance EC2
  - Autoscaling : en s'appuyant sur l'autoscaling Karpenter, le mode automatique EKS surveille les pods non planifiés et permet le déploiement automatique de nouveaux nœuds pour les exécuter. Lorsque les charges de travail se terminent, le mode automatique EKS interrompt et termine dynamiquement les nœuds inutilisés afin d'optimiser l'utilisation des ressources.
  - Mises à niveau : en prenant le contrôle de gestion des nœuds, le mode automatique EKS facilite l'application des correctifs de sécurité, des mises à niveau du système d'exploitation et des composants associés. Ces mises à niveau sont conçues pour interrompre le moins possible vos charges de travail. Le mode automatique EKS impose une durée de vie maximale des nœuds de 21 jours pour garantir que le up-to-date logiciel et. APIs
- Équilibrage de charge : le mode automatique EKS simplifie l'équilibrage de charge en s'intégrant au service Elastic Load Balancing d'Amazon, ce qui permet d'automatiser le provisionnement et la configuration des équilibreurs de charge pour les ressources Service et Ingress de Kubernetes. Il prend en charge des fonctionnalités avancées pour les équilibreurs de charges Application Load Balancer et les équilibreurs de charge Network Load Balancer, en gère tout le cycle de

vie et les met à l'échelle afin de répondre aux besoins du cluster. Cette intégration fournit une solution d'équilibrage de charge prête à être utilisée en production, conforme aux AWS meilleures pratiques, vous permettant de vous concentrer sur les applications plutôt que sur la gestion de l'infrastructure.

- Stockage : le mode automatique EKS configure le stockage éphémère automatiquement, notamment les types de volumes, les tailles et politiques de chiffrement, et les politiques de suppression lors de la terminaison des nœuds.
- Mise en réseau : le mode automatique EKS automatise les tâches réseau essentielles pour la connectivité des pods et des services. Cela inclut le IPv6 support IPv4/et l'utilisation de blocs CIDR secondaires pour étendre les espaces d'adresses IP.
- Gestion des identités et des accès : il n'est pas nécessaire d'installer l'agent d'identité du pod EKS sur les clusters du mode automatique EKS.

Pour plus d'informations sur ces composants, consultez the section called "Comment ça marche".

Configuration

Même si le mode automatique EKS gère efficacement la plupart des services du plan de données sans intervention, il peut arriver que vous souhaitiez adapter le comportement de certains services. Vous pouvez modifier la configuration de vos clusters du mode automatique EKS des manières suivantes :

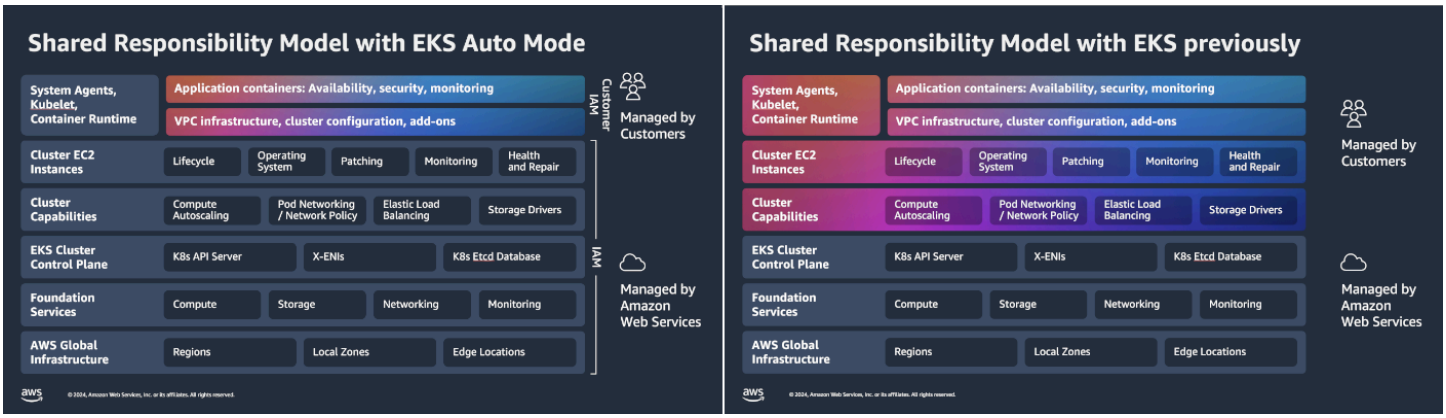
- Kubernetes DaemonSets : plutôt que de modifier les services installés sur vos nœuds, vous pouvez utiliser des daemonsets Kubernetes. Les DaemonSets sont gérés par Kubernetes, mais s'exécutent sur tous les nœuds du cluster. Cela vous permet d'ajouter des services spécialisés pour surveiller ou contrôler l'état de vos nœuds.
- NodePools Personnalisés et NodeClasses : Par défaut NodePools et NodeClasses configurés par le mode automatique d'EKS, vous ne devez pas les modifier. Pour personnaliser le comportement des nœuds, vous pouvez créer des cas supplémentaires NodePools ou NodeClasses pour des cas d'utilisation tels que :
  - Sélection de types d'instances spécifiques (par exemple, processeurs accélérés ou instances Spot EC2).
  - Isolation des charges de travail à des fins de sécurité ou de suivi des coûts.
  - Configuration des paramètres de stockage éphémères tels que les IOPS, la taille et le débit.

- Équilibrage de charge : certains services, comme l'équilibrage de charge, exécutés en tant qu'objets Kubernetes par le mode automatique EKS, peuvent être configurés directement dans vos clusters du mode automatique EKS.

Pour plus d'informations sur les options de configuration du mode automatique EKS, consultez the section called "Configurer".

Modèle de responsabilité partagée

Le modèle de responsabilité AWS partagée définit les responsabilités en matière de sécurité et de conformité entre AWS et les clients. Les images et le texte ci-dessous comparent et contrastent les différences entre le client et les AWS responsabilités entre le mode automatique d'EKS et le mode standard d'EKS.



Le mode automatique EKS transfère une grande partie des responsabilités partagées liées à l'infrastructure Kubernetes du client vers AWS. Avec le mode automatique d'EKS, AWS vous assumez davantage la responsabilité de la sécurité du cloud, qui relevait autrefois de la responsabilité du client et qui est désormais partagée. Les clients peuvent désormais se concentrer davantage sur leurs applications tout en AWS gérant l'infrastructure sous-jacente.

Responsabilité du client

Avec le mode automatique EKS, les clients demeurent responsables des conteneurs des applications, y compris la disponibilité, la sécurité et la surveillance. Ils gardent également le contrôle de l'infrastructure VPC et de la configuration du cluster EKS. Ce modèle permet aux clients de se concentrer sur les préoccupations spécifiques aux applications tout en déléguant la gestion de l'infrastructure du cluster à. AWS Des fonctionnalités optionnelles par nœud peuvent être incluses dans les clusters par le biais de AWS modules complémentaires.

Responsabilité d’ AWS

Avec le mode automatique d'EKS, ses responsabilités AWS s'étendent pour inclure la gestion de plusieurs composants critiques supplémentaires par rapport à ceux déjà gérés dans les clusters EKS n'utilisant pas le mode automatique. Plus précisément, le mode automatique EKS prend en charge la configuration, la gestion, la sécurité et la mise à l’échelle des instances EC2 lancées pour le cluster ainsi que des capacités du cluster liées à l’équilibrage de charge, à la gestion des adresses IP, aux politiques réseau et au stockage par blocs. Les composants suivants sont gérés par AWS le mode automatique EKS :

- Instances EC2 lancées en mode automatique : AWS gère le cycle de vie complet des nœuds en tirant parti des instances gérées par Amazon EC2. Les instances gérées EC2 sont responsables de la configuration du système d’exploitation, de l’application des correctifs, de la surveillance et de la maintenance du système d’exploitation. Dans ce modèle, l’instance elle-même et le système d’exploitation invité qui y est exécuté relèvent de la responsabilité de AWS. Les nœuds utilisent des variantes de Bottlerocket AMIs optimisées pour exécuter des conteneurs. Les Bottlerocket AMIs disposent d'un logiciel verrouillé, de systèmes de fichiers racines immuables et d'un accès réseau sécurisé (pour empêcher les communications directes via SSH ou SSM).
- Capacités du cluster : AWS gère le dimensionnement automatique du calcul, la mise en réseau des pods avec l'application des politiques réseau, l'intégration d'Elastic Load Balancing et la configuration des pilotes de stockage.
- Plan de contrôle du cluster : AWS continue de gérer le serveur d'API Kubernetes, les comptes multiples et la base de données etcd ENIs, comme avec le système EKS standard.
- Services de base et infrastructure globale : assume AWS la responsabilité des services sous-jacents de calcul, de stockage, de mise en réseau et de surveillance, ainsi que de l'infrastructure globale des régions, des zones locales et des emplacements périphériques.

Création d’un cluster avec le mode automatique Amazon EKS

Ce chapitre explique comment créer un cluster Amazon EKS avec le mode automatique activé, en utilisant différents outils et interfaces. Le mode automatique EKS simplifie la création de cluster en configurant et en gérant automatiquement l’infrastructure de cluster computing, de mise en réseau et de stockage du cluster. Vous apprendrez à créer un cluster du mode automatique à l’aide de la CLI AWS, de la AWS Management Console ou de l’outil de ligne de commande eksctl.

> **Note**
>
> Le mode automatique EKS nécessite la version 1.29 ou une version ultérieure de Kubernetes.

Choisissez votre outil préféré en fonction de vos besoins : la AWS Management Console fournit une interface visuelle idéale pour découvrir les fonctionnalités du mode automatique EKS et créer des clusters individuels. La CLI AWS convient parfaitement aux tâches de script et d'automatisation, en particulier lorsqu'il s'agit d'intégrer la création de clusters dans des flux de travail existants ou des pipelines CI/CD. La CLI offre une expérience native Kubernetes et est recommandée pour les utilisateurs familiers des outils Kubernetes, souhaitant effectuer des opérations simplifiées en ligne de commande avec des paramètres par défaut pertinents.

Avant de commencer, assurez-vous d'avoir installé et configuré les prérequis nécessaires, notamment les autorisations IAM appropriées pour créer des clusters EKS. Pour savoir comment installer des outils CLI tels que `kubectl`, `aws` et `eksctl`, consultez Configuration.

Vous pouvez utiliser la CLI AWS, la AWS Management Console, ou la CLI eksctl pour créer un cluster avec le mode automatique Amazon EKS.

Rubriques

- Création d'un cluster du mode automatique EKS à l'aide de la CLI eksctl
- Création d'un cluster en mode automatique EKS à l'aide de la AWS CLI
- Création d'un cluster du mode automatique EKS à l'aide de la AWS Management Console

Création d'un cluster du mode automatique EKS à l'aide de la CLI eksctl

Cette rubrique vous explique comment créer un cluster du mode automatique Amazon EKS à l'aide de l'interface de ligne de commande (CLI) eksctl. Vous pouvez créer un cluster du mode automatique en exécutant une seule commande CLI ou en appliquant un fichier de configuration YAML. Les deux méthodes fournissent les mêmes fonctionnalités, l'approche YAML offrant un contrôle plus précis des paramètres du cluster.

La CLI eksctl simplifie le processus de création et de gestion des clusters du mode automatique EKS en gérant la création et la configuration des ressources AWS sous-jacentes. Avant de continuer, assurez-vous que les informations d'identification AWS et les autorisations nécessaires sont

configurées sur votre ordinateur local. Ce guide suppose que vous connaissez les concepts de base d'Amazon EKS et que vous avez déjà installé les outils CLI requis.

> **Note**
>
> Vous devez installer la version `0.195.0` ou ultérieure d'eksctl. Pour plus d'informations, consultez versions eksctl sur GitHub.

Création d'un cluster du mode automatique EKS à l'aide de la commande CLI

Vous devez avoir installé les outils `aws` et `eksctl`. Vous devez être connecté à la CLI AWS avec des autorisations suffisantes pour gérer les ressources AWS, y compris : instances EC2, réseau EC2, clusters EKS et rôles IAM. Pour de plus amples informations, consultez Configuration.

Exécutez la commande suivante pour créer un nouveau cluster du mode automatique EKS avec

```
eksctl create cluster --name=<cluster-name> --enable-auto-mode
```

Création d'un cluster du mode automatique Amazon EKS à l'aide d'un fichier YAML

Vous devez avoir installé les outils `aws` et `eksctl`. Vous devez être connecté à la CLI AWS avec des autorisations suffisantes pour gérer les ressources AWS, y compris : instances EC2, réseau EC2, clusters EKS et rôles IAM. Pour de plus amples informations, consultez Configuration.

Passez en revue les options de configuration du mode automatique EKS dans l'exemple de ressource ClusterConfig ci-dessous. Pour la spécification complète de ClusterConfig, consultez la documentation eksctl.

AWS recommande d'activer le mode automatique EKS. Si c'est la première fois que vous créez un cluster du mode automatique EKS, laissez la valeur `nodeRoleARN` non spécifiée pour que le système crée un rôle IAM de nœud pour le mode automatique EKS. Si vous disposez déjà d'un rôle IAM de nœud dans votre compte AWS, AWS recommande de le réutiliser.

AWS recommande de ne pas spécifier de valeur pour `nodePools`. Le mode automatique EKS créera des groupes de nœuds par défaut. Vous pouvez utiliser l'API Kubernetes pour créer des groupes de nœuds supplémentaires.

```
# cluster.yaml
apiVersion: eksctl.io/v1alpha5
```

```
kind: ClusterConfig

metadata:
  name: <cluster-name>
  region: <aws-region>

iam:
  # ARN of the Cluster IAM Role
  # optional, eksctl creates a new role if not supplied
  # suggested to use one Cluster IAM Role per account
  serviceRoleARN: <arn-cluster-iam-role>

autoModeConfig:
  # defaults to false
  enabled: boolean
  # optional, defaults to [general-purpose, system].
  # suggested to leave unspecified
  # To disable creation of nodePools, set it to the empty array ([]).
  nodePools: []string
  # optional, eksctl creates a new role if this is not supplied
  # and nodePools are present.
  nodeRoleARN: string
```

Enregistrez le fichier `ClusterConfig` sous `cluster.yaml`, puis utilisez la commande suivante pour créer le cluster :

```
eksctl create cluster -f cluster.yaml
```

Création d'un cluster en mode automatique EKS à l'aide de la AWS CLI

Les clusters du mode automatique EKS automatisent les tâches courantes de gestion du cluster pour le calcul, le stockage et la mise en réseau. Par exemple, les clusters en mode automatique EKS détectent automatiquement lorsque des nœuds supplémentaires sont nécessaires et fournissent de nouvelles EC2 instances pour répondre aux demandes de charge de travail.

Cette rubrique explique comment créer un nouveau cluster en mode automatique EKS à l'aide de la AWS CLI et éventuellement déployer un exemple de charge de travail.

Conditions préalables

- La dernière version de l'interface de ligne de AWS commande (AWS CLI) installée et configurée sur votre appareil. Pour vérifier votre version actuelle, utilisez `aws --version`. Pour installer la

dernière version, consultez la section Installation et configuration rapide avec aws configure dans le Guide de l'utilisateur de l'interface de ligne de AWS commande.

  - Connectez-vous à la CLI avec des autorisations IAM suffisantes pour créer des AWS ressources, notamment des politiques IAM, des rôles IAM et des clusters EKS.
- L'outil de ligne de commande kubectl installé sur votre appareil. AWS suggère que vous utilisiez la même version de kubectl que la version Kubernetes de votre cluster EKS. Pour installer ou mettre à niveau kubectl, consultez the section called “Configurer kubectl et eksctl”.

Spécification des sous-réseaux VPC

Le mode automatique Amazon EKS déploie des nœuds sur des sous-réseaux VPC. Lorsque vous créez un cluster EKS, vous devez spécifier les sous-réseaux VPC dans lesquels les nœuds seront déployés. Vous pouvez utiliser les sous-réseaux VPC par défaut de votre AWS compte ou créer un VPC dédié aux charges de travail critiques.

- AWS suggère de créer un VPC dédié pour votre cluster. Découvrez comment the section called “Création d’un VPC”.
- La console EKS aide à créer un nouveau VPC. Découvrez comment the section called “Console de gestion”.
- Vous pouvez également utiliser le VPC par défaut de votre AWS compte. Suivez les instructions ci-dessous pour trouver le sous-réseau IDs.

Pour trouver le sous-réseau IDs de votre VPC par défaut

À l'aide de la AWS CLI :

1. Exécutez la commande suivante pour répertorier le VPC par défaut et ses sous-réseaux :

```
aws ec2 describe-subnets --filters "Name=vpc-id,Values=$(aws ec2 describe-vpcs
 --query 'Vpcs[?IsDefault==`true`].VpcId' --output text)" --query 'Subnets[*].
{ID:SubnetId,AZ:AvailabilityZone}' --output table
```

2. Enregistrez la sortie et notez le sous-réseau IDs.

   Exemple de sortie :

```
----------------------------------------
|             DescribeSubnets          |
```

```
---------------------------------------
|   SubnetId        |   AvailabilityZone  |
|-------------------|---------------------|
|   subnet-012345678 |   us-west-2a        |
|   subnet-234567890 |   us-west-2b        |
|   subnet-345678901 |   us-west-2c        |
---------------------------------------
```

Rôles IAM des clusters du mode automatique EKS

Rôle IAM du cluster

Le mode automatique d'EKS nécessite un rôle IAM de cluster pour effectuer des actions sur votre AWS compte, telles que le provisionnement de nouvelles EC2 instances. Vous devez créer ce rôle pour accorder à EKS les autorisations nécessaires. AWS recommande d'associer les politiques AWS gérées suivantes au rôle IAM du cluster :

- EKSComputePolitique d'Amazon
- AmazonEKSBlockStoragePolicy
- AmazonEKSLoadBalancingPolicy
- EKSNetworkingPolitique d'Amazon
- EKSClusterPolitique d'Amazon

Rôle IAM de nœud

Lorsque vous créez un cluster du mode automatique EKS, vous spécifiez un rôle IAM de nœud. Lorsque le mode automatique d'EKS crée des nœuds pour traiter les charges de travail en attente, le rôle Node IAM est attribué à chaque nouveau nœud d' EC2 instance. Ce rôle permet au nœud de communiquer avec EKS, mais il n'est généralement pas utilisé par les charges de travail qui s'exécutent sur ce nœud.

Si vous souhaitez accorder des autorisations à des charges de travail exécutées sur un nœud, utilisez l'identité du pod EKS. Pour de plus amples informations, veuillez consulter the section called "Pod Identity".

Vous devez créer ce rôle et y associer la politique AWS gérée suivante :

- AmazonEKSWorkerNodeMinimalPolicy

- AmazonEC2ContainerRegistryPullOnly

Rôle lié à un service

Le mode automatique EKS nécessite également un rôle lié au service, qui est automatiquement créé et configuré par AWS. Pour plus d'informations, consultez AWSServiceRoleForAmazonEKS.

Création d'un rôle IAM de cluster du mode automatique Amazon EKS

Étape 1 : créer la politique d'approbation

Créez une politique d'approbation qui permet au service Amazon EKS d'assumer le rôle. Enregistrez la politique sous `trust-policy.json` :

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

Étape 2 : créer le rôle IAM

Utilisez la politique d'approbation pour créer le rôle IAM du cluster :

```
aws iam create-role \
    --role-name AmazonEKSAutoClusterRole \
    --assume-role-policy-document file://trust-policy.json
```

Étape 3 : noter l'ARN du rôle

Extrayez et enregistrez l'ARN du nouveau rôle pour les étapes ultérieures :

```
aws iam get-role --role-name AmazonEKSAutoClusterRole --query "Role.Arn" --output text
```

Étape 4 : attacher les politiques requises

Associez les politiques AWS gérées suivantes au rôle IAM du cluster pour accorder les autorisations nécessaires :

EKSClusterPolitique d'Amazon :

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSClusterPolicy
```

EKSComputePolitique d'Amazon :

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSComputePolicy
```

Amazon EKSBlock StoragePolicy :

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSBlockStoragePolicy
```

Amazon EKSLoad BalancingPolicy :

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSLoadBalancingPolicy
```

EKSNetworkingPolitique d'Amazon :

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSNetworkingPolicy
```

Création d'un rôle IAM de nœud du mode automatique EKS

Étape 1 : créer la politique d'approbation

Créez une politique d'approbation qui permet au service Amazon EKS d'assumer le rôle. Enregistrez la politique sous `node-trust-policy.json` :

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Étape 2 : créer le rôle IAM du nœud

Utilisez le fichier node-trust-policy.json de l'étape précédente pour définir les entités qui peuvent assumer le rôle. Exécutez la commande suivante pour créer le rôle IAM du nœud :

```
aws iam create-role \
    --role-name AmazonEKSAutoNodeRole \
    --assume-role-policy-document file://node-trust-policy.json
```

Étape 3 : noter l'ARN du rôle

Après la création du rôle, extrayez et enregistrez l'ARN du rôle IAM du nœud. Cet ARN sera nécessaire dans les étapes suivantes. Utilisez la commande suivante pour extraire l'ARN :

```
aws iam get-role --role-name AmazonEKSAutoNodeRole --query "Role.Arn" --output text
```

Étape 4 : attacher les politiques requises

Associez les politiques AWS gérées suivantes au rôle Node IAM pour fournir les autorisations nécessaires :

Amazon EKSWorker NodeMinimalPolicy :

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoNodeRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSWorkerNodeMinimalPolicy
```

Amazon EC2 ContainerRegistryPullOnly :

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoNodeRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEC2ContainerRegistryPullOnly
```

Création d'un cluster du mode automatique EKS

Présentation de

Pour créer un cluster en mode automatique EKS à l'aide de la AWS CLI, vous aurez besoin des paramètres suivants :

- `cluster-name` : nom du cluster.
- `k8s-version` : version Kubernetes (p. ex. : 1.31).
- `subnet-ids`: sous-réseau IDs identifié lors des étapes précédentes.
- `cluster-role-arn` : ARN du rôle IAM du cluster.
- `node-role-arn` : ARN du rôle IAM de nœud.

Configurations de clusters par défaut

Vérifiez ces valeurs et fonctionnalités par défaut avant de créer le cluster :

- `nodePools` : le mode automatique EKS inclut des groupes de nœuds généraux et des groupes de nœuds système par défaut. Pour en savoir plus, consultez Groupes de nœuds.

Remarque : les groupes de nœuds du mode automatique EKS sont différents des groupes de nœuds gérés par Amazon EKS, mais ils peuvent coexister dans le même cluster.

- `computeConfig.enabled`: automatise les tâches de calcul de routine, telles que la création et la suppression d' EC2 instances.

- `kubernetesNetworkConfig.elasticLoadBalancing.enabled` : automatise les tâches d'équilibrage de charge, notamment la création et la suppression d'équilibreur de charges Elastic Load Balancer.
- `storageConfig.blockStorage.enabled` : automatise les tâches de stockage, telles que la création et la suppression de volumes Amazon EBS.
- `accessConfig.authenticationMode` : nécessite des entrées d'accès EKS. Pour en savoir plus, consultez Modes d'authentification EKS.

Exécution de la commande

Utilisez la commande suivante pour créer le cluster :

```
aws eks create-cluster \
  --region ${AWS_REGION} \
  --cli-input-json \
  "{
      \"name\": \"${CLUSTER_NAME}\",
      \"version\": \"${K8S_VERSION}\",
      \"roleArn\": \"${CLUSTER_ROLE_ARN}\",
      \"resourcesVpcConfig\": {
        \"subnetIds\": ${SUBNETS_JSON},
        \"endpointPublicAccess\": true,
        \"endpointPrivateAccess\": true
      },
      \"computeConfig\": {
        \"enabled\": true,
        \"nodeRoleArn\":\"${NODE_ROLE_ARN}\",
        \"nodePools\": [\"general-purpose\", \"system\"]
      },
      \"kubernetesNetworkConfig\": {
        \"elasticLoadBalancing\": {
          \"enabled\": true
        }
      },
      \"storageConfig\": {
        \"blockStorage\": {
          \"enabled\": true
        }
      },
      \"accessConfig\": {
        \"authenticationMode\": \"API\"
```

```
      }
    }
```

Vérification de l'état du cluster

Étape 1 : vérifier la création de cluster

Exécutez la commande suivante pour vérifier l'état de votre cluster. La création d'un cluster prend généralement environ 15 minutes :

```
aws eks describe-cluster --name "${CLUSTER_NAME}" --output json
```

Étape 2 : mettre à jour kubeconfig

Une fois le cluster prêt, mettez à jour votre fichier kubeconfig local pour permettre à `kubectl` de communiquer avec le cluster. Cette configuration utilise la AWS CLI pour l'authentification.

```
aws eks update-kubeconfig --name "${CLUSTER_NAME}"
```

Étape 3 : vérifier les groupes de nœuds

Répertoriez les groupes de nœuds présents dans votre cluster à l'aide de la commande suivante :

```
kubectl get nodepools
```

Étapes suivantes

- Découvrez comment déployer un exemple de charge de travail sur votre nouveau cluster du mode automatique EKS.

Création d'un cluster du mode automatique EKS à l'aide de la AWS Management Console

La création d'un cluster du mode automatique EKS dans la AWS Management Console nécessite moins de configuration que les autres options. EKS s'intègre aux réseaux AWS IAM et VPC pour vous aider à créer les ressources associées à un cluster EKS.

Vous disposez de deux options pour créer un cluster dans la console :

- Configuration rapide (avec le mode automatique EKS)
- Configuration personnalisée

Dans cette rubrique, vous apprendrez à créer un cluster du mode automatique EKS en utilisant l'option Configuration rapide.

Création d'un cluster du mode automatique EKS avec l'option de configuration rapide

Vous devez être connecté à la AWS Management Console avec des autorisations suffisantes pour gérer les ressources AWS, y compris :instances EC2, le réseau EC2, les clusters EKS et les rôles IAM.

1. Navigation vers la console EKS
2. Cliquez sur Créer un cluster
3. Vérifiez que l'option Configuration rapide est sélectionnée
4. Déterminez les valeurs suivantes, ou utilisez les valeurs par défaut pour un cluster de test.
   - Nom du cluster
   - Version de Kubernetes
5. Sélectionnez le rôle IAM du cluster. Si c'est la première fois que vous créez un cluster du mode automatique EKS, utilisez l'option Créer un rôle recommandé.
   - Vous pouvez, si vous le souhaitez, réutiliser un seul rôle IAM de cluster dans votre compte AWS pour tous les clusters du mode automatique EKS.
   - Le rôle IAM du cluster inclut les autorisations requises pour le mode automatique EKS afin de gérer les ressources, notamment les instances EC2, les volumes EBS et les équilibreurs de charge EC2.
   - L'option Créer un rôle recommandé préremplit tous les champs avec les valeurs recommandées. Sélectionnez Suivant, puis Créer. Le rôle utilisera le nom `AmazonEKSAutoClusterRole` suggéré.
   - Si vous avez récemment créé un nouveau rôle, utilisez l'icône Actualiser pour recharger le menu déroulant de sélection des rôles.
6. Sélectionnez le rôle IAM de nœud. Si c'est la première fois que vous créez un cluster du mode automatique EKS, utilisez l'option Créer un rôle recommandé.
   - Vous pouvez éventuellement réutiliser un rôle IAM de nœud unique dans votre AWS compte pour tous les clusters du mode automatique EKS.

- Le rôle IAM de nœud inclut les autorisations nécessaires pour permettre aux nœuds du mode automatique de se connecter au cluster. Le rôle IAM de nœud doit également inclure les autorisations permettant d'extraire les images ECR nécessaires à vos conteneurs.
- L'option Créer un rôle recommandé préremplit tous les champs avec les valeurs recommandées. Sélectionnez Suivant, puis Créer. Le rôle utilisera le nom `AmazonEKSAutoNodeRole` suggéré.
- Si vous avez récemment créé un nouveau rôle, utilisez l'icône Actualiser pour recharger le menu déroulant de sélection des rôles.

7. Sélectionnez le VPC pour votre cluster du mode automatique EKS. Sélectionnez Créer un VPC pour créer un nouveau VPC pour EKS, ou choisissez un VPC que vous avez déjà créé pour EKS.
   - Si vous utilisez la console VPC pour créer un nouveau VPC, AWS vous recommande de créer au moins une passerelle NAT par zone de disponibilité. Sinon, vous pouvez utiliser toutes les autres valeurs par défaut.
   - Pour plus d'informations et de détails sur les exigences du cluster IPv6, consultez the section called "Création d'un VPC".
8. (facultatif) Le mode automatique EKS remplit automatiquement les sous-réseaux privés du VPC sélectionné. Vous pouvez supprimer les sous-réseaux indésirables.
   - EKS sélectionne automatiquement les sous-réseaux privés à partir du VPC conformément aux meilleures pratiques. Vous pouvez éventuellement sélectionner des sous-réseaux supplémentaires dans le VPC, tels que des sous-réseaux publics.
9. (facultatif) Sélectionnez Afficher les valeurs par défaut de la configuration rapide pour passer en revue toutes les valeurs de configuration du nouveau cluster. Le tableau indique que certaines valeurs ne sont plus modifiables une fois le cluster créé.
10. Sélectionnez Créer un cluster. Notez que la création du cluster peut prendre quinze minutes.

Étapes suivantes

- Découvrez comment Déployer un exemple de charge de travail sur votre cluster du mode automatique EKS

Activer le mode automatique EKS sur les clusters EKS existants

Vous pouvez activer le mode automatique EKS sur les clusters EKS existants.

AWS prend en charge les migrations suivantes :

- Migration de Karpenter vers les nœuds du mode automatique EKS. Pour de plus amples informations, consultez the section called "Migration à partir de Karpenter".
- Migration des groupes de nœuds gérés par EKS vers des nœuds du mode automatique EKS. Pour de plus amples informations, consultez the section called "Migration à partir des groupes de nœuds gérés".
- Migration d'EKS Fargate vers le mode automatique EKS. Pour de plus amples informations, consultez the section called "Migration à partir de Fargate".

AWS ne prend pas en charge les migrations suivantes :

- Migration des volumes du contrôleur CSI EBS (à l'aide du module complémentaire Amazon EKS) vers le contrôleur CSI EBS en mode automatique EKS (géré par le mode automatique EKS). Les PVC créés avec l'un ne peuvent pas être montés par l'autre, car ils utilisent deux provisionneurs de volume Kubernetes différents.
  - Le `eks-auto-mode-ebs-migration-tool`(projet AWS Labs) permet la migration entre la norme EBS CSI StorageClass (`ebs.csi.aws.com`) et EKS Auto EBS CSI StorageClass (`ebs.csi.eks.amazonaws.com`). Notez que la migration nécessite la suppression et la recréation des ressources PersistentVolumeClaim/PersistentVolume existantes. Il est donc essentiel de procéder à une validation dans un environnement hors production avant la mise en œuvre.
- Migration des équilibreurs de charge de l'AWS Load Balancer Controller vers le mode automatique EKS

  Vous pouvez installer l'AWS Load Balancer Controller sur un cluster du mode automatique Amazon EKS. Utilisez les options `IngressClass` ou `loadBalancerClass` pour associer les ressources de service et d'entrée au Load Balancer Controller ou au mode automatique EKS.
- Migration de clusters EKS avec des CNI alternatifs ou d'autres configurations réseau non prises en charge

Référence de migration

Utilisez la référence de migration suivante pour configurer les ressources Kubernetes afin qu'elles appartiennent soit à des contrôleurs autogérés, soit au mode automatique EKS.

Capacité	Ressource	Champ	Autogéré	Mode automatique EKS
Stockage en mode bloc	`StorageClass`	`provisioner`	`ebs.csi.aws.com`	`ebs.csi.eks.amazonaws.com`
Équilibrage de charge	`Service`	`loadBalancerClass`	`service.k8s.aws/nlb`	`eks.amazonaws.com/nlb`
Équilibrage de charge	`IngressClass`	`controller`	`ingress.k8s.aws/alb`	`eks.amazonaws.com/alb`
Équilibrage de charge	`IngressClassParams`	`apiversion`	`elbv2.k8s.aws/v1beta1`	`eks.amazonaws.com/v1`
Équilibrage de charge	`TargetGroupBinding`	`apiversion`	`elbv2.k8s.aws/v1beta1`	`eks.amazonaws.com/v1`
Calcul	`NodeClass`	`apiVersion`	`karpenter.sh/v1`	`eks.amazonaws.com/v1`

Migration de volumes EBS

Lors de la migration des charges de travail vers le mode automatique EKS, vous devez gérer la migration des volumes EBS en raison des différents provisionneurs de pilotes CSI :

- Provisionneur du mode automatique EKS : `ebs.csi.eks.amazonaws.com`
- Provisionneur CSI EBS open source : `ebs.csi.aws.com`

Pour migrer vos volumes persistants, procédez comme suit :

1. Modifier la politique de conservation des volumes : modifier la version `persistentVolumeReclaimPolicy` existante de la plate-forme (PV) vers `Retain` afin de garantir que le volume EBS sous-jacent ne soit pas supprimé.
2. Supprimer le PV de Kubernetes : supprimez l'ancienne ressource PV tout en préservant le volume EBS réel.
3. Créez un nouveau PV avec provisionnement statique : créez un nouveau PV qui fait référence au même volume EBS mais fonctionne avec le pilote CSI cible.
4. Liaison à un nouveau PVC : créez un nouveau PVC qui fait spécifiquement référence à votre PV en utilisant le champ `volumeName`.

Considérations

- Assurez-vous que vos applications sont arrêtées avant de commencer cette migration.
- Sauvegardez vos données avant de commencer le processus de migration.
- Ce processus doit être effectué pour chaque volume persistant.
- La charge de travail doit être mise à jour pour utiliser le nouveau PVC.

Migration des équilibreurs de charge

Vous ne pouvez pas transférer directement les équilibreurs de charge existants de l'AWS Load Balancer Controller autogéré vers le mode automatique EKS. Vous devez plutôt mettre en œuvre une stratégie de déploiement bleu/vert. Cela implique de conserver votre configuration d'équilibreur de charge existante tout en créant de nouveaux répartiteurs de charge sous le contrôleur géré.

Afin de minimiser les perturbations du service, nous recommandons une approche de transfert du trafic basée sur le DNS. Commencez par créer de nouveaux équilibreurs de charge à l'aide du mode automatique EKS tout en conservant votre configuration existante opérationnelle. Ensuite, utilisez le routage DNS (tel que Route 53) pour transférer progressivement le trafic des anciens équilibreurs de charge vers les nouveaux. Une fois le trafic migré avec succès et la nouvelle configuration vérifiée, vous pouvez mettre hors service les anciens équilibreurs de charge et le contrôleur autogéré.

Activation du mode automatique EKS sur un cluster existant

Cette rubrique explique comment activer le mode automatique EKS sur des clusters Amazon EKS déjà existants. L'activation du mode automatique sur un cluster existant nécessite la mise à jour des autorisations IAM et la configuration des paramètres essentiels du mode automatique EKS.

Une fois activé, vous pouvez commencer à migrer vos charges de calcul existantes afin de profiter des opérations simplifiées et de la gestion automatisée de l'infrastructure offertes par le mode automatique.

> **Important**
>
> Avant d'activer le mode automatique EKS, vérifiez que vous disposez de la version minimale requise de certains modules complémentaires Amazon EKS. Pour de plus amples informations, veuillez consulter the section called "Versions requises des modules complémentaires".

Avant de commencer, assurez-vous également de disposer d'un accès administrateur à votre cluster Amazon EKS et des autorisations nécessaires pour modifier les rôles IAM. Les étapes décrites dans cette rubrique vous expliquent comment activer le mode automatique à l'aide de la AWS CLI AWS Management Console ou de la CLI.

AWS Management Console

Vous devez être connecté à la AWS console avec l'autorisation de gérer IAM, EKS et les EC2 ressources.

> **Note**
>
> Le rôle IAM d'un cluster EKS ne peut pas être modifié après la création du cluster. Le mode automatique EKS nécessite des autorisations supplémentaires pour ce rôle. Vous devez donc attacher des politiques supplémentaires au rôle actuel.

Mise à jour du rôle IAM du cluster

1. Ouvrez la page de présentation de votre cluster dans la AWS Management Console.
2. Sous ARN du rôle IAM du cluster, sélectionnez Afficher dans IAM.
3. Dans la liste déroulante Ajouter des autorisations, sélectionnez Attacher des politiques.
4. Dans la boîte Recherche, trouvez et sélectionnez les politiques suivantes :
   - `AmazonEKSComputePolicy`
   - `AmazonEKSBlockStoragePolicy`
   - `AmazonEKSLoadBalancingPolicy`

- `AmazonEKSNetworkingPolicy`
- `AmazonEKSClusterPolicy`

5. Sélectionnez Ajouter des autorisations
6. Dans l'onglet Relations d'approbation, sélectionnez Modifier la politique d'approbation
7. Insérez la politique d'approbation du rôle IAM du cluster suivante, puis sélectionnez Mettre à jour la politique

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

Activation du mode automatique EKS

1. Ouvrez la page de présentation de votre cluster dans la AWS Management Console.
2. Sous Mode automatique EKS, sélectionnez Gérer
3. Activez l'option Mode automatique EKS.
4. Dans le menu déroulant Groupe de nœuds EKS, sélectionnez les groupes de nœuds par défaut que vous souhaitez créer.
   - Pour plus d'informations, consultez la section sur les groupes de nœuds en mode automatique EKS. Pour de plus amples informations, veuillez consulter the section called "Création d'un groupe de nœuds".
5. Si vous avez déjà créé un rôle IAM de nœud en mode automatique EKS sur ce AWS compte, sélectionnez-le dans le menu déroulant Rôle de nœud IAM. Si vous n'avez pas encore créé ce rôle, sélectionnez Créer le rôle recommandé et suivez les étapes indiquées.

AWS CLI

Conditions préalables

- Le rôle IAM du cluster EKS existant doit inclure les autorisations suffisantes pour le mode automatique EKS, notamment les politiques suivantes :
  - `AmazonEKSComputePolicy`
  - `AmazonEKSBlockStoragePolicy`
  - `AmazonEKSLoadBalancingPolicy`
  - `AmazonEKSNetworkingPolicy`
  - `AmazonEKSClusterPolicy`
- Le rôle IAM du cluster doit également comporter une politique d'approbation mise à jour incluant l'action `sts:TagSession`. Pour plus d'informations sur la création d'un rôle IAM de cluster, consultez the section called " AWS CLI".
- Assurez-vous que la CLI `aws` est installée, connectée et à jour. Vous devez être autorisé à gérer IAM, EKS et les EC2 ressources. Pour de plus amples informations, veuillez consulter Configuration.

Procédure

Utilisez les commandes suivantes pour activer le mode automatique EKS sur un cluster existant.

> **Note**
>
> Les capacités de calcul, de stockage bloc et d'équilibrage de charge doivent toutes être activées ou désactivées dans la même requête.

```
aws eks update-cluster-config \
 --name $CLUSTER_NAME \
 --compute-config enabled=true \
 --kubernetes-network-config '{"elasticLoadBalancing":{"enabled": true}}' \
 --storage-config '{"blockStorage":{"enabled": true}}'
```

Versions requises des modules complémentaires

Si vous prévoyez d'activer le mode automatique EKS sur un cluster existant, il peut être nécessaire de mettre à jour certains modules complémentaires. Remarques :

- Cela s'applique uniquement aux clusters existants en cours de transition vers le mode automatique EKS.
- Les nouveaux clusters créés avec le mode automatique EKS déjà activé n'ont pas besoin de ces mises à jour.

Si l'un des modules complémentaires suivants est installé, assurez-vous qu'il possède au moins la version minimale spécifiée :

Nom du module complémentaire	Version minimale requise
Plug-in CNI Amazon VPC pour Kubernetes	v1.19.0-eksbuild.1
Kube-proxy	• v1.26.15-eksbuild.19 • v1.27.16-eksbuild.14 • v1.28.15-eksbuild.4 • v1.29.10-eksbuild.3 • v1.30.6-eksbuild.3 • v1.31.2-eksbuild.3
Pilote CSI Amazon EBS	v1.37.0-eksbuild.1
Contrôleur d'instantané CSI	v8.1.0-eksbuild.2
Agent d'identité du pod EKS	v1.3.4-eksbuild.1

Pour de plus amples informations, veuillez consulter the section called "Mettre à jour un module complémentaire".

Étapes suivantes

- Pour migrer les charges de travail de gestion des groupes de nœuds, consultez the section called "Migration à partir des groupes de nœuds gérés".

- Pour migrer depuis Karpenter autogéré, consultez the section called "Migration à partir de Karpenter".

Migration de Karpenter vers le mode automatique EKS à l'aide de kubectl

Cette rubrique décrit en détail le processus de migration des charges de travail de Karpenter vers le mode automatique Amazon EKS à l'aide de kubectl. La migration peut être effectuée progressivement, vous permettant de transférer les charges de travail à votre propre rythme, tout en préservant la stabilité du cluster et la disponibilité des applications pendant toute la transition.

L' step-by-stepapproche décrite ci-dessous vous permet d'exécuter Karpenter et le mode automatique d'EKS côte à côte pendant la période de migration. Cette stratégie de double exploitation garantit une transition fluide, car elle vous permet de valider le comportement des charges de travail en mode automatique EKS avant de désactiver complètement Karpenter. Vous pouvez migrer les applications individuellement ou par groupes, ce qui offre une flexibilité accrue pour répondre à vos exigences opérationnelles spécifiques et à votre tolérance au risque.

Conditions préalables

Avant de commencer la migration, assurez-vous d'avoir :

- Karpenter v1.1 ou une version ultérieure installée sur votre cluster. Pour plus d'informations, consultez Mise à niveau vers la version 1.1.0+ dans la documentation Karpenter.
- `kubectl` installé et connecté à votre cluster. Pour de plus amples informations, veuillez consulter Configuration.

Cette rubrique suppose que vous connaissez Karpenter et NodePools. Pour plus d'informations, consultez la Documentation Karpenter.

Étape 1 : activer le mode automatique EKS sur le cluster

Activez le mode automatique EKS sur votre cluster existant à l'aide de la AWS CLI ou de la console de gestion. Pour de plus amples informations, veuillez consulter the section called "Activation sur un cluster".

> **Note**
>
> Lors de l'activation du mode automatique EKS, n'activez pas le groupe de nœuds `general purpose` à ce stade de la transition. Ce groupe de nœuds n'est pas sélectif.

Pour de plus amples informations, veuillez consulter the section called "Vérifier les groupes de nœuds créés".

Étape 2 : Création d'un mode automatique EKS altéré NodePool

Créez un nouveau mode automatique NodePool pour EKS avec une teinte. Cela garantit que les pods existants ne seront pas automatiquement planifiés sur les nouveaux nœuds du mode automatique EKS. Ce groupe de nœuds utilise la `NodeClass default` intégrée au mode automatique EKS. Pour de plus amples informations, veuillez consulter the section called "Création d'une classe de nœuds".

Exemple de groupe de nœuds avec balise de rejet :

```
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: eks-auto-mode
spec:
  template:
    spec:
      requirements:
        - key: "eks.amazonaws.com/instance-category"
          operator: In
          values: ["c", "m", "r"]
      nodeClassRef:
        group: eks.amazonaws.com
        kind: NodeClass
        name: default
      taints:
        - key: "eks-auto-mode"
          effect: "NoSchedule"
```

Mettez à jour les exigences du groupe de nœuds afin qu'elles correspondent à la configuration Karpenter d'origine à partir de laquelle vous effectuez la migration. Vous devez définir au moins une exigence.

Étape 3 : mettre à jour les charges de travail pour la migration

Identifiez et mettez à jour les charges de travail que vous souhaitez migrer vers le mode automatique EKS. Ajoutez des tolérances et des sélecteurs de nœuds à ces charges de travail :

```
apiVersion: apps/v1
kind: Deployment
spec:
  template:
    spec:
      tolerations:
      - key: "eks-auto-mode"
        effect: "NoSchedule"
      nodeSelector:
        eks.amazonaws.com/compute-type: auto
```

Cette modification permet de planifier la charge de travail sur les nouveaux nœuds du mode automatique EKS.

Le mode automatique EKS utilise des étiquettes différentes de celles de Karpenter. Les étiquettes associées aux instances EC2 gérées commencent par`eks.amazonaws.com`. Pour de plus amples informations, veuillez consulter the section called "Création d'un groupe de nœuds".

Étape 4 : migrer progressivement les charges de travail

Répétez l'étape 3 pour chaque charge de travail que vous souhaitez migrer. Cette méthode vous permet de déplacer les charges de travail individuellement ou par groupes, selon vos besoins et votre tolérance au risque.

Étape 5 : Retirez le Karpenter d'origine NodePool

Une fois que toutes les charges de travail ont été migrées, vous pouvez supprimer le Karpenter d'origine : NodePool

```
kubectl delete nodepool <original-nodepool-name>
```

Étape 6 : Supprimer les taches du mode automatique EKS NodePool (facultatif)

Si vous souhaitez que le mode automatique EKS devienne le mode par défaut pour les nouvelles charges de travail, vous pouvez supprimer les traces du mode automatique EKS : NodePool

```
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
```

```
  name: eks-auto-mode
spec:
  template:
    spec:
      nodeClassRef:
        group: eks.amazonaws.com
        kind: NodeClass
        name: default
      # Remove the taints section
```

Étape 7 : supprimer les sélecteurs de nœuds des charges de travail (facultatif)

Si vous avez supprimé l'altération du mode automatique EKS NodePool, vous pouvez éventuellement supprimer les sélecteurs de nœuds de vos charges de travail, le mode automatique EKS étant désormais le mode par défaut :

```
apiVersion: apps/v1
kind: Deployment
spec:
  template:
    spec:
      # Remove the nodeSelector section
      tolerations:
      - key: "eks-auto-mode"
        effect: "NoSchedule"
```

Étape 8 : désinstaller Karpenter de votre cluster

Les étapes de suppression de Karpenter dépendent de la méthode d'installation que vous avez utilisée. Pour plus d'informations, consultez les Instructions d'installation de Karpenter.

Migration des groupes de nœuds gérés EKS vers le mode automatique EKS

Lors de la transition de votre cluster Amazon EKS vers le mode automatique EKS, vous pouvez migrer en douceur vos charges de travail existantes à partir des groupes de nœuds gérés (Managed Node Groups, MNG) à l'aide de l'outil CLI eksctl. Ce processus garantit la disponibilité continue des applications, tout en permettant au mode automatique EKS d'optimiser vos ressources de calcul. La migration peut être effectuée avec un minimum d'interruptions pour vos applications en cours d'exécution.

Cette rubrique présente les étapes à suivre pour évacuer en toute sécurité les pods de vos groupes de nœuds gérés existants et permettre au mode automatique EKS de les replanifier automatiquement sur de nouvelles instances provisionnées. En suivant cette procédure, vous pourrez tirer parti de la consolidation intelligente des charges de travail en mode automatique EKS, tout en préservant la disponibilité de vos applications pendant toute la durée de la migration.

Prérequis

- Cluster avec le mode automatique EKS activé
- CLI `eksctl` installée et connectée à votre cluster. Pour de plus amples informations, consultez Configuration.
- Karpenter non installé sur le cluster.

Procédure

Utilisez la commande CLI `eksctl` suivante pour commencer à évacuer les pods des instances de groupes de nœuds gérés existants. Le mode automatique EKS créera automatiquement de nouveaux nœuds pour héberger les pods déplacés.

```
eksctl delete nodegroup --cluster=<clusterName> --name=<nodegroupName>
```

Vous devrez exécuter cette commande pour chaque groupe de nœuds géré de votre cluster.

Pour plus d'informations sur cette commande, consultez Suppression et évacuation des groupes de nœuds dans la documentation eksctl.

Migration d'EKS Fargate vers le mode automatique EKS

Cette rubrique décrit en détail le processus de migration des charges de travail d'EKS Fargate vers le mode automatique Amazon EKS à l'aide de `kubectl`. La migration peut être effectuée progressivement, vous permettant de transférer les charges de travail à votre propre rythme, tout en préservant la stabilité du cluster et la disponibilité des applications pendant toute la transition.

L' step-by-stepapproche décrite ci-dessous vous permet d'exécuter EKS Fargate et EKS Auto Mode côte à côte pendant la période de migration. Cette stratégie de double exploitation garantit une transition fluide, car elle vous permet de valider le comportement des charges de travail sous le mode automatique EKS avant de désactiver complètement EKS Fargate. Vous pouvez migrer les

applications individuellement ou par groupes, ce qui offre une flexibilité accrue pour répondre à vos exigences opérationnelles spécifiques et à votre tolérance au risque.

Comparaison entre le mode automatique d'Amazon EKS et EKS avec AWS Fargate

Amazon EKS with AWS Fargate reste une option pour les clients qui souhaitent exécuter EKS, mais le mode automatique Amazon EKS est l'approche recommandée pour l'avenir. Le mode automatique EKS est entièrement conforme à Kubernetes, et prend en charge tous les éléments natifs de Kubernetes ainsi que les outils de plateforme tels qu'Istio, que Fargate ne prend pas en charge. Le mode automatique EKS prend également en charge toutes les options d'exécution EC2, y compris les instances GPU et Spot, permettant ainsi aux clients de bénéficier des remises EC2 négociées et d'autres mécanismes d'optimisation des coûts. Ces fonctionnalités ne sont pas disponibles avec EKS associé à Fargate.

De plus, le mode automatique EKS permet d'obtenir le même modèle d'isolation que Fargate, en utilisant les capacités natives de planification de Kubernetes pour garantir que chaque instance EC2 exécute un seul conteneur d'application. En adoptant le mode automatique d'Amazon EKS, les clients peuvent profiter de tous les avantages liés à l'exécution de Kubernetes sur AWS une plateforme entièrement conforme à Kubernetes qui offre la flexibilité nécessaire pour tirer parti de l'ensemble des options d'achat et d'EC2 tout en conservant la facilité d'utilisation et l'abstraction de la gestion de l'infrastructure proposées par Fargate.

Obtenir une isolation semblable à celle de Fargate en mode automatique EKS

Pour répliquer le modèle d'isolation des pods de Fargate dans lequel chaque pod s'exécute sur sa propre instance dédiée, vous pouvez utiliser les contraintes de propagation de la topologie Kubernetes. Voici l'approche recommandée pour contrôler la distribution des pods entre les nœuds :

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: isolated-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: isolated-app
  template:
    metadata:
      labels:
        app: isolated-app
```

```
      annotations:
        eks.amazonaws.com/compute-type: ec2
    spec:
      topologySpreadConstraints:
      - maxSkew: 1
        topologyKey: kubernetes.io/hostname
        whenUnsatisfiable: DoNotSchedule
        labelSelector:
          matchLabels:
            app: isolated-app
        minDomains: 1
      containers:
      - name: app
        image: nginx
        ports:
        - containerPort: 80
```

Dans cette configuration :

- `maxSkew: 1`garantit que la différence de nombre de pods entre deux nœuds est d'au plus 1, en répartissant efficacement un pod par nœud
- `topologyKey: kubernetes.io/hostname`définit le nœud en tant que domaine topologique
- `whenUnsatisfiable: DoNotSchedule`empêche la planification si la contrainte ne peut pas être respectée
- `minDomains: 1`s'assure qu'au moins un domaine (nœud) existe avant la planification

Le mode automatique d'EKS provisionne automatiquement les nouvelles instances EC2 selon les besoins pour satisfaire cette contrainte, en fournissant le même modèle d'isolation que Fargate tout en vous donnant accès à la gamme complète des types d'instances EC2 et des options d'achat.

Vous pouvez également utiliser les règles d'anti-affinité des pods pour une isolation plus stricte :

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: isolated-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: isolated-app
```

```
  template:
    metadata:
      labels:
        app: isolated-app
      annotations:
        eks.amazonaws.com/compute-type: ec2
    spec:
      affinity:
        podAntiAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
          - labelSelector:
              matchExpressions:
              - key: app
                operator: In
                values:
                - isolated-app
            topologyKey: kubernetes.io/hostname
      containers:
      - name: app
        image: nginx
        ports:
        - containerPort: 80
```

La `podAntiAffinity` règle `requiredDuringSchedulingIgnoredDuringExecution` garantit qu'aucun module portant l'étiquette ne `app: isolated-app` peut être planifié sur le même nœud. Cette approche fournit des garanties d'isolation strictes similaires à celles de Fargate.

Conditions préalables

Avant de commencer la migration, assurez-vous d'avoir

- Mis en place un cluster avec Fargate. Pour de plus amples informations, veuillez consulter the section called "Mise en route".
- Installé et connecté `kubectl` à votre cluster. Pour de plus amples informations, veuillez consulter Configuration.

Étape 1 : vérifier le cluster Fargate

1. Vérifiez si le cluster EKS avec Fargate est en cours d'exécution :

```
kubectl get node
```

```
NAME STATUS ROLES AGE VERSION
fargate-ip-192-168-92-52.ec2.internal Ready <none> 25m v1.30.8-eks-2d5f260
fargate-ip-192-168-98-196.ec2.internal Ready <none> 24m v1.30.8-eks-2d5f260
```

2. Vérifiez les pods en cours d'exécution :

```
kubectl get pod -A
```

```
NAMESPACE NAME READY STATUS RESTARTS AGE
kube-system coredns-6659cb98f6-gxpjz 1/1 Running 0 26m
kube-system coredns-6659cb98f6-gzzsx 1/1 Running 0 26m
```

3. Créez un déploiement dans un fichier appelé `deployment_fargate.yaml` :

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
      annotations:
        eks.amazonaws.com/compute-type: fargate
    spec:
      containers:
      - name: nginx
        image: nginx
        ports:
        - containerPort: 80
```

4. Appliquez le déploiement :

```
kubectl apply -f deployment_fargate.yaml
```

```
deployment.apps/nginx-deployment created
```

5. Vérifiez les pods et les déploiements :

```
kubectl get pod,deploy
```

```
NAME                                    READY   STATUS    RESTARTS   AGE
pod/nginx-deployment-5c7479459b-6trtm   1/1     Running   0          61s
pod/nginx-deployment-5c7479459b-g8ssb   1/1     Running   0          61s
pod/nginx-deployment-5c7479459b-mq4mf   1/1     Running   0          61s

NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/nginx-deployment   3/3     3            3           61s
```

6. Vérifiez le nœud :

```
kubectl get node -owide
```

```
NAME                                      STATUS  ROLES  AGE VERSION
   INTERNAL-IP     EXTERNAL-IP OS-IMAGE       KERNEL-VERSION
 CONTAINER-RUNTIME
fargate-ip-192-168-111-43.ec2.internal  Ready   <none> 31s v1.30.8-eks-2d5f260
 192.168.111.43  <none>      Amazon Linux 2 5.10.234-225.910.amzn2.x86_64
 containerd://1.7.25
fargate-ip-192-168-117-130.ec2.internal Ready   <none> 36s v1.30.8-eks-2d5f260
 192.168.117.130 <none>      Amazon Linux 2 5.10.234-225.910.amzn2.x86_64
 containerd://1.7.25
fargate-ip-192-168-74-140.ec2.internal  Ready   <none> 36s v1.30.8-eks-2d5f260
 192.168.74.140  <none>      Amazon Linux 2 5.10.234-225.910.amzn2.x86_64
 containerd://1.7.25
```

Étape 2 : activer le mode automatique EKS sur le cluster

1. Activez le mode automatique EKS sur votre cluster existant à l'aide de la AWS CLI ou de la console de gestion. Pour de plus amples informations, veuillez consulter the section called "Activation sur un cluster".
2. Vérifiez le groupe de nœuds :

```
kubectl get nodepool
```

```
NAME              NODECLASS   NODES   READY   AGE
general-purpose   default     1       True    6m58s
system            default     0       True    3d14h
```

Étape 3 : mettre à jour les charges de travail pour la migration

Identifiez et mettez à jour les charges de travail que vous souhaitez migrer vers le mode automatique EKS.

Pour migrer une charge de travail de Fargate vers le mode automatique EKS, appliquez l'annotation `eks.amazonaws.com/compute-type: ec2`. Cela garantit que la charge de travail ne sera pas planifiée par Fargate, malgré le profil Fargate, et qu'elle sera absorbée par le mode automatique EKS. NodePool Pour de plus amples informations, veuillez consulter the section called "Création d'un groupe de nœuds".

1. Modifiez vos déploiements (par exemple le fichier `deployment_fargate.yaml`) afin de changer le type de calcul en `ec2` :

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
      annotations:
        eks.amazonaws.com/compute-type: ec2
    spec:
      containers:
```

```
      - name: nginx
        image: nginx
        ports:
        - containerPort: 80
```

2. Appliquez le déploiement. Cette modification permet de planifier la charge de travail sur les nouveaux nœuds du mode automatique EKS :

```
kubectl apply -f deployment_fargate.yaml
```

3. Vérifiez que le déploiement est en cours d'exécution dans le cluster du mode automatique EKS :

```
kubectl get pod -o wide
```

```
NAME                               READY   STATUS    RESTARTS   AGE     IP
    NODE                   NOMINATED NODE   READINESS GATES
nginx-deployment-97967b68d-ffxxh   1/1     Running   0          3m31s
 192.168.43.240   i-0845aafcb51630ffb   <none>           <none>
nginx-deployment-97967b68d-mbcgj   1/1     Running   0          2m37s
 192.168.43.241   i-0845aafcb51630ffb   <none>           <none>
nginx-deployment-97967b68d-qpd8x   1/1     Running   0          2m35s
 192.168.43.242   i-0845aafcb51630ffb   <none>           <none>
```

4. Vérifiez qu'aucun nœud Fargate n'est en cours d'exécution et que le déploiement s'exécute sur les nœuds gérés en mode automatique EKS :

```
kubectl get node -owide
```

```
NAME                 STATUS ROLES  AGE   VERSION              INTERNAL-IP     EXTERNAL-
IP OS-IMAGE                                          KERNEL-VERSION CONTAINER-RUNTIME
i-0845aafcb51630ffb Ready  <none> 3m30s v1.30.8-eks-3c20087 192.168.41.125
 3.81.118.95 Bottlerocket (EKS Auto) 2025.3.14 (aws-k8s-1.30) 6.1.129
 containerd://1.7.25+bottlerocket
```

Étape 4 : migrer progressivement les charges de travail

Répétez l'étape 3 pour chaque charge de travail que vous souhaitez migrer. Cette méthode vous permet de déplacer les charges de travail individuellement ou par groupes, selon vos besoins et votre tolérance au risque.

Étape 5 : supprimer le profil Fargate d'origine

Une fois que toutes les charges de travail ont été migrées, vous pouvez supprimer le profil `fargate` d'origine. Remplacez *<fargate profile name>* par le nom de votre profil Fargate :

```
aws eks delete-fargate-profile --cluster-name eks-fargate-demo-cluster --fargate-
profile-name <fargate profile name>
```

Étape 6 : réduire verticalement CoreDNS

Comme le mode automatique EKS prend en charge CoreDNS, vous devez réduire le déploiement `coredns` à 0 :

```
kubectl scale deployment coredns -n kube-system —-replicas=0
```

Exécution d'exemples de charges de travail dans des clusters du mode automatique EKS

Ce chapitre fournit des exemples de déploiement de différents types de charges de travail sur des clusters Amazon EKS exécutés en mode automatique. Les exemples illustrent des modèles de charges de travail clés, notamment des exemples d'applications, des applications web avec équilibrage de charge, des charges de travail avec état utilisant du stockage persistant, et des charges de travail nécessitant un placement spécifique des nœuds. Chaque exemple inclut des manifestes complets et des instructions de step-by-step déploiement que vous pouvez utiliser comme modèles pour vos propres applications.

Avant de passer aux exemples, assurez-vous qu'un cluster EKS fonctionne en mode automatique et que vous avez installé la AWS CLI et kubectl. Pour de plus amples informations, veuillez consulter Configuration. Les exemples présupposent une connaissance de base des concepts Kubernetes et des commandes kubectl.

Vous pouvez utiliser ces exemples basés sur des cas d'utilisation pour exécuter des charges de travail dans des clusters du mode automatique EKS.

the section called "Déploiement d'une charge de travail de gonflement"

> Montre comment déployer un exemple de charge de travail sur un cluster du mode automatique EKS à l'aide de commandes `kubectl`.

the section called "Déploiement de l'équilibreur de charge"

Montre comment déployer une version conteneurisée du jeu 2048 sur Amazon EKS.

the section called "Déployer une charge de travail avec état"

Montre comment déployer un exemple d'application dynamique sur un cluster du mode automatique EKS.

the section called "Déployer une charge de travail accélérée"

Montre comment déployer des charges de travail accélérées par le matériel sur des nœuds gérés par le mode automatique EKS.

the section called "Contrôle du déploiement"

Montre comment utiliser une annotation pour contrôler si une charge de travail est déployée sur des nœuds gérés par le mode automatique EKS.

Déploiement d'une charge de travail de gonflement dans un cluster du mode automatique Amazon EKS

Ce tutoriel montre comment déployer une charge de travail d'exemple dans un cluster du mode automatique EKS et observer le provisionnement automatique des ressources de calcul nécessaires. Vous utiliserez des `kubectl` commandes pour observer le comportement du cluster et voir par vous-même comment le mode automatique simplifie les opérations de Kubernetes sur. AWSÀ la fin de ce tutoriel, vous comprendrez comment le mode automatique EKS réagit aux déploiements de charges de travail en gérant automatiquement les ressources de calcul sous-jacentes, sans configuration manuelle de groupes de nœuds.

Conditions préalables

- Un cluster du mode automatique Amazon EKS. Notez le nom et AWS la région du cluster.
- Un principal IAM, tel qu'un utilisateur ou un rôle, disposant d'autorisations suffisantes pour gérer le réseau, le calcul et les ressources EKS.
  - Pour plus d'informations, consultez la section Création de rôles et attribution de politiques dans le Guide de l'utilisateur IAM.
- CLI `aws` installée et configurée avec une identité IAM.
- CLI `kubectl` installée et connectée au cluster.

- Pour de plus amples informations, veuillez consulter Configuration.

Étape 1 : consulter les ressources de calcul existantes (facultatif)

Tout d'abord, utilisez `kubectl` pour répertorier les groupes de nœuds de votre cluster.

```
kubectl get nodepools
```

Exemple de résultat :

```
general-purpose
```

Dans ce tutoriel, nous allons déployer une charge de travail configurée pour utiliser le groupe de nœuds `general-purpose`. Ce groupe de nœuds est intégré au mode automatique EKS et propose des valeurs par défaut raisonnables pour des charges générales, comme des microservices et des applications web. Vous pouvez créer votre propre groupe de nœuds. Pour de plus amples informations, veuillez consulter the section called "Création d'un groupe de nœuds".

Ensuite, utilisez `kubectl` pour répertorier les nœuds connectés à votre cluster.

```
kubectl get nodes
```

Si vous venez de créer un cluster du mode automatique EKS, vous n'aurez aucun nœud.

Dans ce tutoriel, vous allez déployer un exemple de charge de travail. Si vous n'avez aucun nœud ou si la charge de travail ne peut pas être adaptée aux nœuds existants, le mode automatique EKS provisionnera un nouveau nœud.

Étape 2 : déployer un exemple d'application sur votre cluster

Consultez le déploiement Kubernetes suivant et enregistrez-le sous `inflate.yaml`

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: inflate
spec:
  replicas: 1
  selector:
```

```
    matchLabels:
      app: inflate
  template:
    metadata:
      labels:
        app: inflate
    spec:
      terminationGracePeriodSeconds: 0
      nodeSelector:
        eks.amazonaws.com/compute-type: auto
      securityContext:
        runAsUser: 1000
        runAsGroup: 3000
        fsGroup: 2000
      containers:
        - name: inflate
          image: public.ecr.aws/eks-distro/kubernetes/pause:3.7
          resources:
            requests:
              cpu: 1
          securityContext:
            allowPrivilegeEscalation: false
```

Notez que le sélecteur `eks.amazonaws.com/compute-type: auto` nécessite que la charge de travail soit déployée sur un nœud du mode automatique Amazon EKS.

Appliquez le déploiement à votre cluster.

```
kubectl apply -f inflate.yaml
```

Étape 3 : surveiller les événements Kubernetes

Utilisez la commande suivante pour surveiller les événements Kubernetes, notamment la création d'un nouveau nœud. Utilisez `ctrl+c` pour arrêter la surveillance des événements.

```
kubectl get events -w --sort-by '.lastTimestamp'
```

Utilisez `kubectl` pour répertorier à nouveau les nœuds connectés à votre cluster. Notez le nœud nouvellement créé.

```
kubectl get nodes
```

Étape 4 : Afficher les nœuds et les instances dans la AWS console

Vous pouvez afficher les nœuds du mode automatique EKS dans la console EKS et les EC2 instances associées dans la EC2 console.

EC2 Les instances déployées par le mode automatique d'EKS sont soumises à des restrictions. Vous ne pouvez pas exécuter de commandes arbitraires sur les nœuds du mode automatique EKS.

Étape 5 : supprimer le déploiement

Utilisez `kubectl` pour supprimer l'exemple de déploiement

```
kubectl delete -f inflate.yaml
```

Si aucune autre charge de travail n'est déployée sur votre cluster, le nœud créé par le mode automatique EKS sera vide.

Dans la configuration par défaut, le mode automatique EKS détecte les nœuds vides depuis trente secondes et les arrête.

Utilisez `kubectl` la EC2 console pour confirmer que l'instance associée a été supprimée.

Déploiement d'une charge de travail de l'équilibreur de charge en mode automatique EKS

Ce guide explique comment déployer une version conteneurisée du jeu 2048 sur Amazon EKS, avec équilibreur de charge et accès Internet.

Prérequis

- Un cluster du mode automatique EKS
- `kubectl` configuré pour interagir avec votre cluster
- Autorisations IAM nécessaires pour créer des ressources ALB

Étape 1 : créer l'espace de noms

Créez d'abord un espace de noms dédié pour l'application de jeu 2048.

Créez un fichier nommé `01-namespace.yaml`:

```
apiVersion: v1
kind: Namespace
metadata:
  name: game-2048
```

Appliquez la configuration de l'espace de noms :

```
kubectl apply -f 01-namespace.yaml
```

Étape 2 : déployer l'application

L'application exécute plusieurs réplicas du conteneur du jeu 2048.

Créez un fichier nommé `02-deployment.yaml`:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  namespace: game-2048
  name: deployment-2048
spec:
  selector:
    matchLabels:
      app.kubernetes.io/name: app-2048
  replicas: 5
  template:
    metadata:
      labels:
        app.kubernetes.io/name: app-2048
    spec:
      containers:
        - image: public.ecr.aws/l6m2t8p7/docker-2048:latest
          imagePullPolicy: Always
          name: app-2048
          ports:
            - containerPort: 80
          resources:
            requests:
              cpu: "0.5"
```

> **Note**
>
> Si vous recevez une erreur lors du chargement de l'image `public.ecr.aws/l6m2t8p7/docker-2048:latest`, vérifiez que votre rôle IAM de nœud possède les autorisations nécessaires pour extraire les images depuis ECR. Pour de plus amples informations, consultez the section called "Rôle IAM de nœud". De plus, l'image `docker-2048` utilisée dans l'exemple est une image `x86_64` et ne fonctionnera pas sur d'autres architectures.

Composants clés :

- Déploiement de 5 réplicas de l'application
- Utilisation d'une image ECR publique
- Demande de 0,5 cœur de processeur par pod
- Exposition du port 80 pour le trafic HTTP

Appliquez le déploiement :

```
kubectl apply -f 02-deployment.yaml
```

Étape 3 : créer le service

Le service expose le déploiement au réseau du cluster.

Créez un fichier nommé `03-service.yaml`:

```
apiVersion: v1
kind: Service
metadata:
  namespace: game-2048
  name: service-2048
spec:
  ports:
    - port: 80
      targetPort: 80
      protocol: TCP
  selector:
    app.kubernetes.io/name: app-2048
```

Composants clés :

- Création d'un service NodePort
- Mappage du port 80 vers le port 80 du conteneur
- Utilisation d'un sélecteur d'étiquettes pour cibler les pods

Appliquez le service :

```
kubectl apply -f 03-service.yaml
```

Étape 4 : configurer l'équilibrage de charge

Vous allez configurer un objet Ingress pour exposer l'application sur Internet.

Commencez par créer la `IngressClass`. Créez un fichier nommé `04-ingressclass.yaml`:

```
apiVersion: networking.k8s.io/v1
kind: IngressClass
metadata:
  labels:
    app.kubernetes.io/name: LoadBalancerController
  name: alb
spec:
  controller: eks.amazonaws.com/alb
```

> **Note**
>
> Le mode automatique EKS nécessite l'ajout de balises sur les sous-réseaux afin d'identifier les sous-réseaux publics et privés.
> Si vous avez créé votre cluster avec `eksctl`, ces balises sont déjà présentes.
> Découvrez comment the section called "Baliser les sous-réseaux".

Créez ensuite la ressource Ingress. Créez un fichier nommé `05-ingress.yaml`:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
```

```
  namespace: game-2048
  name: ingress-2048
  annotations:
    alb.ingress.kubernetes.io/scheme: internet-facing
    alb.ingress.kubernetes.io/target-type: ip
spec:
  ingressClassName: alb
  rules:
    - http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: service-2048
                port:
                  number: 80
```

Composants clés :

- Création d'un ALB orienté Internet
- Utilisation du type de cible IP pour un routage direct vers les pods
- Routage de tout le trafic (/) vers le service du jeu

Appliquez les configurations Ingress :

```
kubectl apply -f 04-ingressclass.yaml
kubectl apply -f 05-ingress.yaml
```

Étape 5 : vérifier le déploiement

1. Vérifiez que tous les pods sont en cours d'exécution :

   ```
   kubectl get pods -n game-2048
   ```

2. Vérifiez que le service a été créé :

   ```
   kubectl get svc -n game-2048
   ```

3. Obtenez le point de terminaison de l'ALB :

```
kubectl get ingress -n game-2048
```

Le champ ADDRESS dans le résultat Ingress affiche le point de terminaison de votre ALB. Attendez 2 à 3 minutes pour que l'ALB soit entièrement provisionné et enregistre toutes les cibles.

Étape 6 : accéder au jeu

Ouvrez votre navigateur web et accédez à l'URL du point de terminaison de l'ALB obtenue à l'étape précédente. L'interface du jeu 2048 devrait s'afficher.

Étape 7 : nettoyage

Pour supprimer toutes les ressources créées dans ce tutoriel :

```
kubectl delete namespace game-2048
```

Cela supprimera toutes les ressources de l'espace de noms, y compris les ressources Déploiement, Service et Ingress.

Ce qui se passe dans les coulisses

1. Le déploiement crée 5 pods exécutant le jeu 2048
2. Le service fournit un accès réseau stable à ces pods
3. Mode automatique EKS :
   - Crée un équilibreur de charge Application Load Balancer dans AWS
   - Configure les groupes cibles pour les pods
   - Définit les règles de routage pour transmettre le trafic au service

Résolution des problèmes

Si le jeu ne se charge pas :

- Assurez-vous que tous les pods sont en cours d'exécution : `kubectl get pods -n game-2048`
- Vérifiez l'état de la ressource Ingress : `kubectl describe ingress -n game-2048`
- Vérifiez la surveillance de l'état de l'ALB : consultez l'état du groupe cible dans la console AWS

Déployer un exemple de charge de travail avec état dans le mode automatique EKS

Ce tutoriel vous guidera dans le déploiement d'un exemple d'application avec état dans votre cluster du mode automatique EKS. L'application écrit des horodatages dans un volume persistant, démontrant ainsi les capacités de provisionnement et de persistance automatiques des volumes EBS du mode automatique EKS.

Prérequis

- Un cluster du mode automatique EKS
- L'interface AWS CLI configurée avec les autorisations appropriées
- `kubectl` installé et configuré
  - Pour de plus amples informations, consultez Configuration.

Étape 1 : configurer votre environnement

1. Définissez vos variables d'environnement :

```
export CLUSTER_NAME=my-auto-cluster
export AWS_REGION="us-west-2"
```

2. Mettez à jour votre kubeconfig :

```
aws eks update-kubeconfig --name "${CLUSTER_NAME}"
```

Étape 2 : créer la classe de stockage

La `StorageClass` définit la manière dont le mode automatique EKS provisionnera les volumes EBS.

Le mode automatique EKS ne crée pas de `StorageClass` pour vous. Vous devez créer une `StorageClass` faisant référence à `ebs.csi.eks.amazonaws.com` pour utiliser les fonctionnalités de stockage du mode automatique EKS.

1. Créez un fichier nommé `storage-class.yaml`:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
```

```
metadata:
  name: auto-ebs-sc
  annotations:
    storageclass.kubernetes.io/is-default-class: "true"
provisioner: ebs.csi.eks.amazonaws.com
volumeBindingMode: WaitForFirstConsumer
parameters:
  type: gp3
  encrypted: "true"
```

2. Appliquez le `StorageClass` :

```
kubectl apply -f storage-class.yaml
```

Composants clés :

- `provisioner: ebs.csi.eks.amazonaws.com` : utilise le mode automatique EKS
- `volumeBindingMode: WaitForFirstConsumer` : retarde la création du volume jusqu'à ce qu'un pod en ait besoin
- `type: gp3` : spécifie le type de volume EBS
- `encrypted: "true"` : EBS utilisera la clé `aws/ebs` par défaut pour activer le chiffrement sur les volumes créés avec cette classe. Cette action est facultative, mais recommandée.
- `storageclass.kubernetes.io/is-default-class: "true"` : Kubernetes utilisera cette classe de stockage par défaut, à moins que vous ne spécifiiez une autre classe de volume dans une demande de volume persistant. Soyez prudent lorsque vous définissez cette valeur si vous migrez depuis un autre contrôleur de stockage. (facultatif)

Étape 3 : créer la revendication de volume persistant

La PVC demande du stockage à la `StorageClass`.

1. Créez un fichier nommé `pvc.yaml`:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: auto-ebs-claim
spec:
```

```
  accessModes:
    - ReadWriteOnce
  storageClassName: auto-ebs-sc
  resources:
    requests:
      storage: 8Gi
```

2. Appliquez le PVC :

```
kubectl apply -f pvc.yaml
```

Composants clés :

- `accessModes: ReadWriteOnce` : le volume peut être monté par un nœud à la fois
- `storage: 8Gi` : demande un volume de 8 Gio
- `storageClassName: auto-ebs-sc` : fait référence à la `StorageClass` que nous avons créée

Étape 4 : déployer l'application

Le déploiement exécute un conteneur qui écrit des horodatages sur le volume persistant.

1. Créez un fichier nommé `deployment.yaml`:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: inflate-stateful
spec:
  replicas: 1
  selector:
    matchLabels:
      app: inflate-stateful
  template:
    metadata:
      labels:
        app: inflate-stateful
    spec:
      terminationGracePeriodSeconds: 0
      nodeSelector:
        eks.amazonaws.com/compute-type: auto
```

```
      containers:
        - name: bash
          image: public.ecr.aws/docker/library/bash:4.4
          command: ["/usr/local/bin/bash"]
          args: ["-c", "while true; do echo $(date -u) >> /data/out.txt; sleep 60;
 done"]
          resources:
            requests:
              cpu: "1"
          volumeMounts:
            - name: persistent-storage
              mountPath: /data
      volumes:
        - name: persistent-storage
          persistentVolumeClaim:
            claimName: auto-ebs-claim
```

2. Appliquez le déploiement :

```
kubectl apply -f deployment.yaml
```

Composants clés :

- Conteneur bash simple qui écrit des horodatages dans un fichier
- Monte le PVC à `/data`
- Demande 1 cœur de processeur
- Utilise le sélecteur de nœuds pour les nœuds gérés par EKS

Étape 5 : vérifier la configuration

1. Vérifiez que le pod est en cours d'exécution :

```
kubectl get pods -l app=inflate-stateful
```

2. Vérifiez que le PVC est lié :

```
kubectl get pvc auto-ebs-claim
```

3. Vérifiez le volume EBS :

```
# Get the PV name
PV_NAME=$(kubectl get pvc auto-ebs-claim -o jsonpath='{.spec.volumeName}')
# Describe the EBS volume
aws ec2 describe-volumes \
  --filters Name=tag:CSIVolumeName,Values=${PV_NAME}
```

4. Vérifiez que les données sont écrites :

```
kubectl exec "$(kubectl get pods -l app=inflate-stateful \
  -o=jsonpath='{.items[0].metadata.name}')" -- \
  cat /data/out.txt
```

Étape 6 : nettoyage

Exécutez la commande suivante pour supprimer toutes les ressources créées dans ce tutoriel :

```
# Delete all resources in one command
kubectl delete deployment/inflate-stateful pvc/auto-ebs-claim storageclass/auto-ebs-sc
```

Ce qui se passe dans les coulisses

1. Le PVC demande du stockage à la `StorageClass`
2. Lorsque le pod est planifié :
   a. Le mode automatique EKS provisionne un volume EBS
   b. Crée un PersistentVolume
   c. Attache le volume au nœud
3. Le pod monte le volume et commence à écrire les horodatages

Contrôleur d'instantané

Le mode automatique EKS est compatible avec le Kubernetes CSI Snapshotter, également appelé contrôleur d'instantané. Cependant, le mode automatique EKS n'inclut pas le contrôleur d'instantané. Vous êtes responsable de l'installation et de la configuration du contrôleur d'instantané. Pour de plus amples informations, consultez the section called "Contrôleur d'instantané CSI".

Veuillez consulter la classe `VolumeSnapshotClass` suivante qui fait référence à la capacité de stockage du mode automatique EKS.

```
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotClass
metadata:
  name: auto-ebs-vsclass
driver: ebs.csi.eks.amazonaws.com
deletionPolicy: Delete
```

En savoir plus sur le contrôleur de snapshots CSI Kubernetes.

Déploiement d'une charge de travail accélérée

Ce didacticiel montre comment le mode automatique d'Amazon EKS simplifie le lancement de charges de travail accélérées par le matériel. Le mode automatique Amazon EKS simplifie les opérations au-delà du cluster lui-même en automatisant les composants clés de l'infrastructure qui fournissent des capacités de calcul, de mise en réseau, d'équilibrage de charge, de stockage et de gestion des identités et des accès dès leur installation.

Le mode automatique Amazon EKS inclut les pilotes et les plug-ins de périphérique requis pour certains types d'instances, tels que les pilotes NVIDIA et AWS Neuron. Vous n'avez pas à installer ni à mettre à jour ces composants.

Le mode automatique EKS gère automatiquement les pilotes pour les accélérateurs suivants :

- AWS Trainium
- AWS Inférentie
- Instances EC2 accélérées NVIDIA GPUs sur Amazon

> **Note**
>
> Le mode automatique EKS inclut le plug-in de périphérique NVIDIA pour Kubernetes. Ce plug-in s'exécute automatiquement et n'est pas visible comme un ensemble de démons dans votre cluster.

Prise en charge du réseautage supplémentaire :

- Elastic Fabric Adapter (EFA)

Le mode automatique Amazon EKS élimine les tâches fastidieuses liées à la gestion des pilotes d'accélérateurs et des plug-ins de périphériques.

Vous pouvez également bénéficier d'économies, car le cluster peut être mis à l'échelle jusqu'à zéro. Il est possible de configurer le mode automatique EKS pour arrêter les instances lorsqu'aucune charge de travail n'est en cours d'exécution. Cela est utile pour les charges de travail d'inférence basées sur des lots.

La section suivante présente un exemple de lancement de charges de travail accélérées avec le mode automatique Amazon EKS.

Conditions préalables

- Un cluster Kubernetes avec le mode automatique Amazon EKS configuré.
- Une classe de nœuds `default` EKS est créée lorsque les groupes de nœuds gérés `general-purpose` ou `system` sont activés.

Étape 1 : déployer une charge de travail GPU

Dans cet exemple, vous allez créer un NodePool pour les charges de travail basées sur NVIDIA qui nécessitent 45 Go de mémoire GPU. Avec le mode automatique EKS, vous utilisez les contraintes de planification Kubernetes pour définir vos exigences d'instance.

Pour déployer le mode automatique Amazon EKS `NodePool` et l'exemple`workload`, consultez les informations suivantes NodePool et la définition du pod, puis enregistrez-les sous `nodepool-gpu.yaml` et `pod.yaml` :

nodepool-gpu.yaml

```
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: gpu
spec:
  disruption:
    budgets:
    - nodes: 10%
    consolidateAfter: 1h
    consolidationPolicy: WhenEmpty
  template:
```

```
    metadata: {}
    spec:
      nodeClassRef:
        group: eks.amazonaws.com
        kind: NodeClass
        name: default
      requirements:
        - key: "karpenter.sh/capacity-type"
          operator: In
          values: ["on-demand"]
        - key: "kubernetes.io/arch"
          operator: In
          values: ["amd64"]
        - key: "eks.amazonaws.com/instance-family"
          operator: In
          values:
          - g6e
          - g6
      taints:
        - key: nvidia.com/gpu
          effect: NoSchedule
      terminationGracePeriod: 24h0m0s
```

pod.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: nvidia-smi
spec:
  nodeSelector:
    eks.amazonaws.com/compute-type: auto
  restartPolicy: OnFailure
  containers:
  - name: nvidia-smi
    image: public.ecr.aws/amazonlinux/amazonlinux:2023-minimal
    args:
    - "nvidia-smi"
    resources:
      requests:
        memory: "30Gi"
        cpu: "3500m"
        nvidia.com/gpu: 1
```

```
      limits:
        memory: "30Gi"
        nvidia.com/gpu: 1
  tolerations:
  - key: nvidia.com/gpu
    effect: NoSchedule
    operator: Exists
```

Notez que le sélecteur `eks.amazonaws.com/compute-type: auto` nécessite que la charge de travail soit déployée sur un nœud du mode automatique Amazon EKS. Cela crée NodePool également une odeur qui permet uniquement de planifier des pods avec des tolérances GPUs pour Nvidia.

Appliquez la charge de travail NodePool et à votre cluster.

```
kubectl apply -f nodepool-gpu.yaml
kubectl apply -f pod.yaml
```

Vous devriez voir la sortie suivante :

```
nodepool.karpenter.sh/gpu configured created
pod/nvidia-smi created
```

Attendez quelques secondes, puis vérifiez les nœuds de votre cluster. Vous devriez voir un nouveau nœud provisionné dans votre cluster du mode automatique Amazon EKS :

```
> kubectl get nodes

NAME        TYPE          CAPACITY    ZONE         NODE                   READY   AGE
gpu-dnknr   g6e.2xlarge   on-demand   us-west-2b   i-02315c7d7643cdee6   True    76s
```

Étape 2 : valider

Vous pouvez constater que le mode automatique Amazon EKS a lancé un `g6e.2xlarge` plutôt qu'un `g6.2xlarge` autonome, car la charge de travail nécessitait une instance avec un GPU L40S, conformément aux contraintes de planification Kubernetes suivantes :

```
...
  nodeSelector:
```

```
    eks.amazonaws.com/instance-gpu-name: l40s
...
    requests:
        memory: "30Gi"
        cpu: "3500m"
        nvidia.com/gpu: 1
      limits:
        memory: "30Gi"
        nvidia.com/gpu: 1
```

Maintenant, consultez les journaux des conteneurs en exécutant la commande suivante :

```
kubectl logs nvidia-smi
```

Exemple de sortie :

```
+-----------------------------------------------------------------------------------------
+
| NVIDIA-SMI 535.230.02             Driver Version: 535.230.02   CUDA Version: 12.2
 |
|-----------------------------------------+----------------------
+----------------------+
| GPU  Name                 Persistence-M | Bus-Id        Disp.A | Volatile Uncorr. ECC
 |
| Fan  Temp   Perf          Pwr:Usage/Cap |         Memory-Usage | GPU-Util  Compute M.
 |
|                                         |                      |               MIG M.
 |
|=========================================+======================
+======================|
|   0  NVIDIA L40S                    On  | 00000000:30:00.0 Off |                    0
 |
| N/A   27C    P8              23W / 350W |      0MiB / 46068MiB |      0%      Default
 |
|                                         |                      |                  N/A
 |
+-----------------------------------------+----------------------
+----------------------+

+-----------------------------------------------------------------------------------------
+
| Processes:
 |
```

```
|  GPU   GI   CI        PID   Type   Process name                            GPU Memory
 |
|        ID   ID                                                             Usage
 |
|
=============================================================================================|
|  No running processes found
 |
+---------------------------------------------------------------------------------------------
+
```

Vous pouvez constater que le conteneur a détecté qu'il s'exécute sur une instance dotée d'un GPU `NVIDIA` et que vous n'avez pas eu à installer de pilote de périphérique, car cela est géré par le mode automatique Amazon EKS.

Étape 3 : nettoyer

Pour supprimer tous les objets créés, utilisez `kubectl` pour supprimer l'exemple de déploiement NodePool afin de mettre fin au nœud :

```
kubectl delete -f nodepool-gpu.yaml
kubectl delete -f pod.yaml
```

Exemple de NodePools référence

Créez un NVIDIA NodePool

Les définitions suivantes sont NodePool les suivantes :

- Lancement uniquement d'instances des familles `g6e` et `g6`
- Consolidation des nœuds lorsqu'ils restent vides pendant 1 heure
  - La valeur de 1 heure pour `consolodateAfter` prend en charge des charges de travail irrégulières et réduit le roulement des nœuds. Vous pouvez ajuster `consolidateAfter` selon les exigences de votre charge de travail.

Exemple NodePool de famille d'instances GPU et de consolidation

```
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
```

```
  name: gpu
spec:
  disruption:
    budgets:
    - nodes: 10%
    consolidateAfter: 1h
    consolidationPolicy: WhenEmpty
  template:
    metadata: {}
    spec:
      nodeClassRef:
        group: eks.amazonaws.com
        kind: NodeClass
        name: default
      requirements:
        - key: "karpenter.sh/capacity-type"
          operator: In
          values: ["on-demand"]
        - key: "kubernetes.io/arch"
          operator: In
          values: ["amd64"]
        - key: "eks.amazonaws.com/instance-family"
          operator: In
          values:
          - g6e
          - g6
      terminationGracePeriod: 24h0m0s
```

Au lieu de définir directement le paramètre `eks.amazonaws.com/instance-gpu-name`, vous pouvez utiliser `eks.amazonaws.com/instance-family` pour spécifier la famille d'instances. Pour consulter d'autres étiquettes courantes influençant la planification, consultez the section called "Étiquettes prises en charge par le mode automatique EKS".

Si vous avez des besoins de stockage spécifiques, vous pouvez ajuster le stockage `iops` éphémère des nœuds `size` et `throughput` en créant le vôtre NodeClassà référencer dans le. NodePool En savoir plus sur les NodeClass options configurables.

Exemple de configuration de stockage pour NodeClass

```
apiVersion: eks.amazonaws.com/v1
kind: NodeClass
metadata:
  name: gpu
```

```
spec:
  ephemeralStorage:
    iops: 3000
    size: 80Gi
    throughput: 125
```

Définition d'un AWS trainium et AWS d'une inférence NodePool

Ce qui suit NodePool contient un `eks.amazonaws.com/instance-category` ensemble qui indique de ne lancer que les instances de la famille Inferentia et Trainium :

```
        - key: "eks.amazonaws.com/instance-category"
          operator: In
          values:
            - inf
            - trn
```

Configurer les paramètres du mode automatique EKS

Ce chapitre décrit comment configurer certains aspects spécifiques de vos clusters du mode automatique Amazon Elastic Kubernetes Service (EKS). Bien que le mode automatique EKS gère automatiquement la plupart des composants de l'infrastructure, vous pouvez personnaliser certaines fonctionnalités pour répondre aux exigences de votre charge de travail.

À l'aide des options de configuration décrites dans cette rubrique, vous pouvez modifier les paramètres réseau, les ressources de calcul et les comportements d'équilibrage de charge tout en conservant les avantages de la gestion automatisée de l'infrastructure. Avant d'apporter des modifications à la configuration, examinez les options disponibles dans les sections suivantes afin de déterminer l'approche la mieux adaptée à vos besoins.

Quelles fonctionnalités voulez-vous configure r ?	Option de configuration
Réseau et stockage des nœuds • Configurer l'emplacement des nœuds sur les sous-réseaux publics et privés • Définir des groupes de sécurité personnal isés pour le contrôle d'accès aux nœuds	the section called "Création d'une classe de nœuds"

Quelles fonctionnalités voulez-vous configure r ?	Option de configuration
• Personnaliser les politiques de traduction d'adresses réseau (SNAT) • Activer les politiques réseau Kubernetes, la journalisation et la surveillance détaillées des politiques réseau • Définir les paramètres de stockage éphémère (taille, IOPS, débit) • Configurer le stockage éphémère chiffré avec des clés KMS personnalisées • Isoler le trafic des pods dans des sous-rése aux distincts des nœuds	
Ressources de calcul des nœuds • Sélectionnez des types et des familles d' EC2 instances spécifiques • Définition des architectures de processeur (x86_64,) ARM64 • Configurer les types de capacité (à la demande, spot) • Spécifier les zones de disponibilité • Configurer les rejets et les étiquettes des nœuds • Définir les limites de ressources pour l'utilisa tion du processeur et de la mémoire	the section called "Création d'un groupe de nœuds"

Quelles fonctionnalités voulez-vous configure r ?	Option de configuration
Pools de nœuds à capacité statique • Maintenez un nombre fixe de nœuds, quelle que soit la demande en modules • Configurer une capacité prévisible pour les instances réservées • Définissez un nombre spécifique de répliques pour un encombrement constant de l'infrast ructure • Redimensionnez manuellement les pools de nœuds statiques • Surveiller l'état du pool de nœuds à capacité statique	the section called "Pools de nœuds à capacité statique"
Paramètres de l'Application Load Balancer • Déployer des équilibreurs de charge internes ou connectés à Internet • Configurer la répartition de charge entre zones • Définir les délais d'inactivité • Activer HTTP/2 et le support WebSocket • Configurer les paramètres de surveillance de l'état • Spécifier les paramètres du certificat TLS • Définir les attributs du groupe cible • Définir le type d'adresse IP (IPv4double pile)	the section called "Création d'une classe de ressources Ingress"

Quelles fonctionnalités voulez-vous configure r ?	Option de configuration
Paramètres du Network Load Balancer • Configurer le routage IP direct des pods • Activer la répartition de charge entre zones • Définir le délai d'inactivité de connexion • Configurer les paramètres de surveillance de l'état • Spécifier l'emplacement du sous-réseau • Définir le type d'adresse IP (IPv4double pile) • Configurer la préservation de l'adresse IP source du client • Définir les attributs du groupe cible	the section called "Créer un service"
Paramètres de la classe de stockage • Définir les types de volumes EBS (gp3, io1, io2, etc.) • Configurer le chiffrement des volumes et l'utilisation des clés KMS • Définir les paramètres IOPS et de débit • Définir comme classe de stockage par défaut • Définir des balises personnalisées pour les volumes provisionnés	the section called "Créez StorageClass"

Quelles fonctionnalités voulez-vous configure r ?	Option de configuration
Contrôle de l'utilisation de l'ODCR • Configurer le déploiement des charges de travail dans les réservations de capacité EC2 à la demande • Sélectionnez explicitement un ODCRs identifiant spécifique pour une utilisation ciblée de la capacité • Utilisez la sélection basée sur des balises pour cibler les groupes de ODCRs • Filtrer ODCRs par AWS compte propriétaire pour les scénarios entre comptes • Contrôlez si les charges de travail utilisent automatiquement Open ODCRs	the section called "Contrôle des réserves de capacité"
Sécurité avancée des nœuds • Activer les modules cryptographiques validés par la norme FIPS 140-2 • Configurer le mode de sécurité du verrouill age du noyau	the section called "Sécurité avancée"

Création d'une classe de nœuds pour Amazon EKS

Les classes de nœuds Amazon EKS sont des modèles qui offrent un contrôle précis de la configuration de vos nœuds gérés par le mode automatique EKS. Une classe de nœuds définit les paramètres d'infrastructure appliqués à des groupes de nœuds de votre cluster EKS, notamment la configuration réseau, les paramètres de stockage et le balisage des ressources. Cette rubrique explique comment créer et configurer une classe de nœuds pour répondre à vos exigences opérationnelles spécifiques.

Lorsque vous devez personnaliser la façon dont le mode automatique EKS provisionne et configure les instances EC2 au-delà des paramètres par défaut, la création d'une classe de nœuds vous

permet de contrôler précisément des paramètres d'infrastructure essentiels. Par exemple, vous pouvez spécifier le placement dans des sous-réseaux privés pour renforcer la sécurité, configurer un stockage éphémère des instances pour les charges de travail sensibles aux performances ou appliquer des étiquettes personnalisées pour l'allocation des coûts.

Création d'une classe de nœuds

Pour créer une `NodeClass`, procédez comme suit :

1. Créez un fichier YAML (par exemple `nodeclass.yaml`) contenant la configuration de votre classe de nœuds
2. Appliquez la configuration à votre cluster à l'aide de `kubectl`
3. Référencez la classe de nœuds dans la configuration de votre groupe de nœuds. Pour de plus amples informations, veuillez consulter the section called "Création d'un groupe de nœuds".

Vous devez avoir installé et configuré `kubectl`. Pour de plus amples informations, veuillez consulter Configuration.

Exemple de classe de nœuds simple

Voici un exemple de classe de nœuds :

```
apiVersion: eks.amazonaws.com/v1
kind: NodeClass
metadata:
  name: private-compute
spec:
  subnetSelectorTerms:
    - tags:
        Name: "private-subnet"
        kubernetes.io/role/internal-elb: "1"
  securityGroupSelectorTerms:
    - tags:
        Name: "eks-cluster-sg"
  ephemeralStorage:
    size: "160Gi"
```

Cela NodeClass augmente la quantité de stockage éphémère sur le nœud.

Appliquez cette configuration en utilisant :

```
kubectl apply -f nodeclass.yaml
```

Ensuite, référencez la classe de nœuds dans la configuration de votre groupe de nœuds. Pour de plus amples informations, veuillez consulter the section called "Création d'un groupe de nœuds".

Création d'une entrée d'accès pour la classe de nœuds

Si vous créez une classe de nœuds personnalisée, vous devez créer une entrée d'accès EKS pour permettre aux nœuds de rejoindre le cluster. EKS crée automatiquement les entrées d'accès lorsque vous utilisez la classe de nœuds intégrée et les groupes de nœuds intégrés.

Pour plus d'informations sur le fonctionnement des entrées d'accès, consultez the section called "Entrées d'accès".

Lorsque vous créez des entrées d'accès pour les classes de nœuds du mode automatique EKS, vous devez utiliser le type d'entrée d'accès `EC2`.

Création d'une entrée d'accès à l'aide de la CLI

Pour créer une entrée d'accès pour les nœuds EC2 et associer la politique de nœud automatique EKS :

Mettez à jour les commandes CLI suivantes avec le nom de votre cluster et l'ARN du rôle de nœud. L'ARN du rôle de nœud est spécifié dans le fichier YAML de la classe de nœuds.

```
# Create the access entry for EC2 nodes
aws eks create-access-entry \
  --cluster-name <cluster-name> \
  --principal-arn <node-role-arn> \
  --type EC2

# Associate the auto node policy
aws eks associate-access-policy \
  --cluster-name <cluster-name> \
  --principal-arn <node-role-arn> \
  --policy-arn arn:aws: eks::aws:cluster-access-policy/AmazonEKSAutoNodePolicy \
  --access-scope type=cluster
```

Créez une entrée d'accès avec CloudFormation

Pour créer une entrée d'accès pour les nœuds EC2 et associer la politique de nœud automatique EKS :

Mettez à jour ce qui suit CloudFormation avec le nom de votre cluster et l'ARN du rôle de nœud. L'ARN du rôle de nœud est spécifié dans le fichier YAML de la classe de nœuds.

```
EKSAutoNodeRoleAccessEntry:
  Type: AWS::EKS::AccessEntry
  Properties:
    ClusterName: <cluster-name>
    PrincipalArn: <node-role-arn>
    Type: "EC2"
    AccessPolicies:
      - AccessScope:
          Type: cluster
        PolicyArn: arn:aws: eks::aws:cluster-access-policy/AmazonEKSAutoNodePolicy
  DependsOn: [ <cluster-name> ] # previously defined in CloudFormation
```

Pour plus d'informations sur le déploiement CloudFormation de stacks, voir Getting started with CloudFormation

Spécification de la classe de nœuds

```
apiVersion: eks.amazonaws.com/v1
kind: NodeClass
metadata:
  name: my-node-class
spec:
  # Required fields

  # role and instanceProfile are mutually exclusive fields.
  role: MyNodeRole  # IAM role for EC2 instances
  # instanceProfile: eks-MyNodeInstanceProfile  # IAM instance-profile for EC2
 instances

  subnetSelectorTerms:
    - tags:
        Name: "private-subnet"
        kubernetes.io/role/internal-elb: "1"
    # Alternative using direct subnet ID
    # - id: "subnet-0123456789abcdef0"

  securityGroupSelectorTerms:
    - tags:
        Name: "eks-cluster-sg"
    # Alternative approaches:
```

```
    # - id: "sg-0123456789abcdef0"
    # - name: "eks-cluster-security-group"

  # Optional: Pod subnet selector for advanced networking
  podSubnetSelectorTerms:
    - tags:
        Name: "pod-subnet"
        kubernetes.io/role/pod: "1"
    # Alternative using direct subnet ID
    # - id: "subnet-0987654321fedcba0"
# must include Pod security group selector also
 podSecurityGroupSelectorTerms:
  - tags:
      Name: "eks-pod-sg"
    # Alternative using direct security group ID
    # - id: "sg-0123456789abcdef0"

  # Optional: Selects on-demand capacity reservations and capacity blocks
  # for EKS Auto Mode to prioritize.
  capacityReservationSelectorTerms:
    - id: cr-56fac701cc1951b03
    # Alternative Approaches
    - tags:
        Name: "targeted-odcr"
      # Optional owning account ID filter
      owner: "012345678901"

  # Optional fields
  snatPolicy: Random  # or Disabled

  networkPolicy: DefaultAllow  # or DefaultDeny
  networkPolicyEventLogs: Disabled  # or Enabled

  ephemeralStorage:
    size: "80Gi"    # Range: 1-59000Gi or 1-64000G or 1-58Ti or 1-64T
    iops: 3000      # Range: 3000-16000
    throughput: 125 # Range: 125-1000
    # Optional KMS key for encryption
    kmsKeyID: "arn:aws: kms:region:account:key/key-id"
    # Accepted formats:
    # KMS Key ID
    # KMS Key ARN
    # Key Alias Name
    # Key Alias ARN
```

```
  advancedNetworking:
    # Optional: Controls whether public IP addresses are assigned to instances that are
 launched with the nodeclass.
    # If not set, defaults to the MapPublicIpOnLaunch setting on the subnet.
    associatePublicIPAddress: false

    # Optional: Forward proxy, commonly requires certificateBundles as well
    # for EC2, see https://repost.aws/knowledge-center/eks-http-proxy-containerd-
automation
    httpsProxy: http://192.0.2.4:3128 #commonly port 3128 (Squid) or 8080 (NGINX) #Max
 255 characters
    #httpsProxy: http://[2001:db8::4]:3128 # IPv6 address with port, use []
    noProxy: #Max 50 entries
        - localhost #Max 255 characters each
        - 127.0.0.1
        #- ::1 # IPv6 localhost
        #- 0:0:0:0:0:0:0:1 # IPv6 localhost
        - 169.254.169.254 # EC2 Instance Metadata Service
        #- [fd00:ec2::254] # IPv6 EC2 Instance Metadata Service
        # Domains to exclude, put all VPC endpoints here
        - .internal
        - .eks.amazonaws.com
    # ipv4PrefixSize is default to Auto which is prefix and fallback to secondary IP.
 "32" is the secondary IP mode.
    ipv4PrefixSize: Auto # or "32"

    # enableV4Egress is default to true. Setting it to false when using network policy
 or blocking IPv4 traffic in IPv6 clusters
    enableV4Egress: false

  advancedSecurity:
    # Optional, US regions only: Specifying `fips: true` will cause nodes in the
 nodeclass to run FIPS compatible AMIs.
    fips: false

  # Optional: Custom certificate bundles.
  certificateBundles:
    - name: "custom-cert"
      data: "base64-encoded-cert-data"

  # Optional: Additional EC2 tags (with restrictions)
  tags:
    Environment: "production"
```

```
    Team: "platform"
    # Note: Cannot use restricted tags like:
    # - kubernetes.io/cluster/*
    # - karpenter.sh/provisioner-name
    # - karpenter.sh/nodepool
    # - karpenter.sh/nodeclaim
    # - karpenter.sh/managed-by
    # - eks.amazonaws.com/nodeclass
```

Considérations

- Si vous souhaitez vérifier le volume de stockage local d'une instance, vous pouvez décrire le nœud pour voir la ressource de stockage éphémère.
- Chiffrement du volume : EKS utilise la clé KMS personnalisée configurée pour chiffrer le volume racine en lecture seule de l'instance et le read/write volume de données.
- Remplacement du rôle IAM du nœud : si vous modifiez le rôle IAM du nœud associé à une `NodeClass`, vous devrez créer une nouvelle entrée d'accès. EKS crée automatiquement une entrée d'accès pour le rôle IAM du nœud lors de la création du cluster. Le rôle IAM du nœud nécessite la stratégie d'accès `AmazonEKSAutoNodePolicy` EKS. Pour de plus amples informations, veuillez consulter the section called "Entrées d'accès".
- Densité maximale de pods : EKS limite le nombre maximal de pods sur un nœud à 110. Cette limite s'applique après le calcul existant du nombre maximal de pods. Pour de plus amples informations, veuillez consulter the section called "Types d'instances Amazon EC2".
- Balises : si vous souhaitez propager des balises de Kubernetes vers EC2, vous devez configurer des autorisations IAM supplémentaires. Pour de plus amples informations, veuillez consulter the section called "Identités et accès".
- Classe de nœuds par défaut : ne nommez pas votre classe de nœuds personnalisée `default`. En effet, le mode automatique EKS inclut une `NodeClass` appelée `default`, provisionnée automatiquement lorsque vous activez au moins un `NodePool` intégré. Pour plus d'informations sur l'activation des `NodePools` intégrés, consultez the section called "Vérifier les groupes de nœuds créés".
- Comportement des **`subnetSelectorTerms`** avec plusieurs sous-réseaux : si plusieurs sous-réseaux correspondent aux conditions `subnetSelectorTerms` ou sont fournis par ID, le mode automatique EKS crée des nœuds répartis sur l'ensemble des sous-réseaux.
  - Si les sous-réseaux se trouvent dans des zones de disponibilité différentes (AZs), vous pouvez utiliser les fonctionnalités de Kubernetes telles que les contraintes de propagation de la topologie

des pods et le routage adapté à la topologie pour répartir les pods et le trafic entre les zones, respectivement.

- S'il existe plusieurs sous-réseaux dans la même zone de disponibilité qui correspondent aux `subnetSelectorTerms`, le mode automatique EKS crée des pods sur chaque nœud répartis dans ces sous-réseaux de cette même zone de disponibilité. Le mode automatique EKS crée également des interfaces réseau secondaires sur chaque nœud dans les autres sous-réseaux de la même zone de disponibilité. Il choisit en fonction du nombre d'adresses IP disponibles dans chaque sous-réseau, afin d'utiliser les sous-réseaux plus efficacement. Cependant, vous ne pouvez pas spécifier quel sous-réseau le mode automatique EKS utilise pour chaque pod ; si vous avez besoin que des pods s'exécutent dans des sous-réseaux spécifiques, utilisez the section called "Sélection des sous-réseaux pour les pods".

Sélection des sous-réseaux pour les pods

Les champs `podSubnetSelectorTerms` et `podSecurityGroupSelectorTerms` activent des configurations réseau avancées en permettant aux pods de s'exécuter dans des sous-réseaux différents de ceux des nœuds. Cette séparation offre un meilleur contrôle du routage du trafic réseau et des politiques de sécurité. Notez que les `podSecurityGroupSelectorTerms` sont requises avec les `podSubnetSelectorTerms`.

Cas d'utilisation

Utilisez `podSubnetSelectorTerms` lorsque vous devez :

- Séparez le trafic d'infrastructure (node-to-node communication) du trafic d'applications (Pod-to-Pod communication)
- Appliquer des configurations réseau différentes aux sous-réseaux des nœuds et aux sous-réseaux des pods.
- Mettre en œuvre des politiques de sécurité ou des règles de routage distinctes pour les nœuds et les pods.
- Configurer des serveurs proxy inverses ou des filtres réseau spécifiquement pour le trafic des nœuds sans affecter le trafic des pods. Utiliser `advancedNetworking` et `certificateBundles` pour définir votre serveur proxy inverse ainsi que tout certificat autosigné ou privé pour ce serveur.

Exemple de configuration

```
apiVersion: eks.amazonaws.com/v1
```

```
kind: NodeClass
metadata:
  name: advanced-networking
spec:
  role: MyNodeRole

  # Subnets for EC2 instances (nodes)
  subnetSelectorTerms:
    - tags:
        Name: "node-subnet"
        kubernetes.io/role/internal-elb: "1"

  securityGroupSelectorTerms:
    - tags:
        Name: "eks-cluster-sg"

  # Separate subnets for Pods
  podSubnetSelectorTerms:
    - tags:
        Name: "pod-subnet"
        kubernetes.io/role/pod: "1"

  podSecurityGroupSelectorTerms:
  - tags:
      Name: "eks-pod-sg"
```

Considérations relatives aux sélecteurs de sous-réseaux pour les pods

- Réduction de la densité de pods : moins de pods peuvent s'exécuter sur chaque nœud lorsqu'on utilise `podSubnetSelectorTerms`, car l'interface réseau principale du nœud se trouve dans le sous-réseau du nœud et ne peut pas être utilisée pour les pods dans le sous-réseau des pods.
- Limites des sélecteurs de sous-réseaux : les configurations standard `subnetSelectorTerms` et `securityGroupSelectorTerms` ne s'appliquent pas à la sélection du sous-réseau des pods.
- Planification du réseau : assurez-vous de disposer d'un espace d'adresses IP suffisant dans les sous-réseaux des nœuds et des pods pour répondre aux besoins de vos charges de travail.
- Configuration du routage : vérifiez que la table de routage et les listes de contrôle d'accès (ACL) réseau des sous-réseaux destinés aux pods sont correctement configurées pour permettre la communication entre les sous-réseaux des nœuds et ceux des pods.

- Zones de disponibilité : vérifiez que vous avez créé des sous-réseaux Pod sur plusieurs AZs. Si vous utilisez un sous-réseau spécifique pour les pods, celui-ci doit se trouver dans la même zone de disponibilité que le sous-réseau des nœuds.

Mode IP secondaire pour les pods

Les `ipv4PrefixSize` champs permettent des configurations réseau avancées en autorisant uniquement l'attribution d'adresses IP secondaires aux nœuds. Cette fonctionnalité n'alloue pas de préfixe (/28) aux nœuds et ne conserve qu'une seule adresse IP secondaire comme minimale. IPTarget

Cas d'utilisation

Utilisez `ipv4PrefixSize` lorsque vous devez :

- Utilisation IP réduite : une seule adresse IP sera réchauffée dans chaque nœud.
- Diminution du taux de rotation des capsules : la vitesse de création des capsules n'est pas une préoccupation majeure.
- Aucune fragmentation du préfixe : la fragmentation provoquée par le préfixe est une préoccupation majeure ou un obstacle à l'utilisation du mode automatique.

Exemple de configuration

```
apiVersion: eks.amazonaws.com/v1
kind: NodeClass
metadata:
  name: advanced-networking
spec:
  role: MyNodeRole

  advancedNetworking:
    ipv4PrefixSize: "32"
```

Considérations relatives au mode IP secondaire

- Vitesse de création de pods réduite : étant donné qu'une seule adresse IP secondaire est réchauffée, le service IPAM a besoin de plus de temps pour effectuer la création d'un plus grand IPs nombre de pods.

Désactivez la IPv4 sortie des IPv6 pods en IPv6 clusters.

Les `enableV4Egress` champs sont vrais par défaut. Pour les IPv6 clusters en mode automatique, la fonctionnalité peut être désactivée. Le mode automatique ne créera donc pas d' IPv4 interface de sortie uniquement pour les IPv6 pods. Ceci est important car l'interface IPv4 réseau ne sera pas sécurisée par la fonctionnalité de politique réseau. Les politiques réseau ne seront appliquées que sur l'interface principale du Pod (c'est-à-dire) eth0.

Cas d'utilisation

Utilisez `enableV4Egress` lorsque vous devez :

- Utiliser le IPv6 cluster : le trafic de IPv4 sortie est autorisé par défaut.
- Utiliser la politique réseau : Actuellement, la politique réseau EKS ne prend pas en charge le double stack. La désactivation de V4egress peut protéger le trafic des pods sortant des pods de manière inattendue.

Exemple de configuration

```
apiVersion: eks.amazonaws.com/v1
kind: NodeClass
metadata:
  name: advanced-networking
spec:
  role: MyNodeRole

  advancedNetworking:
    enableV4Egress: false
```

Considérations relatives à la désactivation d'EnableV4egress

- Politique réseau dans le IPv6 cluster : les IPv6 clusters autorisent IPv4 le trafic par défaut. Le réglage `enableV4Egress: false` bloque le IPv4 trafic sortant, offrant ainsi une sécurité renforcée, en particulier lorsqu'il est utilisé avec des politiques réseau.

Create a Node Pool for EKS Auto Mode

Les groupes de nœuds Amazon EKS offrent une méthode flexible pour gérer les ressources de calcul dans votre cluster Kubernetes. Cette rubrique explique comment créer et configurer des groupes de

nœuds à l'aide de Karpenter, un outil de provisionnement de nœuds qui permet d'optimiser l'étendue du cluster et l'utilisation des ressources. Grâce à la ressource NodePool de Karpenter, vous pouvez définir des exigences précises pour vos ressources de calcul, notamment les types d'instances, les zones de disponibilité, les architectures et les types de capacité.

Les groupes de nœuds intégrés `system` et `general-purpose` ne peuvent pas être modifiés. Vous pouvez uniquement les activer ou les désactiver. Pour de plus amples informations, consultez the section called "Vérifier les groupes de nœuds créés".

La spécification NodePool permet un contrôle précis des ressources de cluster computing EKS à l'aide de diverses étiquettes et exigences prises en charge. Ces options incluent la définition des catégories d'instances EC2, la configuration du processeur, la sélection des zones de disponibilité, le choix de l'architecture (ARM64 ou AMD64) et le type de capacité (Spot ou à la demande). Vous pouvez également définir des limites de ressources pour l'utilisation du processeur et de la mémoire, afin que votre cluster respecte les contraintes opérationnelles requises.

Le mode automatique EKS utilise des étiquettes Kubernetes reconnues afin de proposer des méthodes uniformes et standardisées d'identification des caractéristiques des nœuds. Ces étiquettes, par exemple `topology.kubernetes.io/zone` pour les zones de disponibilité et `kubernetes.io/arch` pour l'architecture du processeur, respectent les conventions établies de Kubernetes. De plus, des étiquettes spécifiques à EKS (préfixées par `eks.amazonaws.com/`) étendent cette fonctionnalité en ajoutant des attributs spécifiques à AWS, tels que les types d'instance, les fabricants de processeurs, les capacités GPU et les spécifications réseau. Ce système d'étiquetage standardisé assure une intégration fluide avec les outils Kubernetes existants, tout en offrant une intégration approfondie à l'infrastructure AWS.

Création d'un NodePool

Pour créer un NodePool pour votre cluster Amazon EKS, procédez comme suit :

1. Créez un fichier YAML nommé `nodepool.yaml` contenant la configuration requise de votre NodePool. Vous pouvez utiliser l'exemple de configuration ci-dessous.
2. Appliquez le NodePool à votre cluster :

```
kubectl apply -f nodepool.yaml
```

3. Vérifiez que le NodePool a été créé avec succès :

```
kubectl get nodepools
```

4. (Facultatif) Surveillez l'état du NodePool :

```
kubectl describe nodepool default
```

Assurez-vous que votre NodePool fait référence à un NodeClass valide existant dans votre cluster. La NodeClass définit les configurations spécifiques à AWS pour vos ressources de calcul. Pour de plus amples informations, consultez the section called "Création d'une classe de nœuds".

Exemple de NodePool

```
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: my-node-pool
spec:
  template:
    metadata:
      labels:
        billing-team: my-team
    spec:
      nodeClassRef:
        group: eks.amazonaws.com
        kind: NodeClass
        name: default

      requirements:
        - key: "eks.amazonaws.com/instance-category"
          operator: In
          values: ["c", "m", "r"]
        - key: "eks.amazonaws.com/instance-cpu"
          operator: In
          values: ["4", "8", "16", "32"]
        - key: "topology.kubernetes.io/zone"
          operator: In
          values: ["us-west-2a", "us-west-2b"]
        - key: "kubernetes.io/arch"
          operator: In
          values: ["arm64", "amd64"]

  limits:
    cpu: "1000"
```

```
    memory: 1000Gi
```

Étiquettes prises en charge par le mode automatique EKS

Le mode automatique EKS prend en charge les étiquettes bien connues suivantes.

> **Note**
>
> Le mode automatique EKS utilise des étiquettes différentes de celles de Karpenter. Les étiquettes associées aux instances gérées EC2 commencent par `eks.amazonaws.com`.

Étiquette	exemple	Description
topology.kubernetes.io/zone	us-east-2a	AWS région
node.kubernetes.io/instance-type	g4dn.8xlarge	Type d'instance AWS
kubernetes.io/arch	amd64	Architectures définies par les valeurs GOARCH sur l'instance
karpenter.sh/capacity-type	spot	Les types de capacité comprennent `spot`, `on-demand`
eks.amazonaws.com/instance-hypervisor	nitro	Types d'instance utilisant un hyperviseur spécifique
eks.amazonaws.com/compute-type	auto	Identifie les nœuds gérés par le mode automatique EKS
eks.amazonaws.com/instance-encryption-in-transit-supported	true	Types d'instance prenant en charge (ou non) le chiffrement en transit
eks.amazonaws.com/instance-category	g	Types d'instance de la même catégorie, généralement la chaîne précédant le numéro de génération

Étiquette	exemple	Description
eks.amazonaws.com/instance-generation	4	Numéro de génération du type d'instance au sein d'une même catégorie d'instances
eks.amazonaws.com/instance-family	g4dn	Types d'instance présentant des propriétés similaires, mais des quantités de ressources différentes
eks.amazonaws.com/instance-size	8xlarge	Types d'instance présentant des quantités de ressources similaires, mais des propriétés différentes
eks.amazonaws.com/instance-cpu	32	Nombre de processeurs sur l'instance
eks.amazonaws.com/instance-cpu-manufacturer	`aws`	Nom du fabricant du processeur
eks.amazonaws.com/instance-memory	131072	Nombre de mébioctets de mémoire sur l'instance
eks.amazonaws.com/instance-ebs-bandwidth	9500	Nombre maximal de mégabits d'EBS disponibles sur l'instance
eks.amazonaws.com/instance-network-bandwidth	131072	Nombre de mégabits de base disponibles sur l'instance
eks.amazonaws.com/instance-gpu-name	t4	Nom du GPU présent sur l'instance, le cas échéant
eks.amazonaws.com/instance-gpu-manufacturer	nvidia	Nom du fabricant du GPU
eks.amazonaws.com/instance-gpu-count	1	Nombre de GPU présents sur l'instance
eks.amazonaws.com/instance-gpu-memory	16384	Quantité de mémoire GPU en mébioctets sur l'instance

Étiquette	exemple	Description
eks.amazonaws.com/instance-local-nvme	900	Quantité de stockage local NVMe en gibioctets sur l'instance

> **Note**
>
> Le mode automatique EKS ne prend en charge que certains types d'instances et impose des exigences de taille minimale. Pour de plus amples informations, consultez the section called "Référence des instances prises en charge par le mode automatique EKS".

Étiquettes non prises en charge par le mode automatique EKS

Le mode automatique EKS ne prend pas en charge les étiquettes suivantes.

- Le mode automatique EKS ne prend en charge que Linux
  - `node.kubernetes.io/windows-build`
  - `kubernetes.io/os`

Désactivation des groupes de nœuds intégrés

Si vous créez des groupes de nœuds personnalisés, vous pouvez désactiver les groupes de nœuds intégrés. Pour de plus amples informations, consultez the section called "Vérifier les groupes de nœuds créés".

Cluster sans groupes de nœuds intégrés

Vous pouvez créer un cluster sans groupes de nœuds intégrés. Cela est utile si votre organisation a mis en place des groupes de nœuds personnalisés.

> **Note**
>
> Lorsque vous créez un cluster sans groupes de nœuds intégrés, la NodeClass `default` n'est pas provisionnée automatiquement. Vous devrez créer une NodeClass personnalisée. Pour de plus amples informations, consultez the section called "Création d'une classe de nœuds".

Présentation :

1. Créez un cluster EKS avec les valeurs `nodePools` et `nodeRoleArn` laissées vides.
   - Exemple eksctl `autoModeConfig` :

     ```
     autoModeConfig:
       enabled: true
       nodePools: []
       # Do not set a nodeRoleARN
     ```

     Pour plus d'informations, consultez the section called "CLI eksctl".
2. Créez une NodeClass personnalisée avec un ARN de rôle de nœud
   - Pour plus d'informations, consultez the section called "Création d'une classe de nœuds".
3. Créez une entrée d'accès pour la classe de nœuds personnalisée
   - Pour plus d'informations, consultez the section called "Création d'une entrée d'accès pour la classe de nœuds".
4. Créez un groupe de nœuds personnalisé, comme décrit ci-dessus.

Interruption

Vous pouvez configurer le mode automatique EKS afin de gérer l'interruption des nœuds à partir de votre NodePool de plusieurs façons. Vous pouvez utiliser `spec.disruption.consolidationPolicy`, `spec.disruption.consolidateAfter` ou `spec.template.spec.expireAfter`. Vous pouvez également limiter le taux d'interruption du mode automatique EKS à l'aide du paramètre `spec.disruption.budgets` du NodePool. Vous pouvez également contrôler les fenêtres temporelles et le nombre de nœuds interrompus simultanément. Pour obtenir des instructions sur la configuration de ce comportement, consultez la section Interruption dans la documentation de Karpenter.

Vous pouvez configurer l'interruption pour les groupes de nœuds afin de :

- Identifier les instances sous-utilisées et consolider les charges de travail.
- Créer un budget d'interruption pour limiter la fréquence des arrêts de nœuds liés à la dérive, au vidage ou à la consolidation.

Par défaut, le mode automatique EKS :

- Consolide les instances sous-utilisées.
- Met fin aux instances au bout de 336 heures.
- Définit un budget d'interruption unique correspondant à 10 % des nœuds.
- Permet le remplacement des nœuds en cas de dérive lors de la publication d'une nouvelle AMI du mode automatique (environ une fois par semaine).

Délai de grâce avant terminaison

Lorsque le `terminationGracePeriod` n'est pas explicitement définie dans une NodePool EKS Auto, le système applique automatiquement un délai de grâce de 24 heures à la NodeClaim associée. Même si les utilisateurs d'EKS Auto ne verront pas un `terminationGracePeriod` défini par défaut dans leurs configurations personnalisées de NodePool, ils pourront l'observer dans la NodeClaim. La fonctionnalité demeure identique, qu'elle soit définie explicitement dans la NodePool ou appliquée par défaut dans la NodeClaim, ce qui garantit un comportement prévisible lors de la terminaison des nœuds dans le cluster.

Pools de nœuds à capacité statique en mode automatique EKS

Le mode automatique Amazon EKS prend en charge les pools de nœuds à capacité statique qui maintiennent un nombre fixe de nœuds, quelle que soit la demande de pods. Les pools de nœuds à capacité statique sont utiles pour les charges de travail qui nécessitent une capacité prévisible, des instances réservées ou des exigences de conformité spécifiques lorsque vous devez maintenir une empreinte d'infrastructure cohérente.

Contrairement aux pools de nœuds dynamiques qui évoluent en fonction des demandes de planification des pods, les pools de nœuds à capacité statique maintiennent le nombre de nœuds que vous avez configurés.

Exemple de base

Voici un pool de nœuds simple à capacité statique qui gère 5 nœuds :

```
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: my-static-nodepool
spec:
  replicas: 5  # Maintain exactly 5 nodes
```

```
  template:
    spec:
      nodeClassRef:
        group: eks.amazonaws.com
        kind: NodeClass
        name: default

      requirements:
        - key: "eks.amazonaws.com/instance-category"
          operator: In
          values: ["m", "c"]
        - key: "topology.kubernetes.io/zone"
          operator: In
          values: ["us-west-2a", "us-west-2b"]

  limits:
    nodes: 8
```

Configuration d'un pool de nœuds à capacité statique

Pour créer un pool de nœuds à capacité statique, définissez le `replicas` champ dans votre spécification. NodePool Le `replicas` champ définit le nombre exact de nœuds que le pool de nœuds conservera.

Contraintes liées au pool de nœuds à capacité statique

Les pools de nœuds à capacité statique présentent plusieurs contraintes et comportements importants :

Contraintes de configuration :

- Impossible de changer de mode : une fois que vous avez configuré `replicas` un pool de nœuds, vous ne pouvez pas le supprimer. Le pool de nœuds ne peut pas passer du mode statique au mode dynamique.
- Limites de ressources limitées : seul le `limits.nodes` champ est pris en charge dans la section des limites. Les limites de processeur et de mémoire ne sont pas applicables.
- Aucun champ de pondération : le `weight` champ ne peut pas être défini sur les pools de nœuds à capacité statique car la sélection des nœuds n'est pas basée sur la priorité.

Comportement opérationnel :

- Aucune consolidation : les nœuds des pools à capacité statique ne sont pas pris en compte pour la consolidation en fonction de leur utilisation.
- Opérations de dimensionnement : les opérations de dimensionnement évitent les budgets liés aux interruptions de service des nœuds tout en les respectant PodDisruptionBudgets.
- Remplacement des nœuds : les nœuds sont toujours remplacés pour cause de dérive (comme les mises à jour d'AMI) et d'expiration en fonction de votre configuration.

Dimensionner un pool de nœuds à capacité statique

Vous pouvez modifier le nombre de répliques dans un pool de nœuds à capacité statique à l'aide de la commande suivante : `kubectl scale`

```
# Scale down to 5 nodes
kubectl scale nodepool static-nodepool --replicas=5
```

Lors de la réduction de la taille, le mode automatique d'EKS met fin aux nœuds de manière harmonieuse, en respectant PodDisruptionBudgets et en autorisant la replanification des pods en cours d'exécution vers les nœuds restants.

Surveillez les pools de nœuds à capacité statique

Utilisez les commandes suivantes pour surveiller vos pools de nœuds à capacité statique :

```
# View node pool status
kubectl get nodepool static-nodepool

# Get detailed information including current node count
kubectl describe nodepool static-nodepool

# Check the current number of nodes
kubectl get nodepool static-nodepool -o jsonpath='{.status.nodes}'
```

Le `status.nodes` champ indique le nombre actuel de nœuds gérés par le pool de nœuds, qui devrait correspondre au `replicas` nombre souhaité dans des conditions normales.

Exemples de configuration

Pool de nœuds à capacité statique de base

```
apiVersion: karpenter.sh/v1
```

```
kind: NodePool
metadata:
  name: basic-static
spec:
  replicas: 5

  template:
    spec:
      nodeClassRef:
        group: eks.amazonaws.com
        kind: NodeClass
        name: default

      requirements:
        - key: "eks.amazonaws.com/instance-category"
          operator: In
          values: ["m"]
        - key: "topology.kubernetes.io/zone"
          operator: In
          values: ["us-west-2a"]

  limits:
    nodes: 8  # Allow scaling up to 8 during operations
```

Capacité statique avec des types d'instances spécifiques

```
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: reserved-instances
spec:
  replicas: 20

  template:
    metadata:
      labels:
        instance-type: reserved
        cost-center: production
    spec:
      nodeClassRef:
        group: eks.amazonaws.com
        kind: NodeClass
        name: default
```

```
      requirements:
        - key: "node.kubernetes.io/instance-type"
          operator: In
          values: ["m5.2xlarge"]  # Specific instance type
        - key: "karpenter.sh/capacity-type"
          operator: In
          values: ["on-demand"]
        - key: "topology.kubernetes.io/zone"
          operator: In
          values: ["us-west-2a", "us-west-2b", "us-west-2c"]

  limits:
    nodes: 25

  disruption:
    # Conservative disruption for production workloads
    budgets:
      - nodes: 10%
```

Pool de nœuds à capacité statique multizone

```
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: multi-zone-static
spec:
  replicas: 12  # Will be distributed across specified zones

  template:
    metadata:
      labels:
        availability: high
    spec:
      nodeClassRef:
        group: eks.amazonaws.com
        kind: NodeClass
        name: default

      requirements:
        - key: "eks.amazonaws.com/instance-category"
          operator: In
          values: ["c", "m"]
```

```
        - key: "eks.amazonaws.com/instance-cpu"
          operator: In
          values: ["8", "16"]
        - key: "topology.kubernetes.io/zone"
          operator: In
          values: ["us-west-2a", "us-west-2b", "us-west-2c"]
        - key: "karpenter.sh/capacity-type"
          operator: In
          values: ["on-demand"]

  limits:
    nodes: 15

  disruption:
    budgets:
      - nodes: 25%
```

Bonnes pratiques

Planification des capacités :

- Définissez `limits.nodes` une valeur supérieure `replicas` à pour permettre une mise à l'échelle temporaire lors des opérations de remplacement de nœuds.
- Tenez compte de la capacité maximale requise lors de la dérive des nœuds ou des mises à jour de l'AMI lors de la définition des limites.

Sélection de l'instance :

- Utilisez des types d'instances spécifiques lorsque vous avez des instances réservées ou des exigences matérielles spécifiques.
- Évitez les exigences trop restrictives susceptibles de limiter la disponibilité des instances lors du dimensionnement.

Gestion des perturbations :

- Configurez les budgets d'interruption appropriés pour équilibrer la disponibilité avec les opérations de maintenance.
- Tenez compte de la tolérance de votre application en matière de remplacement de nœuds lors de la définition des pourcentages de budget.

Surveillance :

- Surveillez régulièrement le `status.nodes` terrain pour vous assurer que la capacité souhaitée est maintenue.
- Configurez des alertes lorsque le nombre réel de nœuds s'écarte des répliques souhaitées.

Répartition des zones :

- Pour une haute disponibilité, répartissez la capacité statique sur plusieurs zones de disponibilité.
- Lorsque vous créez un pool de nœuds à capacité statique qui couvre plusieurs zones de disponibilité, le mode automatique EKS distribue les nœuds dans les zones spécifiées, mais il n'est pas garanti que la distribution soit uniforme.
- Pour une distribution prévisible et uniforme entre les zones de disponibilité, créez des pools de nœuds distincts à capacité statique, chacun étant épinglé à une zone de disponibilité spécifique conformément aux exigences. `topology.kubernetes.io/zone`
- Si vous avez besoin de 12 nœuds répartis uniformément sur trois zones, créez trois pools de nœuds avec 4 répliques chacun, plutôt qu'un pool de nœuds avec 12 répliques sur trois zones.

Résolution des problèmes

Les nœuds n'atteignent pas les répliques souhaitées :

- Vérifiez si la `limits.nodes` valeur est suffisante
- Vérifiez que vos exigences ne limitent pas trop la sélection d'instances
- Vérifiez les quotas de AWS service pour les types d'instances et les régions que vous utilisez

Le remplacement du nœud prend trop de temps :

- Ajustez les budgets d'interruption pour permettre un plus grand nombre de remplacements simultanés
- Vérifiez si PodDisruptionBudgets vous empêchez la terminaison du nœud

Arrêt inattendu du nœud :

- Vérifiez les `terminationGracePeriod` paramètres `expireAfter` et

- Vérifiez la présence de terminaisons manuelles de nœuds ou AWS d'événements de maintenance

Créez un IngressClass pour configurer un Application Load Balancer

Le mode automatique EKS automatise les tâches courantes liées à l'équilibrage de charge, y compris l'exposition des applications du cluster à Internet.

AWS suggère d'utiliser des équilibreurs de charge d'application (ALB) pour gérer le trafic HTTP et HTTPS. Les équilibreurs de charge Application Load Balancer peuvent acheminer les requêtes en fonction de leur contenu. Pour plus d'informations sur les équilibreurs de charge Application Load Balancer, consultez Qu'est-ce qu'un Application Load Balancer ?

Le mode automatique EKS crée et configure les équilibreurs de charge d'application ()ALBs. Par exemple, le mode automatique EKS crée un équilibreur de charge lorsque vous créez un objet Kubernetes `Ingress` et le configure pour acheminer le trafic vers votre charge de travail dans le cluster.

Présentation

1. Créez une charge de travail que vous souhaitez exposer à Internet.
2. Créez une `IngressClassParams` ressource en spécifiant AWS des valeurs de configuration spécifiques telles que le certificat à utiliser SSL/TLS et les sous-réseaux VPC.
3. Créez une ressource `IngressClass` en spécifiant que le mode automatique EKS sera le contrôleur utilisé pour cette ressource.
4. Créez une ressource `Ingress` qui associe un chemin et un port HTTP à une charge de travail du cluster.

Le mode automatique EKS créera un équilibreur de charge Application Load Balancer pointant vers la charge de travail définie dans la ressource `Ingress`, en utilisant la configuration définie dans la ressource `IngressClassParams`.

Conditions préalables

- Le mode automatique EKS activé sur un cluster Amazon EKS
- Kubectl configuré pour se connecter à votre cluster
  - Vous pouvez utiliser `kubectl apply -f <filename>` pour appliquer les exemples de fichiers YAML de configuration ci-dessous à votre cluster.

> **Note**
>
> Le mode automatique EKS nécessite l'ajout de balises sur les sous-réseaux afin d'identifier les sous-réseaux publics et privés.
> Si vous avez créé votre cluster avec `eksctl`, ces balises sont déjà présentes.
> Découvrez comment the section called "Baliser les sous-réseaux".

Étape 1 : créer une charge de travail

Tout d'abord, créez une charge de travail que vous souhaitez exposer à Internet. Il peut s'agir de n'importe quelle ressource Kubernetes qui gère le trafic HTTP, telle qu'un déploiement ou un service.

Cet exemple utilise un service HTTP simple appelé `service-2048`, qui écoute sur le port `80`. Créez ce service et son déploiement en appliquant le manifeste `2048-deployment-service.yaml` suivant :

```
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: deployment-2048
spec:
  selector:
    matchLabels:
      app.kubernetes.io/name: app-2048
  replicas: 2
  template:
    metadata:
      labels:
        app.kubernetes.io/name: app-2048
    spec:
      containers:
        - image: public.ecr.aws/l6m2t8p7/docker-2048:latest
          imagePullPolicy: Always
          name: app-2048
          ports:
            - containerPort: 80
---
apiVersion: v1
kind: Service
```

```
metadata:
  name: service-2048
spec:
  ports:
    - port: 80
      targetPort: 80
      protocol: TCP
  type: NodePort
  selector:
    app.kubernetes.io/name: app-2048
```

Appliquez la configuration à votre cluster :

```
kubectl apply -f 2048-deployment-service.yaml
```

Les ressources répertoriées ci-dessus seront créées dans l'espace de noms par défaut. Vous pouvez le vérifier en exécutant la commande suivante :

```
kubectl get all -n default
```

Étape 2 : créer IngressClassParams

Créez un `IngressClassParams` objet pour spécifier AWS des options de configuration spécifiques pour l'Application Load Balancer. Dans cet exemple, nous créons une ressource `IngressClassParams` nommée `alb` (qui sera utilisée à l'étape suivante), afin de définir le mode de fonctionnement de l'équilibreur de charge comme `internet-facing`, dans un fichier appelé `alb-ingressclassparams.yaml`.

```
apiVersion: eks.amazonaws.com/v1
kind: IngressClassParams
metadata:
  name: alb
spec:
  scheme: internet-facing
```

Appliquez la configuration à votre cluster :

```
kubectl apply -f alb-ingressclassparams.yaml
```

Étape 3 : créer IngressClass

Créez un `IngressClass` qui fait référence aux valeurs de configuration AWS spécifiques définies dans la `IngressClassParams` ressource dans un fichier nomméalb-ingressclass.yaml. Notez le nom de la ressource `IngressClass`. Dans cet exemple, la ressource `IngressClass` et la ressource `IngressClassParams` portent toutes deux le nom `alb`.

Utilisez l'annotation `is-default-class` pour indiquer si les ressources `Ingress` doivent utiliser cette classe par défaut.

```
apiVersion: networking.k8s.io/v1
kind: IngressClass
metadata:
  name: alb
  annotations:
    # Use this annotation to set an IngressClass as Default
    # If an Ingress doesn't specify a class, it will use the Default
    ingressclass.kubernetes.io/is-default-class: "true"
spec:
  # Configures the IngressClass to use EKS Auto Mode
  controller: eks.amazonaws.com/alb
  parameters:
    apiGroup: eks.amazonaws.com
    kind: IngressClassParams
    # Use the name of the IngressClassParams set in the previous step
    name: alb
```

Pour plus d'informations sur les options de configuration, consultez the section called "IngressClassParams Référence".

Appliquez la configuration à votre cluster :

```
kubectl apply -f alb-ingressclass.yaml
```

Étape 4 : créer une ressource Ingress

Créez une ressource `Ingress` dans un fichier nommé `alb-ingress.yaml`. Le but de cette ressource est d'associer les chemins et les ports de l'équilibreur de charge Application Load Balancer aux charges de travail de votre cluster. Dans cet exemple, nous créons une ressource `Ingress` nommée `2048-ingress` qui achemine le trafic vers un service nommé `service-2048` sur le port 80.

Pour plus d'informations sur la configuration de cette ressource, consultez Ingress dans la documentation Kubernetes.

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: 2048-ingress
spec:
  # this matches the name of IngressClass.
  # this can be omitted if you have a default ingressClass in cluster: the one with
 ingressclass.kubernetes.io/is-default-class: "true"  annotation
  ingressClassName: alb
  rules:
    - http:
        paths:
          - path: /*
            pathType: ImplementationSpecific
            backend:
              service:
                name: service-2048
                port:
                  number: 80
```

Appliquez la configuration à votre cluster :

```
kubectl apply -f alb-ingress.yaml
```

Étape 5 : vérifier l'état

Utilisez `kubectl` pour afficher l'état de la ressource `Ingress`. Il peut falloir quelques minutes pour que l'équilibreur de charge soit disponible.

Utilisez le nom de la ressource `Ingress` définie à l'étape précédente. Par exemple :

```
kubectl get ingress 2048-ingress
```

Une fois la ressource prête, extrayez le nom de domaine de l'équilibreur de charge.

```
kubectl get ingress 2048-ingress -o
 jsonpath='{.status.loadBalancer.ingress[0].hostname}'
```

Pour afficher le service dans un navigateur web, vérifiez le port et le chemin indiqués dans la résolution `Ingress`.

Étape 6 : nettoyage

Pour nettoyer l'équilibreur de charge, utilisez la commande suivante :

```
kubectl delete ingress 2048-ingress
kubectl delete ingressclass alb
kubectl delete ingressclassparams alb
```

Le mode automatique EKS supprimera automatiquement l'équilibreur de charge associé dans votre AWS compte.

IngressClassParams Référence

Le tableau ci-dessous constitue une référence rapide des options de configuration les plus courantes.

Champ	Description	Exemple de valeur
`scheme`	Définit si l'ALB est interne ou orienté Internet	`internet-facing`
`namespaceSelector`	Limite les espaces de noms qui peuvent l'utiliser IngressClass	`environment: prod`
`group.name`	Regroupe plusieurs ressources Ingress pour partager un même ALB	`retail-apps`
`ipAddressType`	Définit le type d'adresse IP de l'ALB	`dualstack`
`subnets.ids`	Liste des sous-réseaux pour le déploiement IDs d'ALB	`subnet-xxxx, subnet-yyyy`
`subnets.tags`	Filtres de balises pour sélectionner les sous-réseaux destinés à l'ALB	`Environment: prod`

Champ	Description	Exemple de valeur
`certificateARNs`	ARNs de certificats SSL à utiliser	`arn:aws: acm:region:account:certificate/id`
`tags`	Tags personnalisés pour les AWS ressources	`Environment: prod, Team: platform`
`loadBalancerAttributes`	Attributs spécifiques à l'équilibreur de charge	`idle_timeout.timeout_seconds: 60`

Considérations

- Vous ne pouvez pas utiliser les annotations sur et IngressClass pour configurer les équilibreurs de charge avec le mode automatique EKS.
- Vous ne pouvez pas effectuer de réglage ListenerAttributeavec le mode automatique EKS.
- Vous devez mettre à jour le rôle IAM du cluster pour permettre la propagation des balises depuis Kubernetes vers les ressources Load AWS Balancer. Pour de plus amples informations, veuillez consulter the section called “AWS Balises personnalisées pour les ressources EKS Auto”.
- Pour plus d'informations sur l'association de ressources au mode automatique EKS ou au AWS Load Balancer Controller autogéré, consultez. the section called “Référence de migration”
- Pour plus d'informations sur la résolution des problèmes liés aux équilibreurs de charge, consultez the section called “Dépannage”.
- Pour des considérations supplémentaires concernant l'utilisation de la fonctionnalité d'équilibrage de charge du mode automatique EKS, consultez the section called “Équilibrage de charge”.

Les tableaux suivants fournissent une comparaison détaillée des modifications IngressClassParams, des annotations d'entrée et des TargetGroupBinding configurations pour le mode automatique EKS. Ces tableaux mettent en évidence les principales différences entre la fonctionnalité d'équilibrage de charge du mode automatique EKS et le contrôleur d'équilibreur de charge open source, notamment l'évolution des versions d'API, les fonctionnalités obsolètes et les nouveaux noms de paramètres.

IngressClassParams

Précédent	Nouveau	Description
`elbv2.k8s.aws/v1beta1`	`eks.amazonaws.com/v1`	Modifications de version d'API
`spec.certificateArn`	`spec.certificateARNs`	Support pour plusieurs certificats ARNs
`spec.subnets.tags`	`spec.subnets.matchTags`	Modification du schéma de correspondance des sous-réseaux
`spec.listeners.listenerAttributes`	Non pris en charge	Pas encore pris en charge par le mode automatique EKS

Annotations Ingress

Précédent	Nouveau	Description
`kubernetes.io/ingress.class`	Non pris en charge	Utilisation de `spec.ingressClassName` sur les objets Ingress
`alb.ingress.kubernetes.io/group.name`	Non pris en charge	Spécifier les groupes dans IngressClass uniquement
`alb.ingress.kubernetes.io/waf-acl-id`	Non pris en charge	Utilisation de WAF v2 à la place
`alb.ingress.kubernetes.io/web-acl-id`	Non pris en charge	Utilisation de WAF v2 à la place
`alb.ingress.kubernetes.io/shield-advanced-protection`	Non pris en charge	Intégration Shield désactivée

Précédent	Nouveau	Description
`alb.ingress.kubernetes.io/auth-type: oidc`	Non pris en charge	Type d'authentification OIDC non pris en charge actuellement

TargetGroupBinding

Précédent	Nouveau	Description
`elbv2.k8s.aws/v1beta1`	`eks.amazonaws.com/v1`	Modifications de version d'API
`spec.targetType` facultatif	`spec.targetType` obligatoire	Spécifications explicite du type de cible
`spec.networking.ingress.from`	Non pris en charge	Ne prend plus en charge NLB sans groupes de sécurité

Pour utiliser la TargetGroupBinding fonctionnalité personnalisée, vous devez étiqueter le groupe cible avec la eks-cluster-name balise eks : avec le nom du cluster afin d'accorder au contrôleur les autorisations IAM nécessaires. Sachez que le contrôleur supprimera le groupe cible lorsque la TargetGroupBinding ressource ou le cluster sera supprimé.

Utilisation des annotations de service pour configurer les équilibreurs de charge Network Load Balancer

Découvrez comment configurer les équilibreurs de charge Network Load Balancer (NLB) dans Amazon EKS à l'aide des annotations du service Kubernetes. Cette rubrique décrit les annotations prises en charge par le mode automatique d'EKS pour personnaliser le comportement du NLB, notamment l'accessibilité à Internet, les bilans de santé, la SSL/TLS résiliation et les modes de ciblage IP.

Lorsque vous créez un service Kubernetes de type `LoadBalancer` EKS Auto Mode, EKS approvisionne et configure automatiquement un AWS Network Load Balancer en fonction des annotations que vous spécifiez. Cette approche déclarative vous permet de gérer les configurations

des équilibreurs de charge directement via vos manifestes Kubernetes, conformément aux pratiques d'infrastructure en tant que code.

Le mode automatique EKS gère le provisionnement du Network Load Balancer par défaut pour tous les types LoadBalancer de services. Aucune installation ou configuration de contrôleur supplémentaire n'est requise. La spécification `loadBalancerClass: eks.amazonaws.com/nlb` est automatiquement définie comme valeur par défaut du cluster, simplifiant ainsi le déploiement et assurant la compatibilité avec les charges de travail Kubernetes existantes.

> **Note**
>
> Le mode automatique EKS nécessite l'ajout de balises sur les sous-réseaux afin d'identifier les sous-réseaux publics et privés.
> Si vous avez créé votre cluster avec `eksctl`, ces balises sont déjà présentes.
> Découvrez comment the section called "Baliser les sous-réseaux".

Exemple de service

Pour plus d'informations sur la ressource Kubernetes `Service`, consultez la documentation Kubernetes.

Consultez l'exemple de ressource `Service` ci-dessous :

```
apiVersion: v1
kind: Service
metadata:
  name: echoserver
  annotations:
    # Specify the load balancer scheme as internet-facing to create a public-facing
 Network Load Balancer (NLB)
    service.beta.kubernetes.io/aws-load-balancer-scheme: internet-facing
spec:
  selector:
    app: echoserver
  ports:
    - port: 80
      targetPort: 8080
      protocol: TCP
  type: LoadBalancer
  # Specify the new load balancer class for NLB as part of EKS Auto Mode feature
```

```
  # For clusters with Auto Mode enabled, this field can be omitted as it's the default
  loadBalancerClass: eks.amazonaws.com/nlb
```

Annotations couramment utilisées

Le tableau ci-dessous présente les annotations couramment utilisées et prises en charge par le mode automatique EKS. Notez que le mode automatique EKS ne prend pas en charge toutes les annotations existantes.

> **Tip**
>
> Toutes les annotations suivantes doivent être préfixées par `service.beta.kubernetes.io/`

Champ	Description	Exemple
`aws-load-balancer-type`	Spécifie le type d'équilibreur de charge. Utilisez `external` pour les nouveaux déploiements.	`external`
`aws-load-balancer-nlb-target-type`	Spécifie s'il faut acheminer le trafic vers les instances de nœuds ou directement vers le pod IPs. Utilisez `instance` pour les déploiements standards ou `ip` pour un routage direct vers les pods.	`instance`
`aws-load-balancer-scheme`	Contrôle si l'équilibreur de charge est interne ou accessible depuis Internet.	`internet-facing`
`aws-load-balancer-healthcheck-protocol`	Protocole de surveillance de l'état pour le groupe cible. Les options courantes sont `TCP` (par défaut) ou `HTTP`.	`HTTP`

Champ	Description	Exemple
`aws-load-balancer-healthcheck-path`	Le chemin HTTP pour les contrôles de santé lors de l'utilisation HTTP/HTTPS du protocole.	`/healthz`
`aws-load-balancer-healthcheck-port`	Port utilisé pour la surveillance de l'état. Il peut s'agir d'un numéro de port spécifique ou du `traffic-port` .	`traffic-port`
`aws-load-balancer-subnets`	Spécifie les sous-réseaux dans lesquels créer l'équilibreur de charge. Peut utiliser un sous-réseau IDs ou des noms.	`subnet-xxxx, subnet-yyyy`
`aws-load-balancer-ssl-cert`	ARN du certificat SSL fourni par Certificate Manager for HTTPS/TLS. AWS	`arn:aws: acm:region:account:certificate/cert-id`
`aws-load-balancer-ssl-ports`	Spécifie les ports devant utiliser SSL/TLS.	`443, 8443`
`load-balancer-source-ranges`	Plages CIDR autorisées à accéder à l'équilibreur de charge.	`10.0.0.0/24, 192.168.1.0/24`
`aws-load-balancer-additional-resource-tags`	AWS Balises supplémentaires à appliquer à l'équilibreur de charge et aux ressources associées.	`Environment=prod,Team=platform`
`aws-load-balancer-ip-address-type`	Spécifie si l'équilibreur de charge utilise IPv4 ou double pile (IPv4 + IPv6).	`ipv4` ou `dualstack`

Considérations

- Vous devez mettre à jour le rôle IAM du cluster pour permettre la propagation des balises depuis Kubernetes vers les ressources Load AWS Balancer. Pour de plus amples informations, veuillez consulter the section called “AWS Balises personnalisées pour les ressources EKS Auto”.
- Pour plus d'informations sur l'association de ressources au mode automatique EKS ou au AWS Load Balancer Controller autogéré, consultez. the section called “Référence de migration”
- Pour plus d’informations sur la résolution des problèmes liés aux équilibreurs de charge, consultez the section called “Dépannage”.
- Pour des considérations supplémentaires concernant l’utilisation de la fonctionnalité d’équilibrage de charge du mode automatique EKS, consultez the section called “Équilibrage de charge”.

Lors de la migration vers le mode automatique EKS pour l’équilibrage de charge, plusieurs modifications concernant les annotations de service et la configuration des ressources sont nécessaires. Les tableaux suivants présentent les principales différences entre l’ancienne et la nouvelle mise en œuvre, incluant les options non prises en charge et les alternatives recommandées.

Annotations de service

Précédent	Nouveau	Description
`service.beta.kubernetes.io/load-balancer-source-ranges`	Non pris en charge	Utiliser `spec.loadBalancerSourceRanges` sur le service
`service.beta.kubernetes.io/aws-load-balancer-type`	Non pris en charge	Utiliser `spec.loadBalancerClass` sur le service
`service.beta.kubernetes.io/aws-load-balancer-internal`	Non pris en charge	Utilisation des `service.beta.kubernetes.io/aws-load-balancer-scheme`
`service.beta.kubernetes.io/aws-load-`	Non pris en charge	Utiliser `service.beta.kubernetes.io/aws-load-balancer-`

Précédent	Nouveau	Description
`balancer-proxy-protocol`		`target-group-attributes` à la place
Divers attributs de l'équilibreur de charge	Non pris en charge	Utilisation des `service.beta.kubernetes.io/aws-load-balancer-attributes`
`service.beta.kubernetes.io/aws-load-balancer-access-log-enabled`	Non pris en charge	Utiliser `service.beta.kubernetes.io/aws-load-balancer-attributes` à la place
`service.beta.kubernetes.io/aws-load-balancer-access-log-s3-bucket-name`	Non pris en charge	Utiliser `service.beta.kubernetes.io/aws-load-balancer-attributes` à la place
`service.beta.kubernetes.io/aws-load-balancer-access-log-s3-bucket-prefix`	Non pris en charge	Utiliser `service.beta.kubernetes.io/aws-load-balancer-attributes` à la place
`service.beta.kubernetes.io/aws-load-balancer-cross-zone-load-balancing-enabled`	Non pris en charge	Utiliser `service.beta.kubernetes.io/aws-load-balancer-attributes` à la place

Pour migrer à partir des annotations d'attributs d'équilibreur de charge désormais obsolètes, regroupez ces paramètres dans l'annotation `service.beta.kubernetes.io/aws-load-balancer-attributes`. Cette annotation accepte une liste de paires clé-valeur séparées par des virgules afin de définir plusieurs attributs de l'équilibreur de charge. Par exemple, pour spécifier la journalisation des accès et l'équilibrage de charge entre zones, utilisez le format suivant :

```
service.beta.kubernetes.io/aws-load-balancer-attributes:
 access_logs.s3.enabled=true,access_logs.s3.bucket=my-bucket,access_logs.s3.prefix=my-
prefix,load_balancing.cross_zone.enabled=true
```

Ce format consolidé offre une méthode plus cohérente et flexible pour configurer les attributs de l'équilibreur de charge, tout en réduisant le nombre d'annotations nécessaires. Passez en revue vos configurations de service existantes et mettez-les à jour pour utiliser ce format consolidé.

TargetGroupBinding

Précédent	Nouveau	Description
`elbv2.k8s.aws/v1beta1`	`eks.amazonaws.com/v1`	Modifications de version d'API
`spec.targetType` facultatif	`spec.targetType` obligatoire	Spécifications explicite du type de cible
`spec.networking.ingress.from`	Non pris en charge	Ne prend plus en charge NLB sans groupes de sécurité

Remarque : Pour utiliser la TargetGroupBinding fonctionnalité personnalisée, vous devez étiqueter le groupe cible avec le `eks:eks-cluster-name` tag portant le nom du cluster afin d'accorder au contrôleur les autorisations IAM nécessaires. Sachez que le contrôleur supprimera le groupe cible lorsque la TargetGroupBinding ressource ou le cluster sera supprimé.

Création d'une classe de stockage

Une `StorageClass` dans le mode automatique Amazon EKS définit la façon dont les volumes Amazon EBS sont automatiquement provisionnés lorsque les applications demandent du stockage persistant. Cette page explique comment créer et configurer une `StorageClass` compatible avec le mode automatique Amazon EKS pour le provisionnement de volumes EBS.

En configurant une `StorageClass`, vous pouvez définir les paramètres par défaut de vos volumes EBS, notamment le type de volume, le chiffrement, le nombre d'IOPS et d'autres paramètres liés au stockage. Vous pouvez également configurer le `StorageClass` pour utiliser les clés AWS KMS pour la gestion du chiffrement.

Le mode automatique EKS ne crée pas de `StorageClass` pour vous. Vous devez créer une `StorageClass` faisant référence à `ebs.csi.eks.amazonaws.com` pour utiliser les fonctionnalités de stockage du mode automatique EKS.

Tout d'abord, créez un fichier nommé `storage-class.yaml` :

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: auto-ebs-sc
  annotations:
    storageclass.kubernetes.io/is-default-class: "true"
allowedTopologies:
- matchLabelExpressions:
  - key: eks.amazonaws.com/compute-type
    values:
    - auto
provisioner: ebs.csi.eks.amazonaws.com
volumeBindingMode: WaitForFirstConsumer
parameters:
  type: gp3
  encrypted: "true"
```

Ensuite, appliquez la classe de stockage à votre cluster.

```
kubectl apply -f storage-class.yaml
```

Composants clés :

- `provisioner: ebs.csi.eks.amazonaws.com` : utilise le mode automatique EKS
- `allowedTopologies` : en spécifiant `matchLabelExpressions` pour filtrer selon `eks.amazonaws.com/compute-type:auto`, vous vous assurez que si vos pods ont besoin qu'un volume soit automatiquement provisionné via le mode automatique, ceux-ci ne seront pas programmés sur des nœuds qui ne sont pas gérés par le mode automatique.
- `volumeBindingMode: WaitForFirstConsumer` : retarde la création du volume jusqu'à ce qu'un pod en ait besoin
- `type: gp3` : spécifie le type de volume EBS
- `encrypted: "true"` : EBS chiffre tous les volumes créés avec la `StorageClass`. EBS utilisera l'alias de clé `aws/ebs` par défaut. Pour plus d'informations, consultez Fonctionnement du

chiffrement Amazon EBS dans le Guide de l'utilisateur Amazon EBS. Cette valeur est facultative, mais recommandée.

- `storageclass.kubernetes.io/is-default-class: "true"` : Kubernetes utilisera cette classe de stockage par défaut, à moins que vous ne spécifiiez une autre classe de volume dans une demande de volume persistant. Cette valeur est facultative. Faites preuve de prudence lorsque vous définissez cette valeur si vous effectuez une migration depuis un autre contrôleur de stockage.

Utilisation d'une clé KMS autogérée pour chiffrer les volumes EBS

Pour utiliser une clé KMS autogérée afin de chiffrer des volumes EBS automatisés par le mode automatique EKS, vous devez :

1. Créer une clé KMS autogérée.
   - Pour plus d'informations, consultez Création d'une clé KMS de chiffrement symétrique ou Utilisation de KMS par Amazon Elastic Block Store (Amazon EBS).
2. Créez une nouvelle politique permettant l'accès à la clé KMS.
   - Utilisez l'exemple de politique IAM ci-dessous comme modèle pour créer la politique. Insérez l'ARN de votre nouvelle clé KMS autogérée. Pour plus d'informations, consultez la section Création de rôles et attachement de politiques (console) dans le guide de l'utilisateur AWS IAM.
3. Attachez la politique au rôle du cluster EKS.
   - Utilisez la AWS console pour trouver l'ARN du rôle de cluster EKS. Les informations relatives au rôle sont visibles dans la section Présentation. Pour de plus amples informations, veuillez consulter the section called "Rôle IAM du cluster".
4. Mettez à jour la `StorageClass` pour faire référence à l'ID de la clé KMS dans le champ `parameters.kmsKeyId`.

Exemple de politique IAM KMS autogérée

Mettez à jour les valeurs suivantes dans la politique ci-dessous :

- `<account-id>`— Votre identifiant de AWS compte, tel que `111122223333`
- `<aws-region>`— La AWS région de votre cluster, telle que `us-west-2`

```
{
```

```
  "Version":"2012-10-17",
  "Id": "key-auto-policy-3",
  "Statement": [
      {
          "Sid": "Enable IAM User Permissions",
          "Effect": "Allow",
          "Principal": {
              "AWS": "arn:aws:iam::123456789012:root"
          },
          "Action": "kms:*",
          "Resource": "*"
      },
      {
        "Sid": "Allow access through EBS for all principals in the account that are
 authorized to use EBS",
        "Effect": "Allow",
        "Principal": {
            "AWS": "*"
        },
        "Action": [
            "kms:Encrypt",
            "kms:Decrypt",
            "kms:ReEncrypt*",
            "kms:GenerateDataKey*",
            "kms:CreateGrant",
            "kms:DescribeKey"
        ],
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "kms:CallerAccount": "123456789012",
                "kms:ViaService": "ec2.us-east-1.amazonaws.com"
            }
        }
    }
  ]
}
```

Exemple de **`StorageClass`** KMS autogérée

```
parameters:
  type: gp3
  encrypted: "true"
```

```
  kmsKeyId: <custom-key-arn>
```

Référence des paramètres **`StorageClass`**

Pour des informations générales sur les ressources `StorageClass` de Kubernetes, consultez Classes de stockage dans la documentation Kubernetes.

THe `parameters`section de la `StorageClass` ressource est spécifique à AWS. Utilisez le tableau ci-dessous pour consulter les options disponibles.

Parameters	Valeurs	Par défaut	Description
« csi.storage.k8s.io/fstype »	xfs, ext2, ext3, ext4	ext4	Type de système de fichiers qui sera formaté lors de la création du volume. Ce paramètre n'est pas sensible à la casse !
« type »	io1, io2, gp2, gp3, sc1, st1, standard, sbp1, sbg1	gp3	Type de volume EBS.
« iopsPerGB »			Nombre d'opérations d'E/S par seconde par Gio. Peut être spécifié pour IO1 IO2, et les GP3 volumes.
« Autoriser Auto IOPSPer GBIncreas e »	true, false	false	Lorsque `"true"` le pilote CSI augmente le nombre d'IOPS pour un volume trop faible pour correspon dre à la plage d'IOPS prise en charge par. `iopsPerGB *`

Parameters	Valeurs	Par défaut	Description
			`<volume size>` AWS Cela permet au provisionnement dynamique de réussir en tout temps, même si l'utilisateur spécifie une capacité PVC trop faible ou une valeur `iopsPerGB` insuffisante. Cependant, cette fonctionnalité peut entraîner des coûts plus élevés, car ces volumes obtiennent un niveau d'IOPS supérieur à celui demandé dans `iopsPerGB` .
« iops »			Nombre d'opérations d'E/S par seconde. Peut être spécifié pour IO1 IO2, et les GP3 volumes.
« throughput »		125	Le débit est exprimé en Mio/s. Efficace uniquement lorsque le type de volume gp3 est spécifié.

Parameters	Valeurs	Par défaut	Description
« encrypted »	true, false	false	Indique si le volume doit être chiffré ou non. Les valeurs valides sont « true » ou « false ».
« blockExpress »	true, false	false	Permet la création de volumes io2 Block Express.
"kmsKeyId"			ARN complet de la clé à utiliser pour chiffrer le volume. Si elle n'est pas spécifiée, AWS utilisera la clé KMS par défaut pour la région dans laquelle se trouve le volume. Si elle n'a pas été modifiée, cette clé autogénérée s'appeler a `/aws/ebs`.
« blockSize »			Taille des blocs utilisée lors du formatage du système de fichiers sous-jace nt. Prise en charge uniquement sur les nœuds Linux et pour les types de systèmes de fichiers `ext2`, `ext3`, `ext4` ou `xfs`.

Parameters	Valeurs	Par défaut	Description
« inodeSize »			Taille des inodes utilisée lors du formatage du système de fichiers. Prise en charge uniquement sur les nœuds Linux et pour les types de systèmes de fichiers `ext2`, `ext3`, `ext4` ou `xfs`.
"bytesPerInode"			Nombre de `bytes-per-inode` utilisé lors du formatage du système de fichiers sous-jacent. Prise en charge uniquement sur les nœuds Linux et pour les types de systèmes de fichiers `ext2`, `ext3` et `ext4`.
"numberOfInodes"			Nombre de `number-of-inodes` utilisé lors du formatage du système de fichiers sous-jacent. Prise en charge uniquement sur les nœuds Linux et pour les types de systèmes de fichiers `ext2`, `ext3` et `ext4`.

Parameters	Valeurs	Par défaut	Description
« poste 4 BigAlloc »	true, false	false	Modifie le système de fichiers `ext4` pour activer l'allocation de blocs en clusters en utilisant l'option de formatage `bigalloc`. Avertissement : `bigalloc` peut ne pas être entièrement pris en charge par le noyau Linux de vos nœuds.
« poste 4 ClusterSi ze »			Taille des clusters utilisée lors du formatage d'un système de fichiers `ext4` lorsque la fonctionn alité `bigalloc` est activée. Remarque : le paramètre `ext4BigAl loc` doit être défini sur true.

Pour plus d'informations, voir le pilote AWS EBS CSI activé. GitHub

Considérations

> **Note**
>
> Vous pouvez uniquement déployer des charges de travail en fonction du mode automatique EKS StorageClasses sur les nœuds du mode automatique EKS. Si votre cluster comporte différents types de nœuds, vous devez configurer vos charges de travail afin qu'elles

s'exécutent uniquement sur les nœuds du mode automatique EKS. Pour de plus amples informations, veuillez consulter the section called "Contrôle du déploiement".

La capacité de stockage par blocs du mode automatique EKS est différente de celle du pilote CSI EBS.

- Provisionnement statique
  - Si vous souhaitez utiliser des volumes EBS créés en externe avec le mode automatique EKS, vous devez ajouter manuellement une AWS balise avec la clé `eks:eks-cluster-name` et la valeur du nom du cluster.
- Rejet de démarrage des nœuds
  - Vous ne pouvez pas utiliser la fonctionnalité de rejet de démarrage des nœuds pour empêcher la planification des pods avant que la capacité de stockage ne soit prête
- Balises personnalisées sur les volumes provisionnés dynamiquement
  - Vous ne pouvez pas utiliser le paramètre CLI extra-tag pour configurer des balises personnalisées sur les volumes EBS provisionnés dynamiquement
  - Vous pouvez utiliser le balisage `StorageClass` pour ajouter des balises personnalisées. Le mode automatique EKS ajoutera des balises aux AWS ressources associées. Vous devrez mettre à jour le rôle IAM du cluster afin d'utiliser des balises personnalisées. Pour de plus amples informations, veuillez consulter the section called "AWS Balises personnalisées pour les ressources EKS Auto".
- Métriques de performances détaillées d'EBS
  - Vous ne pouvez pas accéder aux métriques Prometheus pour les performances détaillées d'EBS

Installation du module complémentaire CSI Snapshot Controller

Le mode automatique EKS est compatible avec le module complémentaire Amazon EKS CSI Snapshot Controller.

AWS vous suggère de configurer ce module complémentaire pour qu'il s'exécute sur le pool de `system` nœuds intégré.

Pour en savoir plus, consultez :

- the section called "Exécuter les modules complémentaires critiques"

- the section called "Vérifier les groupes de nœuds créés"
- the section called "Contrôleur d'instantané CSI"

Pour installer le contrôleur d'instantané dans le groupe de nœuds système

1. Ouvrez votre cluster EKS dans la AWS console
2. Dans l'onglet Modules complémentaires, sélectionnez Obtenir plus de modules complémentaires
3. Sélectionnez le module CSI Snapshot Controller, puis Suivant
4. Sur la page Configurer les paramètres des modules complémentaires sélectionnés, sélectionnez Paramètres de configuration facultatifs pour afficher le schéma de configuration des modules complémentaires
   a. Insérez le fichier YAML suivant pour associer le contrôleur d'instantané au groupe de nœuds `system`. Le contrôleur d'instantané inclut une tolérance pour le rejet `CriticalAddonsOnly`.

   ```
   {
           "nodeSelector": {
               "karpenter.sh/nodepool": "system"
           }
   }
   ```

   b. Sélectionnez Suivant
5. Vérifier la configuration du module complémentaire, puis sélectionnez Créer

Désactivation du mode automatique EKS

Vous pouvez désactiver le mode automatique EKS sur un cluster EKS existant. Cette opération est destructrice.

- EKS mettra fin à toutes les EC2 instances gérées par le mode automatique EKS.
- EKS supprime tous les équilibreurs de charge gérés par le mode automatique EKS.
- EKS ne supprime pas les volumes EBS provisionnés par le mode automatique EKS.

Le mode automatique EKS est conçu pour gérer entièrement les ressources qu'il crée. Toute intervention manuelle peut empêcher le mode automatique EKS de nettoyer correctement ces ressources lors de sa désactivation. Par exemple, si vous faites référence à un groupe de sécurité géré depuis des règles d'un groupe de sécurité externe et que vous oubliez de supprimer cette

référence avant de désactiver le mode automatique EKS pour un cluster, le groupe de sécurité géré peut ne pas être supprimé (fuite de ressource). Les étapes ci-dessous expliquent comment supprimer un groupe de sécurité ayant fui.

Désactiver le mode automatique EKS (AWS console)

1. Ouvrez la page de présentation de votre cluster dans la AWS Management Console.
2. Sous Mode automatique EKS, sélectionnez Gérer
3. Basculez Mode automatique EKS sur `off`.

Si un groupe de sécurité géré n'est pas supprimé à la fin du processus, vous pouvez le supprimer manuellement en suivant les instructions de la section Suppression d'un groupe de sécurité.

Désactiver le mode automatique EKS (AWS CLI)

Utilisez la commande suivante pour désactiver le mode automatique EKS sur un cluster existant.

La CLI `aws` doit être installée et vous devez être connecté avec des autorisations suffisantes pour gérer les clusters EKS. Pour de plus amples informations, veuillez consulter Configuration.

> **Note**
>
> Les capacités de calcul, de stockage bloc et d'équilibrage de charge doivent toutes être activées ou désactivées dans la même requête.

```
aws eks update-cluster-config \
 --name $CLUSTER_NAME \
 --compute-config enabled=false \
 --kubernetes-network-config '{"elasticLoadBalancing":{"enabled": false}}' \
 --storage-config '{"blockStorage":{"enabled": false}}'
```

Vous pouvez vérifier si un groupe de sécurité géré par le mode automatique EKS n'a pas été supprimé après la désactivation en procédant comme suit :

```
aws ec2 describe-security-groups \
    --filters Name=tag:eks:eks-cluster-name,Values=<cluster-Name> Name=tag-
key,Values=ingress.eks.amazonaws.com/resource,service.eks.amazonaws.com/resource --
query "SecurityGroups[*].[GroupName]"
```

Pour supprimer ensuite le groupe de sécurité :

```
aws ec2 delete-security-group --group-name=<sg-name>
```

Mise à jour de la version Kubernetes d'un cluster du mode automatique Amazon EKS

Cette rubrique explique comment mettre à jour la version Kubernetes de votre cluster du mode automatique. Le mode automatique EKS simplifie le processus de mise à jour en gérant la coordination entre la mise à jour du plan de contrôle et le remplacement des nœuds, tout en préservant la disponibilité des charges de travail grâce aux budgets d'interruption des pods.

Lors de la mise à niveau d'un cluster du mode automatique EKS, de nombreux composants traditionnellement mis à jour manuellement sont désormais gérés automatiquement par le service. La compréhension des aspects automatisés du processus de mise à jour et de vos responsabilités permet d'assurer une transition fluide entre les versions de votre cluster.

Informations sur les mises à jour avec le mode automatique EKS

Une fois la mise à niveau du plan de contrôle lancée, le mode automatique EKS procède à la mise à jour des nœuds de votre cluster. Au fur et à mesure que les anciens nœuds arrivent à expiration, le mode automatique EKS les remplace par de nouveaux nœuds. Les nouveaux nœuds exécutent la nouvelle version Kubernetes correspondante. Le mode automatique EKS respecte les budgets d'interruption des pods pendant le processus de mise à jour des nœuds.

De plus, il n'est plus nécessaire de mettre à jour manuellement certains composants tels que :

- CNI Amazon VPC
- Contrôleur de l'équilibreur de charge AWS
- CoreDNS
- `kube-proxy`
- Karpenter
- Pilote AWS EBS CSI

Le mode automatique EKS remplace ces composants par des fonctionnalités de service intégrées.

Vous restez toutefois responsable de la mise à jour des éléments suivants :

- Applications et charges de travail déployées sur votre cluster
- Modules complémentaires et contrôleurs autogérés
- Modules complémentaires Amazon EKS
  - Découvrez comment… the section called “Mettre à jour un module complémentaire”

Découvrez les bonnes pratiques pour les mises à niveau des clusters

Démarrage de la mise à jour du cluster

Pour démarrer la mise à jour d'un cluster, consultez the section called “Mises à jour de la version Kubernetes”.

Activer ou désactiver la fonction intégrée NodePools

Le mode automatique EKS en possède deux intégrés NodePools. Vous pouvez les activer ou les désactiver à NodePools l'aide de la AWS console, de la CLI ou de l'API.

NodePool Référence intégrée

- `system`
  - Cela NodePool a une `CriticalAddonsOnly` odeur. De nombreux modules complémentaires EKS, tels que CoreDNS, tolèrent ce rejet. Utilisez ce groupe de nœuds système pour séparer les applications critiques du cluster.
  - Prend en charge les architectures `amd64` et `arm64`.
- `general-purpose`
  - Cela NodePool permet de lancer des nœuds pour des charges de travail générales dans votre cluster.
  - Utilise uniquement l'architecture `amd64`.

Les deux sont intégrés NodePools :

- Utiliser l'EKS par défaut NodeClass
- Utiliser uniquement la EC2 capacité à la demande
- Utiliser les familles d' EC2 instances C, M et R
- Exiger des EC2 instances de génération 5 ou plus récentes

> **Note**
>
> L'activation d'au moins une fonction intégrée NodePool est requise pour qu'EKS puisse configurer la « valeur par défaut » NodeClass. Si vous désactivez toutes les NodePools fonctionnalités intégrées, vous devrez créer une personnalisation NodeClass et en configurer une NodePool pour l'utiliser. Pour plus d'informations sur NodeClasses, voirthe section called "Création d'une classe de nœuds".

Procédure

Conditions préalables

- La dernière version de l'interface de ligne de AWS commande (AWS CLI) installée et configurée sur votre appareil. Pour vérifier votre version actuelle, utilisez `aws --version`. Pour installer la dernière version, consultez la section Installation et configuration rapide avec aws configure dans le Guide de l'utilisateur de l'interface de ligne de AWS commande.
  - Connectez-vous à la CLI avec des autorisations IAM suffisantes pour créer des AWS ressources, notamment des politiques IAM, des rôles IAM et des clusters EKS.

Activer avec la AWS CLI

Utilisez la commande suivante pour activer les deux fonctions intégrées NodePools :

```
aws eks update-cluster-config \
  --name <cluster-name> \
  --compute-config '{
    "nodeRoleArn": "<node-role-arn>",
    "nodePools": ["general-purpose", "system"],
    "enabled": true
  }' \
  --kubernetes-network-config '{
  "elasticLoadBalancing":{"enabled": true}
  }' \
  --storage-config '{
  "blockStorage":{"enabled": true}
  }'
```

Vous pouvez modifier la commande pour activer de manière sélective le NodePools.

Désactiver avec la AWS CLI

Utilisez la commande suivante pour désactiver les deux fonctions intégrées NodePools :

```
aws eks update-cluster-config \
  --name <cluster-name> \
  --compute-config '{
  "enabled": true,
  "nodePools": []
  }' \
  --kubernetes-network-config '{
  "elasticLoadBalancing":{"enabled": true}}' \
  --storage-config '{
  "blockStorage":{"enabled": true}
  }'
```

Contrôle du déploiement d'une charge de travail sur les nœuds du mode automatique EKS

Lors de l'exécution de charges de travail dans un cluster EKS utilisant le mode automatique EKS, il peut être nécessaire de contrôler si certaines charges de travail doivent s'exécuter sur des nœuds du mode automatique EKS ou sur d'autres types de calcul. Cette section explique comment utiliser des sélecteurs de nœuds et des règles d'affinité pour garantir que vos charges de travail soient planifiées sur l'infrastructure de calcul souhaitée.

Les exemples de cette rubrique montrent comment utiliser l'étiquette `eks.amazonaws.com/compute-type` pour imposer ou empêcher le déploiement d'une charge de travail sur les nœuds du mode automatique EKS. Cette approche est particulièrement utile dans les clusters en mode mixte, où vous utilisez à la fois le mode automatique EKS et d'autres types de calcul, tels que les fournisseurs Karpenter autogérés ou les groupes de nœuds gérés par EKS.

Les nœuds du mode automatique EKS définissent l'étiquette `eks.amazonaws.com/compute-type` sur `auto`. Vous pouvez utiliser cette étiquette pour contrôler si une charge de travail est déployée sur des nœuds gérés par le mode automatique EKS.

Exigence de déploiement d'une charge de travail sur les nœuds du mode automatique EKS

> **Note**
>
> Cette valeur `nodeSelector` n'est pas obligatoire pour le mode automatique EKS. Cette valeur `nodeSelector` n'est pertinente que si vous exécutez un cluster en mode mixte, avec des types de nœuds non gérés par le mode automatique EKS. Par exemple, un cluster peut combiner une capacité de calcul statique fournie par les groupes de nœuds gérés par EKS et une capacité dynamique fournie par le mode automatique EKS.

Vous pouvez ajouter ce `nodeSelector` aux déploiements ou à d'autres charges de travail afin d'indiquer à Kubernetes à les planifier exclusivement sur des nœuds du mode automatique EKS.

```
apiVersion: apps/v1
kind: Deployment
spec:
  template:
    nodeSelector:
      eks.amazonaws.com/compute-type: auto
```

Exigence de non-déploiement d'une charge de travail sur les nœuds du mode automatique EKS

Vous pouvez ajouter ce `nodeAffinity` aux déploiements ou à d'autres charges de travail afin d'indiquer à Kubernetes de ne pas les planifier sur des nœuds du mode automatique EKS.

```
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: eks.amazonaws.com/compute-type
            operator: NotIn
            values:
            - auto
```

Exécutez les modules complémentaires critiques sur des instances dédiées.

Dans cette rubrique, vous apprendrez comment déployer une charge de travail avec une tolérance `CriticalAddonsOnly` afin que le mode automatique EKS la planifie sur le pool de nœuds `system`.

Le pool de nœuds `system` intégré au mode automatique EKS est conçu pour exécuter des modules complémentaires essentiels sur des instances dédiées. Cette séparation garantit que les composants essentiels disposent de ressources dédiées et sont isolés des charges de travail générales, ce qui améliore la stabilité et les performances globales du cluster.

Ce guide explique comment déployer des modules complémentaires dans le pool de nœuds `system` en utilisant la tolérance `CriticalAddonsOnly` et les sélecteurs de nœuds appropriés. En suivant ces étapes, vous pouvez vous assurer que vos applications critiques sont planifiées sur les nœuds `system` dédiés, en tirant parti des avantages en termes d'isolation et d'allocation des ressources offerts par la structure de pool de nœuds spécialisée du mode automatique EKS.

Le mode automatique EKS dispose de deux pools de nœuds intégrés : `general-purpose` et `system`. Pour de plus amples informations, consultez the section called "Vérifier les groupes de nœuds créés".

Le but du pool de nœuds `system` est de séparer les modules complémentaires critiques sur différents nœuds. Les nœuds provisionnés par le pool de nœuds `system` ont un rejet Kubernetes `CriticalAddonsOnly`. Kubernetes ne planifiera les pods sur ces nœuds que s'ils ont une tolérance correspondante. Pour plus d'informations, consultez Rejets et tolérances dans la documentation de Kubernetes.

Prérequis

- Le cluster du mode automatique EKS avec le groupe de nœuds `system` intégré activé. Pour plus d'informations, consultez the section called "Vérifier les groupes de nœuds créés".
- `kubectl` installé et configuré. Pour de plus amples informations, consultez Configuration.

Procédure

Consultez l'exemple YAML ci-dessous. Notez les configurations suivantes :

- `nodeSelector`— Cela associe la charge de travail au pool de nœuds `system` intégré. Ce pool de nœuds doit être activé avec l'API AWS. Pour de plus amples informations, consultez the section called "Vérifier les groupes de nœuds créés".

- `tolerations` : cette tolérance permet de surmonter le rejet `CriticalAddonsOnly` des nœuds dans le groupe de nœuds `system`.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: sample-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: sample-app
  template:
    metadata:
      labels:
        app: sample-app
    spec:
      nodeSelector:
        karpenter.sh/nodepool: system
      tolerations:
      - key: "CriticalAddonsOnly"
        operator: "Exists"
      containers:
      - name: app
        image: nginx:latest
        resources:
          requests:
            cpu: "500m"
            memory: "512Mi"
```

Pour mettre à jour une charge de travail afin qu'elle s'exécute sur le pool de nœuds `system`, vous devez :

1. Mettez à jour la charge de travail existante pour ajouter les configurations suivantes décrites ci-dessus :
   - `nodeSelector`
   - `tolerations`
2. Déployez la charge de travail mise à jour sur votre cluster avec `kubectl apply`

Une fois la charge de travail mise à jour, elle s’exécutera sur des nœuds dédiés.

Utilisation des politiques réseau avec le mode automatique EKS

Présentation de

À mesure que les clients adaptent leurs environnements applicatifs à l'aide d'EKS, l'isolation du trafic réseau devient de plus en plus fondamentale pour empêcher tout accès non autorisé aux ressources à l'intérieur et à l'extérieur du cluster. Cela est particulièrement important dans un environnement mutualisé dans lequel plusieurs charges de travail indépendantes s'exécutent côte à côte dans le cluster. Les politiques réseau Kubernetes vous permettent d'améliorer le niveau de sécurité du réseau pour vos charges de travail Kubernetes, ainsi que leur intégration avec les points de terminaison externes au cluster. Le mode automatique EKS prend en charge différents types de politiques réseau.

Isolation des couches 3 et 4

Les politiques réseau standard de Kubernetes s'appliquent aux couches 3 et 4 du modèle de réseau OSI et vous permettent de contrôler le flux de trafic au niveau de l'adresse IP ou du port au sein de votre cluster Amazon EKS.

Cas d’utilisation

- Segmentez le trafic réseau entre les charges de travail afin de garantir que seules les applications associées puissent communiquer entre elles.
- Isolez les locataires au niveau de l'espace de noms à l'aide de politiques visant à appliquer la séparation des réseaux.

Application basée sur le DNS

Les clients déploient généralement des charges de travail dans EKS qui font partie d'un environnement distribué plus large, dont certaines doivent communiquer avec des systèmes et des services extérieurs au cluster (trafic en direction du nord). Ces systèmes et services peuvent être hébergés dans le AWS cloud ou AWS totalement externes. Les politiques basées sur le système de noms de domaine (DNS) vous permettent de renforcer votre posture de sécurité en adoptant une approche plus stable et prévisible pour empêcher tout accès non autorisé des pods aux ressources ou points de terminaison externes au cluster. Ce mécanisme élimine le besoin de suivre manuellement et d'autoriser la création de listes d'adresses IP spécifiques. En sécurisant les ressources grâce à une approche basée sur le DNS, vous bénéficiez également d'une plus grande

flexibilité pour mettre à jour l'infrastructure externe sans avoir à assouplir votre posture de sécurité ou à modifier les politiques réseau en cas de modifications apportées aux serveurs et hôtes en amont. Vous pouvez filtrer le trafic sortant vers des points de terminaison externes à l'aide d'un nom de domaine complet (FQDN) ou d'un modèle de correspondance pour un nom de domaine DNS. Cela vous donne la flexibilité supplémentaire d'étendre l'accès à plusieurs sous-domaines associés à un point de terminaison externe au cluster particulier.

Cas d’utilisation

- Utilisez une approche basée sur le DNS pour filtrer l'accès depuis un environnement Kubernetes vers des points de terminaison externes au cluster.
- Accès sécurisé aux AWS services dans un environnement mutualisé.
- Gérez l'accès au réseau, des modules aux charges de travail sur site dans vos environnements de cloud hybride.

Règles relatives à l'administration (ou à l'échelle du cluster)

Dans certains cas, tels que les scénarios multi-locataires, les clients peuvent avoir besoin d'appliquer une norme de sécurité réseau qui s'applique à l'ensemble du cluster. Au lieu de définir et de gérer de manière répétitive une politique distincte pour chaque espace de noms, vous pouvez utiliser une seule stratégie pour gérer de manière centralisée les contrôles d'accès au réseau pour les différentes charges de travail du cluster, quel que soit leur espace de noms. Ces types de politiques vous permettent d'étendre le champ d'application des règles de filtrage de votre réseau appliquées aux couches 3, 4 et lors de l'utilisation des règles DNS.

Cas d’utilisation

- Gérez de manière centralisée les contrôles d'accès au réseau pour toutes les charges de travail (ou un sous-ensemble de celles-ci) de votre cluster EKS.
- Définissez une posture de sécurité réseau par défaut sur l'ensemble du cluster.
- Étendez les normes de sécurité organisationnelles à l'étendue du cluster de manière plus efficace sur le plan opérationnel.

Prise en main

Conditions préalables

- Cluster Amazon EKS avec le mode automatique EKS activé

- kubectl configuré pour se connecter à votre cluster

Étape 1 : activer le contrôleur de politiques réseau

Pour utiliser les politiques réseau avec le mode automatique EKS, vous devez d'abord activer le Network Policy Controller en appliquant un ConfigMap à votre cluster.

1. Créez un fichier nommé `enable-network-policy.yaml` avec le contenu suivant:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: amazon-vpc-cni
  namespace: kube-system
data:
  enable-network-policy-controller: "true"
```

2. Appliquez le ConfigMap à votre cluster :

```
kubectl apply -f enable-network-policy.yaml
```

Étape 2 : créer et tester des politiques réseau

Votre cluster du mode automatique EKS est maintenant configuré pour prendre en charge les politiques réseau Kubernetes. Vous pouvez tester cette configuration à l'aide de the section called "Démonstration de politique Stars".

Étape 3 : ajuster la configuration de l'agent de politique réseau dans la classe de nœuds (facultatif)

Vous pouvez éventuellement créer une nouvelle classe de nœuds pour modifier le comportement par défaut de l'agent de stratégie réseau sur les nœuds ou activer la journalisation des événements de politique réseau. Pour cela, procédez comme suit :

1. Créez ou modifiez un fichier YAML de classe de nœud (par exemple, `nodeclass-network-policy.yaml`) avec le contenu suivant :

```
apiVersion: eks.amazonaws.com/v1
kind: NodeClass
metadata:
  name: network-policy-config
```

```
spec:
  # Optional: Changes default network policy behavior
  networkPolicy: DefaultAllow
  # Optional: Enables logging for network policy events
  networkPolicyEventLogs: Enabled
  # Include other Node Class configurations as needed
```

2. Appliquez la configuration de la classe de nœuds à votre cluster :

```
kubectl apply -f nodeclass-network-policy.yaml
```

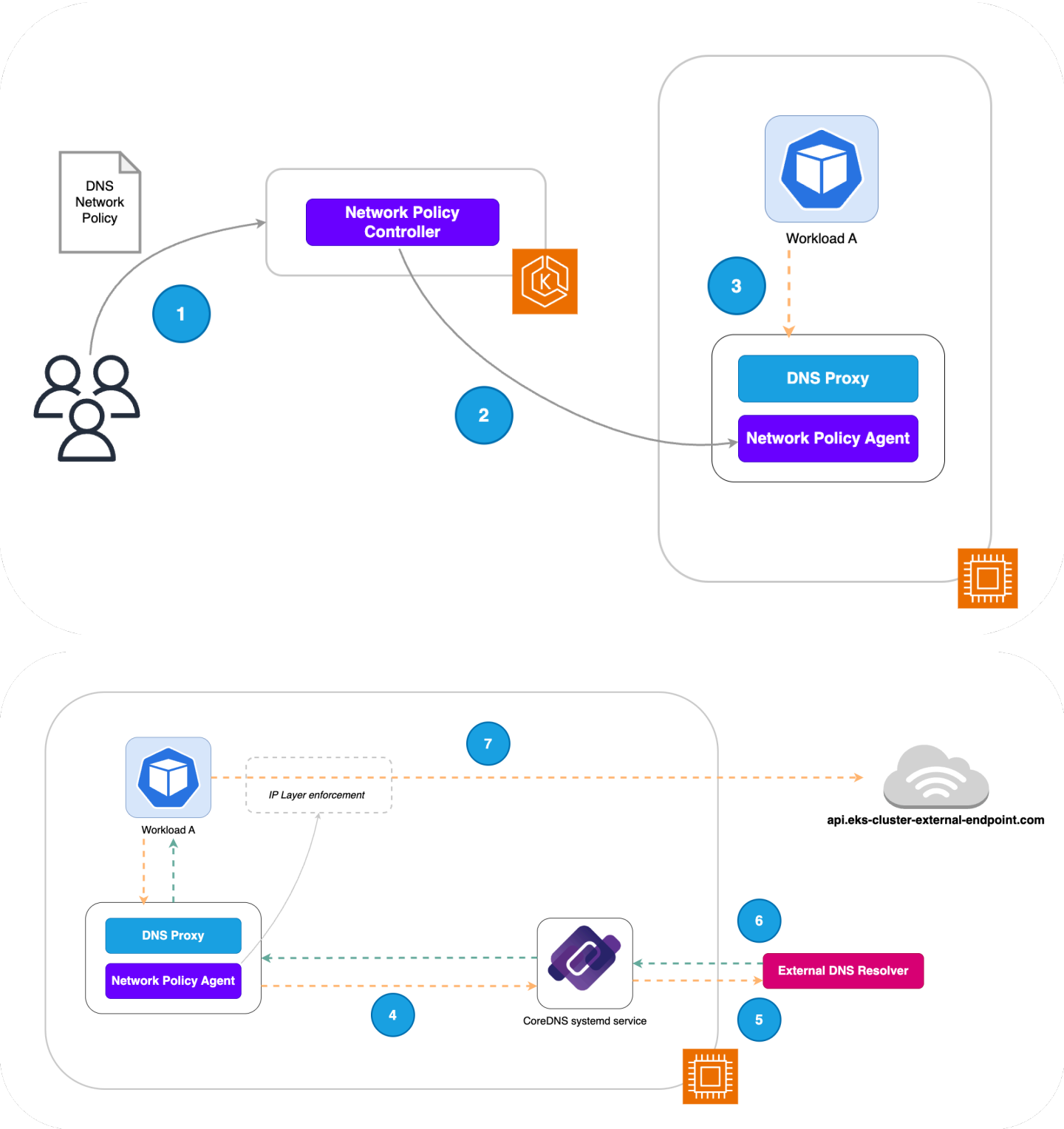
3. Vérifiez que la classe de nœuds a été créée :

```
kubectl get nodeclass network-policy-config
```

4. Mettez à jour votre groupe de nœuds pour utiliser cette classe de nœuds. Pour de plus amples informations, veuillez consulter the section called "Création d'un groupe de nœuds".

Fonctionnement

Politique réseau basée sur le DNS



1. L'équipe de la plateforme applique une politique basée sur le DNS au cluster EKS.

2. Le Network Policy Controller est chargé de surveiller la création des politiques au sein du cluster, puis de réconcilier les points de terminaison des politiques. Dans ce cas d'utilisation, le contrôleur de stratégie réseau demande à l'agent du nœud de filtrer les demandes DNS en fonction des domaines autorisés dans la politique créée. Les noms de domaine sont autorisés à l'aide du nom de domaine complet ou d'un nom de domaine correspondant à un modèle défini dans la configuration des ressources Kubernetes.
3. La charge de travail A tente de résoudre l'adresse IP d'un point de terminaison externe au cluster. La demande DNS passe d'abord par un proxy qui filtre ces demandes en fonction de la liste d'autorisation appliquée par le biais de la politique réseau.
4. Une fois que la demande DNS passe par la liste d'autorisation du filtre DNS, elle est transmise par proxy à CoreDNS,
5. CoreDNS envoie à son tour la demande au résolveur DNS externe (Amazon Route 53 Resolver) pour obtenir la liste des adresses IP sous-jacentes au nom de domaine.
6. Les données résolues IPs avec TTL sont renvoyées dans la réponse à la demande DNS. Ils IPs sont ensuite écrits dans une carte eBPF qui est utilisée à l'étape suivante pour l'application de la couche IP.
7. Les sondes eBPF connectées à l'interface Pod veth filtreront ensuite le trafic sortant de la charge de travail A vers le point de terminaison externe au cluster en fonction des règles en place. Cela garantit que les pods ne peuvent envoyer du trafic externe au cluster que vers les domaines autorisés IPs répertoriés. Leur validité IPs est basée sur le TTL extrait du résolveur DNS externe (Amazon Route 53 Resolver).

Utilisation de la politique du réseau d'applications

Il `ApplicationNetworkPolicy` combine les fonctionnalités des politiques réseau standard de Kubernetes avec le filtrage basé sur le DNS au niveau de l'espace de noms en utilisant une seule définition de ressource personnalisée (CRD). Par conséquent, `ApplicationNetworkPolicy` ils peuvent être utilisés pour :

1. Définition des restrictions au niveau des couches 3 et 4 de la pile réseau à l'aide de blocs IP et de numéros de port.
2. Définir des règles qui s'appliquent à la couche 7 de la pile réseau et vous permettre de filtrer le trafic en fonction de FQDNs.

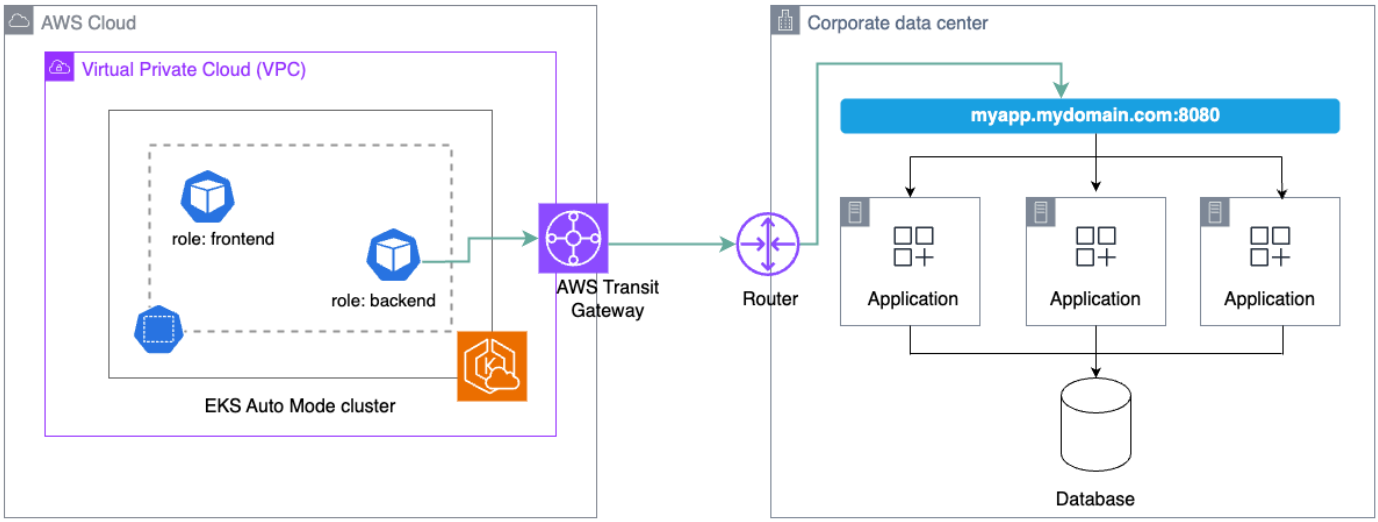
Remarque importante : les règles basées sur le DNS définies à l'aide du ne `ApplicationNetworkPolicy` s'appliquent qu'aux charges de travail exécutées dans des instances lancées en mode EKS Auto. EC2

Exemple

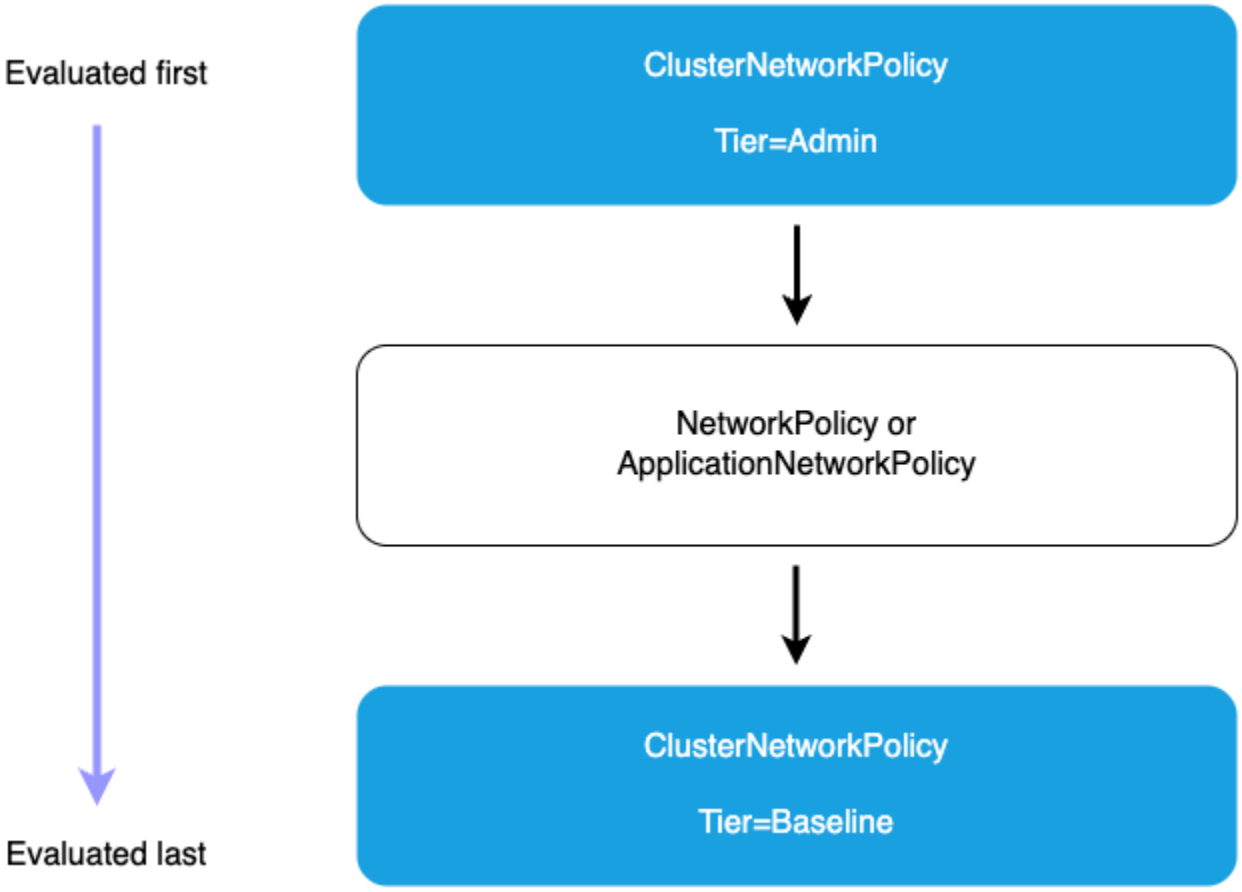
Dans votre cluster EKS Auto Mode, vous avez une charge de travail qui doit communiquer avec une application sur site située derrière un équilibreur de charge portant un nom DNS. Vous pouvez y parvenir en appliquant la politique réseau suivante :

```
apiVersion: networking.k8s.aws/v1alpha1
kind: ApplicationNetworkPolicy
metadata:
  name: my-onprem-app-egress
  namespace: galaxy
spec:
  podSelector:
    matchLabels:
      role: backend
  policyTypes:
  - Egress
  egress:
  - to:
    - domainNames:
      - "myapp.mydomain.com"
    ports:
    - protocol: TCP
      port: 8080
```

Au niveau du réseau Kubernetes, cela permettrait de sortir de tous les pods de l'espace de noms « galaxy » étiqueté avec `role: backend` pour se connecter au nom de domaine myapp.mydomain.com sur le port TCP 8080. En outre, vous devez configurer la connectivité réseau pour le trafic sortant de votre VPC vers le centre de données de votre entreprise.



Politique réseau de l'administrateur (ou du cluster)



Utilisation de la politique réseau du cluster

Lorsque vous utilisez un`ClusterNetworkPolicy`, les politiques du niveau administrateur sont évaluées en premier et ne peuvent pas être remplacées. Lorsque les politiques de niveau administrateur ont été évaluées, les politiques délimitées par espace de noms standard sont utilisées pour exécuter les règles de segmentation du réseau appliquées. Cela peut être accompli en utilisant l'un `ApplicationNetworkPolicy` ou l'autre`NetworkPolicy`. Enfin, les règles du niveau de référence qui définissent les restrictions réseau par défaut pour les charges de travail des clusters seront appliquées. Ces règles du niveau de base peuvent être remplacées par les politiques délimitées de l'espace de noms si nécessaire.

Exemple

Votre cluster contient une application que vous souhaitez isoler des charges de travail des autres locataires. Vous pouvez bloquer explicitement le trafic du cluster en provenance d'autres espaces de noms afin d'empêcher l'accès réseau à l'espace de noms de charge de travail sensible.

```
apiVersion: networking.k8s.aws/v1alpha1
kind: ClusterNetworkPolicy
metadata:
  name: protect-sensitive-workload
spec:
  tier: Admin
  priority: 10
  subject:
    namespaces:
      matchLabels:
        kubernetes.io/metadata.name: earth
  ingress:
    - action: Deny
      from:
      - namespaces:
          matchLabels: {} # Match all namespaces.
      name: select-all-deny-all
```

Considérations

Comprendre l'ordre d'évaluation des politiques

Les fonctionnalités de politique réseau prises en charge dans EKS sont évaluées dans un ordre spécifique afin de garantir une gestion du trafic prévisible et sécurisée. Il est donc important de

comprendre le flux d'évaluation afin de concevoir une posture de sécurité réseau efficace pour votre environnement.

1. Politiques du niveau administrateur (évaluées en premier) : toutes les politiques du niveau administrateur ClusterNetworkPolicies sont évaluées avant les autres politiques. Au niveau administrateur, les politiques sont traitées par ordre de priorité (le numéro de priorité le plus bas en premier). Le type d'action détermine ce qui se passe ensuite.
   - Refuser l'action (priorité la plus élevée) : lorsqu'une politique d'administration associée à une action de refus correspond au trafic, ce trafic est immédiatement bloqué, quelles que soient les autres politiques. Aucune autre ClusterNetworkPolicy NetworkPolicy règle n'est traitée. Cela garantit que les contrôles de sécurité à l'échelle de l'organisation ne peuvent pas être remplacés par des politiques au niveau de l'espace de noms.
   - Autoriser l'action : une fois les règles de refus évaluées, les politiques d'administration comportant des actions d'autorisation sont traitées par ordre de priorité (le numéro de priorité le plus bas en premier). Lorsqu'une action Autoriser correspond, le trafic est accepté et aucune autre évaluation de politique n'est effectuée. Ces politiques peuvent accorder l'accès à plusieurs espaces de noms en fonction de sélecteurs d'étiquettes, offrant ainsi un contrôle centralisé des charges de travail pouvant accéder à des ressources spécifiques.
   - Action de transfert : les actions de transfert prévues dans les politiques du niveau administrateur délèguent la prise de décision aux niveaux inférieurs. Lorsque le trafic correspond à une règle Pass, l'évaluation ignore toutes les règles de niveau administrateur restantes pour ce trafic et passe directement au NetworkPolicy niveau. Cela permet aux administrateurs de déléguer explicitement le contrôle de certains modèles de trafic aux équipes d'application. Par exemple, vous pouvez utiliser les règles Pass pour déléguer la gestion du trafic intra-espace à des administrateurs d'espaces de noms tout en maintenant des contrôles stricts sur l'accès externe.
2. Niveau de politique réseau : si aucune politique de niveau administrateur ne correspond à Deny ou Allow, ou si une action Pass correspond, les NetworkPolicy ressources traditionnelles ApplicationNetworkPolicy et limitées à l'espace de noms sont ensuite évaluées. Ces politiques fournissent un contrôle précis au sein des espaces de noms individuels et sont gérées par les équipes d'application. Les politiques étendues à l'espace de noms ne peuvent être que plus restrictives que les politiques d'administration. Ils ne peuvent pas annuler la décision de refus d'une politique d'administration, mais ils peuvent restreindre davantage le trafic autorisé ou approuvé par les politiques d'administration.
3. Politiques d'administration du niveau de référence : si aucune politique d'administration ou limitée à un espace de noms ne correspond au trafic, le niveau de référence est évalué. ClusterNetworkPolicies Ils fournissent des postures de sécurité par défaut qui peuvent

être remplacées par des politiques délimitées par des espaces de noms, permettant aux administrateurs de définir des valeurs par défaut à l'échelle de l'organisation tout en donnant aux équipes la flexibilité de les personnaliser selon les besoins. Les politiques de base sont évaluées par ordre de priorité (le numéro de priorité le plus bas en premier).

4. Refus par défaut (si aucune politique ne correspond) : ce deny-by-default comportement garantit que seules les connexions explicitement autorisées sont autorisées, garantissant ainsi un niveau de sécurité élevé.

Appliquer le principe du moindre privilège

- Commencez par des politiques restrictives et ajoutez progressivement des autorisations selon les besoins : commencez par mettre en œuvre des deny-by-default politiques au niveau du cluster, puis ajoutez progressivement des règles d'autorisation au fur et à mesure que vous validez les exigences de connectivité légitimes. Cette approche oblige les équipes à justifier explicitement chaque connexion externe, créant ainsi un environnement plus sécurisé et plus contrôlable.
- Auditez régulièrement et supprimez les règles de politique non utilisées : les politiques réseau peuvent s'accumuler au fil du temps à mesure que les applications évoluent, laissant derrière elles des règles obsolètes qui étendent inutilement votre surface d'attaque. Mettez en œuvre un processus de révision régulier pour identifier et supprimer les règles de politique qui ne sont plus nécessaires, afin de garantir que votre posture de sécurité reste stricte et maintenable.
- Utilisez des noms de domaine spécifiques plutôt que des modèles généraux lorsque cela est possible. Si les modèles génériques sont pratiques, ils `*.amazonaws.com` permettent également d'accéder à un large éventail de services. Dans la mesure du possible, spécifiez des noms de domaine exacts, par exemple `s3.us-west-2.amazonaws.com` pour limiter l'accès aux seuls services spécifiques dont vos applications ont besoin, réduisant ainsi le risque de déplacement latéral si une charge de travail est compromise.

Utilisation de politiques basées sur le DNS dans EKS

- Les règles basées sur le DNS définies à l'aide du `ApplicationNetworkPolicy` ne s'appliquent qu'aux charges de travail exécutées dans des instances lancées en mode EKS Auto. EC2 Si vous utilisez un cluster en mode mixte (composé à la fois de nœuds de travail EKS Auto et de nœuds de travail autres qu'EKS Auto), vos règles basées sur le DNS ne sont efficaces que dans les nœuds de travail en mode EKS Auto (instances EC2 gérées).

Validation de vos politiques DNS

- Utilisez des clusters intermédiaires qui reflètent la topologie du réseau de production pour les tests - Votre environnement intermédiaire doit reproduire l'architecture réseau, les dépendances externes et les modèles de connectivité de production afin de garantir des tests de politiques précis. Cela inclut les configurations VPC correspondantes, le comportement de résolution DNS et l'accès aux mêmes services externes dont vos charges de travail de production ont besoin.
- Mettez en œuvre des tests automatisés pour les chemins réseau critiques - Créez des tests automatisés qui valident la connectivité aux services externes essentiels dans le cadre de votre CI/CD pipeline. Ces tests doivent vérifier que les flux de trafic légitimes sont autorisés tandis que les connexions non autorisées sont bloquées, afin de valider en permanence que vos politiques réseau maintiennent le bon niveau de sécurité au fur et à mesure de l'évolution de votre infrastructure.
- Surveillez le comportement des applications après les modifications des politiques : après avoir déployé des politiques réseau nouvelles ou modifiées en production, surveillez de près les journaux des applications, les taux d'erreur et les indicateurs de performance pour identifier rapidement les problèmes de connectivité. Établissez des procédures d'annulation claires afin de pouvoir annuler rapidement les modifications de politique si elles entraînent un comportement inattendu des applications ou des interruptions de service.

Interaction avec le pare-feu DNS Amazon Route 53

Les politiques d'administration et de réseau d'EKS sont d'abord évaluées au niveau du module lorsque le trafic est initié. Si une politique réseau EKS autorise la sortie vers un domaine spécifique, le pod exécute alors une requête DNS qui atteint le résolveur Route 53. À ce stade, les règles du pare-feu DNS Route 53 sont évaluées. Si le pare-feu DNS bloque la requête de domaine, la résolution DNS échoue et la connexion ne peut pas être établie, même si la politique réseau EKS l'autorise. Cela crée des couches de sécurité complémentaires : les politiques réseau basées sur le DNS d'EKS fournissent un contrôle des sorties au niveau du pod pour répondre aux exigences d'accès spécifiques aux applications et aux limites de sécurité mutualisées, tandis que le pare-feu DNS fournit une protection à l'échelle du VPC contre les domaines malveillants connus et applique des listes de blocage à l'échelle de l'organisation.

Baliser les sous-réseaux pour le mode automatique EKS

Si vous utilisez la fonctionnalité d'équilibrage de charge du mode automatique d'EKS, vous devez ajouter des AWS balises à vos sous-réseaux VPC.

Contexte

Ces balises identifient les sous-réseaux comme étant associés au cluster et, plus important encore, indiquent si le sous-réseau est public ou privé.

Les sous-réseaux publics ont un accès direct à Internet via une passerelle Internet. Ils sont utilisés pour les ressources qui doivent être accessibles au public, telles que les équilibreurs de charge.

Les sous-réseaux privés n'ont pas d'accès direct à Internet et utilisent des passerelles NAT pour le trafic sortant. Ils sont utilisés pour les ressources internes telles que les nœuds EKS qui n'ont pas besoin d'être publics IPs.

Pour en savoir plus sur les passerelles NAT et les passerelles Internet, consultez Connecter votre VPC à d'autres réseaux dans le Guide de l'utilisateur Amazon Virtual Private Cloud (VPC).

Exigence

À l'heure actuelle, les sous-réseaux utilisés pour l'équilibrage de charge par le mode automatique EKS doivent être balisés avec l'une des balises suivantes.

Sous-réseaux publics

Les sous-réseaux publics sont utilisés pour les équilibreurs de charge connectés à Internet. Ces sous-réseaux doivent comporter les balises suivantes :

Clé	Valeur
`kubernetes.io/role/elb`	1 ou ``

Sous-réseaux privés

Les sous-réseaux privés sont utilisés pour les équilibreurs de charge internes. Ces sous-réseaux doivent comporter les balises suivantes :

Clé	Valeur
`kubernetes.io/role/internal-elb`	1 ou ``

Procédure

Avant de commencer, identifiez les sous-réseaux publics (avec accès à une passerelle Internet) et les sous-réseaux privés (utilisant une passerelle NAT). Vous aurez besoin d'autorisations pour modifier les ressources VPC.

AWS Management Console

1. Ouvrez la console Amazon VPC et accédez à Subnets.
2. Sélectionnez le sous-réseau à étiqueter.
3. Choisissez l'onglet Tags, puis sélectionnez Ajouter un tag.
4. Ajoutez la balise appropriée :
   - Pour les sous-réseaux publics : Key=`kubernetes.io/role/elb`
   - Pour les sous-réseaux privés : Key=`kubernetes.io/role/internal-elb`
5. Définissez la valeur sur 1 ou laissez le champ vide.
6. Enregistrez et répétez l'opération pour les sous-réseaux restants.

AWS CLI

Pour les sous-réseaux publics :

```
aws ec2 create-tags \
    --resources subnet-ID \
    --tags Key=kubernetes.io/role/elb,Value=1
```

Pour les sous-réseaux privés :

```
aws ec2 create-tags \
    --resources subnet-ID \
    --tags Key=kubernetes.io/role/internal-elb,Value=1
```

Remplacez `subnet-ID` par votre ID de sous-réseau réel.

Génération de rapports de conformité CIS à partir des nœuds Kubernetes à l'aide de kubectl debug

Cette rubrique explique comment générer des rapports de conformité CIS (Center for Internet Security) pour les nœuds Amazon EKS à l'aide de la commande `kubectl debug`. Cette commande

permet de créer temporairement un conteneur de débogage sur un nœud Kubernetes et d'exécuter les vérifications CIS à l'aide de l'outil `apiclient`. L'outil `apiclient` fait partie de l'OS Bottlerocket, utilisé par les nœuds du mode automatique EKS.

Conditions préalables

Avant de commencer, assurez-vous de disposer des éléments suivants :

- Accès à un cluster Amazon EKS avec la commande `kubectl` configurér (la version installée doit être au minimum v1.32.0 ; tapez `kubectl version` pour vérifier).
- Autorisations IAM appropriées pour déboguer les nœuds.
- Profil valide autorisant les opérations de débogage (par exemple, `sysadmin`).

Pour plus d'informations sur l'utilisation de profils de débogage avec `kubectl`, consultez la section Débogage d'un pod ou d'un nœud lors de l'application d'un profil dans la documentation de Kubernetes.

Procédure

1. Déterminez l'ID d' AWS instance du nœud sur lequel vous souhaitez exécuter le rapport. Utilisez la commande suivante pour afficher la liste des nœuds du cluster. L'ID d'instance se trouve dans la colonne du nom et commence par `i-` :

   ```
   kubectl get nodes
   ```

   ```
   NAME                  STATUS   ROLES    AGE   VERSION
   i-0ea0ba0f8ef9ad609   Ready    <none>   62s   v1.30.10-eks-1a9dacd
   ```

2. Exécutez la commande ci-dessous en remplaçant `<instance-id>` par l'ID d'instance du nœud ciblé :

   ```
   kubectl debug node/<instance-id> -it --profile=sysadmin --image=public.ecr.aws/
   amazonlinux/amazonlinux:2023 -- bash -c "yum install -q -y util-linux-core; nsenter -
   t 1 -m apiclient report cis --level 1 --format text"
   ```

   Les composants de cette commande sont les suivants :

   - `kubectl debug node/<instance-id>` : crée une session de débogage sur l'ID d'instance EC2 spécifié.

- `-it` : alloue un TTY (shell de ligne de commande) et maintient stdin ouvert pour une utilisation interactive.
- `--profile=sysadmin` : utilise le profil `kubectl` spécifié avec les autorisations appropriées.
- `--image=public.ecr.aws/amazonlinux/amazonlinux:2023` : utilise `amazonlinux:2023` comme image de conteneur pour le débogage.
- `bash -c "…"` : exécute les commandes suivantes dans un shell bash :
  - `yum install -q -y util-linux-core` : installe discrètement le package d'utilitaires requis.
  - `nsenter -t 1 -m` : exécute `nsenter` pour accéder à l'espace de noms du processus hôte (PID 1).
  - `apiclient report cis --level 1 --format text` : exécute le rapport de conformité CIS au niveau 1, avec un résultat au format texte.

3. Passez en revue le résultat au format texte du rapport.

Interprétation du résultat

La commande génère un rapport au format texte indiquant l'état de conformité des différents contrôles CIS. Le résultat comprend :

- Contrôle CIS individuel IDs
- Description de chaque contrôle
- État Réussite, Échec ou Ignoré pour chaque vérification
- Détails expliquant les éventuels problèmes de conformité

Voici un exemple de résultat du rapport exécuté sur une instance de Bottlerocket :

```
Benchmark name:  CIS Bottlerocket Benchmark
Version:         v1.0.0
Reference:       https://www.cisecurity.org/benchmark/bottlerocket
Benchmark level: 1
Start time:      2025-04-11T01:40:39.055623436Z

[SKIP] 1.2.1     Ensure software update repositories are configured (Manual)
[PASS] 1.3.1     Ensure dm-verity is configured (Automatic)[PASS] 1.4.1     Ensure
 setuid programs do not create core dumps (Automatic)
```

```
[PASS] 1.4.2     Ensure address space layout randomization (ASLR) is enabled
 (Automatic)
[PASS] 1.4.3     Ensure unprivileged eBPF is disabled (Automatic)
[PASS] 1.5.1     Ensure SELinux is configured (Automatic)
[SKIP] 1.6       Ensure updates, patches, and additional security software are
 installed (Manual)
[PASS] 2.1.1.1   Ensure chrony is configured (Automatic)
[PASS] 3.2.5     Ensure broadcast ICMP requests are ignored (Automatic)
[PASS] 3.2.6     Ensure bogus ICMP responses are ignored (Automatic)
[PASS] 3.2.7     Ensure TCP SYN Cookies is enabled (Automatic)
[SKIP] 3.4.1.3   Ensure IPv4 outbound and established connections are configured
 (Manual)
[SKIP] 3.4.2.3   Ensure IPv6 outbound and established connections are configured
 (Manual)
[PASS] 4.1.1.1   Ensure journald is configured to write logs to persistent disk
 (Automatic)
[PASS] 4.1.2     Ensure permissions on journal files are configured (Automatic)

Passed:          11
Failed:          0
Skipped:         4
Total checks:    15
```

Pour plus d'informations sur le référentiel, consultez Référentiel Kubernetes dans la documentation du Center for Internet Security (CIS).

Ressources connexes

- Référence CIS Bottlerocket dans la documentation du système d'exploitation Bottlerocket.
- Débogage des pods en cours d'exécution dans la documentation de Kubernetes.
- Référentiel Kubernetes dans la documentation du Center for Internet Security (CIS)

Activation du chiffrement des volumes EBS avec des clés KMS gérées par le client pour le mode automatique EKS

Vous pouvez chiffrer le volume racine éphémère des instances gérées par le mode automatique EKS à l'aide d'une clé KMS gérée par le client.

Le mode automatique Amazon EKS utilise des rôles liés à des services pour déléguer des autorisations à d'autres AWS services lors de la gestion de volumes EBS chiffrés pour vos clusters

Kubernetes. Cette rubrique explique comment configurer la stratégie de clé requise lorsque vous spécifiez une clé gérée par le client pour le chiffrement Amazon EBS avec le mode automatique EKS.

Considérations :

- Le mode automatique EKS n'a pas besoin d'autorisation supplémentaire pour utiliser la clé AWS gérée par défaut afin de protéger les volumes chiffrés de votre compte.
- Cette rubrique traite du chiffrement des volumes éphémères, les volumes racines pour les instances. EC2 Pour plus d'informations sur le chiffrement des volumes de données utilisés pour les charges de travail, consultez the section called "Créez StorageClass".

Présentation de

Les clés AWS KMS suivantes peuvent être utilisées pour le chiffrement du volume racine Amazon EBS lorsque le mode automatique EKS lance des instances :

- Clé gérée par AWS : clé de chiffrement créée, détenue et gérée par Amazon EBS dans votre compte. Il s'agit de la clé de chiffrement par défaut d'un nouveau compte.
- Clé gérée par le client : clé de chiffrement personnalisée que vous créez, détenez et gérez vous-même.

> **Note**
>
> La clé doit être symétrique. Amazon EBS ne prend pas en charge les clés asymétriques gérées par le client.

Étape 1 : configurer la stratégie de clé

Vos clés KMS doivent inclure une stratégie de clé permettant au mode automatique EKS de lancer des instances avec des volumes Amazon EBS chiffrés à l'aide d'une clé gérée par le client.

Configurez votre stratégie de clé selon la structure suivante :

> **Note**
>
> Cette stratégie inclut uniquement les autorisations nécessaires au mode automatique EKS. La stratégie de clé peut nécessiter des autorisations supplémentaires si d'autres identités ont besoin d'utiliser la clé ou de gérer des autorisations.

```
{
    "Version":"2012-10-17",
    "Id": "MyKeyPolicy",
    "Statement": [
        {
            "Sid": "Allow use of the key",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::123456789012:role/ClusterServiceRole"
                ]
            },
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*",
                "kms:DescribeKey"
            ],
            "Resource": "*"
        },
        {
            "Sid": "Allow attachment of persistent resources",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::123456789012:role/ClusterServiceRole"
                ]
            },
            "Action": [
                "kms:CreateGrant",
                "kms:ListGrants",
                "kms:RevokeGrant"
            ],
            "Resource": "*",
```

```
            "Condition": {
                "Bool": {
                    "kms:GrantIsForAWSResource": "true"
                }
            }
        }
    ]
}
```

Assurez-vous de le remplacer <account-id> par votre numéro de AWS compte actuel.

Lors de la configuration de la stratégie de clé :

- Le ClusterServiceRole doit disposer des autorisations IAM nécessaires pour utiliser la clé KMS lors des opérations de chiffrement
- Cette kms:GrantIsForAWSResource condition garantit que les subventions ne peuvent être créées que pour AWS des services

Étape 2 : Configuration à l' NodeClass aide de votre clé gérée par le client

Après avoir configuré la politique des clés, référencez la clé KMS dans votre NodeClass configuration du mode automatique EKS :

```
apiVersion: eks.amazonaws.com/v1
kind: NodeClass
metadata:
  name: my-node-class
spec:
  # Insert existing configuration

  ephemeralStorage:
    size: "80Gi"  # Range: 1-59000Gi or 1-64000G or 1-58Ti or 1-64T
    iops: 3000    # Range: 3000-16000
    throughput: 125  # Range: 125-1000

    # KMS key for encryption
    kmsKeyID: "arn:aws: kms:<region>:<account-id>:key/<key-id>"
```

Remplacez les valeurs d'exemple par vos valeurs réelles :

- <region>avec votre AWS région

- `<account-id>`avec votre identifiant AWS de compte
- `<key-id>` par l'ID de votre clé KMS

Vous pouvez spécifier la clé KMS selon l'un des formats suivants :

- ID de clé KMS : `1a2b3c4d-5e6f-1a2b-3c4d-5e6f1a2b3c4d`
- ARN de clé KMS : `arn:aws: kms:us-west-2:111122223333:key/1a2b3c4d-5e6f-1a2b-3c4d-5e6f1a2b3c4d`
- Nom d'alias de clé : `alias/eks-auto-mode-key`
- ARN d'alias de clé : `arn:aws: kms:us-west-2:111122223333:alias/eks-auto-mode-key`

Appliquez la NodeClass configuration à l'aide de kubectl :

```
kubectl apply -f nodeclass.yaml
```

Ressources connexes

- Création d'une classe de nœuds pour Amazon EKS
- Pour plus d'informations, consultez le guide du développeur du service de gestion des AWS clés
  - Autorisations pour les AWS services dans les politiques clés
  - Modification d'une stratégie de clé
  - Subventions dans AWS KMS

Mise à jour des contrôles d'organisation pour le mode automatique EKS

Certains contrôles organisationnels peuvent empêcher le mode automatique EKS de fonctionner correctement. Dans ce cas, vous devez mettre à jour ces contrôles afin de permettre au mode automatique EKS de disposer des autorisations nécessaires pour gérer les instances EC2 en votre nom.

Le mode automatique EKS utilise un rôle de service pour lancer les instances EC2 qui constituent les nœuds du mode automatique EKS. Un rôle de service est un rôle IAM créé dans votre compte, qu'un service peut assumer afin d'effectuer des actions en votre nom. Les politiques de contrôle des services (SCPs) s'appliquent toujours aux actions effectuées avec des rôles de service. Ainsi, une

SCP peut bloquer certaines opérations du mode automatique EKS. L'occurrence la plus courante se produit lorsqu'un SCP est utilisé pour restreindre les Amazon Machine Images (AMIs) qui peuvent être lancées. Pour permettre au mode automatique EKS de fonctionner, modifiez le SCP afin de permettre le lancement à AMIs partir des comptes du mode automatique EKS.

Vous pouvez également utiliser la AMIs fonctionnalité EC2 Allowed pour limiter la visibilité sur AMIs d'autres comptes. Si vous utilisez cette fonctionnalité, vous devez élargir les critères d'images afin d'inclure également les comptes d'AMI en mode automatique EKS dans les régions concernées.

Exemple de SCP pour tout bloquer AMIs sauf le mode automatique EKS AMIs

L'exemple ci-dessous empêche l'appel de `ec2:RunInstances` sauf si l'AMI appartient au compte d'AMI en mode automatique EKS pour us-west-2 ou us-east-1.

> **Note**
>
> Il est important de ne pas utiliser la clé de contexte `ec2:Owner`. Amazon est propriétaire des comptes AMI en mode automatique EKS et la valeur de cette clé sera toujours `amazon`. La construction d'un SCP qui permet le `ec2:Owner` lancement AMIs s'il permet de lancer n'importe quelle AMI appartenant à Amazon, et pas seulement celles destinées au mode automatique d'EKS. `amazon` *

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "DenyAMI",
      "Effect": "Deny",
      "Action": "ec2:RunInstances",
      "Resource": "arn:*:ec2:*::image/ami-*",
      "Condition": {
        "StringNotEquals": {
          "aws:ResourceAccount": [
            "767397842682",
            "992382739861"
          ]
        }
      }
    }
  ]
```

```
}
```

Comptes AMI en mode automatique EKS

AWS les comptes qui varient selon les régions hébergent le mode automatique EKS public AMIs.

AWS Région	Compte
af-south-1	471112993317
ap-east-1	590183728416
ap-east-2	381492200852
ap-northeast-1	851725346105
ap-northeast-2	992382805010
ap-northeast-3	891377407544
ap-south-1	975049899075
ap-south-2	590183737426
ap-southeast-1	339712723301
ap-southeast-2	058264376476
ap-southeast-3	471112941769
ap-southeast-4	590183863144
ap-southeast-5	654654202513
ap-southeast-6	905418310314
ap-southeast-7	533267217478
ca-central-1	992382439851
ca-west-1	767397959864

eu-central-1	891376953411
eu-central-2	381492036002
eu-north-1	339712696471
eu-south-1	975049955519
eu-south-2	471112620929
eu-west-1	381492008532
eu-west-2	590184142468
eu-west-3	891376969258
il-central-1	590183797093
me-central-1	637423494195
me-south-1	905418070398
mx-central-1	211125506622
sa-east-1	339712709251
us-east-1	992382739861
us-east-2	975050179949
us-west-1	975050035094
us-west-2	767397842682
us-gov-east-1	446077414359
us-gov-west-1	446098668741

Association d'une adresse IP publique

Lorsque l'appel `ec2:RunInstances` est effectué, la valeur du champ `AssociatePublicIpAddress` lors du lancement d'une instance est déterminée automatiquement en fonction du type de sous-réseau dans lequel l'instance est lancée. Une SCP peut être utilisée pour imposer que cette valeur soit explicitement définie sur « false », quel que soit le sous-réseau utilisé pour le lancement. Dans ce cas, le NodeClass champ `spec.advancedNetworking.associatePublicIPAddress` peut également être défini sur false pour satisfaire aux exigences du SCP.

```
{
      "Sid": "DenyPublicEC2IPAddesses",
      "Effect": "Deny",
      "Action": "ec2:RunInstances",
      "Resource": "arn:aws:ec2:*:*:network-interface/*",
      "Condition": {
          "BoolIfExists": {
              "ec2:AssociatePublicIpAddress": "true"
          }
      }
  }
```

Contrôle du déploiement des charges de travail dans les réserves de capacité avec le mode automatique EKS

Vous pouvez contrôler le déploiement des charges de travail dans des réserves de capacité. Le mode automatique d'EKS prend en charge les réservations de capacité EC2 à la demande (ODCRs) et les blocs de EC2 capacité pour le ML.

> **Tip**
>
> Par défaut, le mode automatique d'EKS se lance automatiquement dans les blocs de capacité ouverts ODCRs et ML. Lors de l'utilisation `capacityReservationSelectorTerms` dans la NodeClass définition, le mode automatique d'EKS n'utilisera plus automatiquement les réservations de capacité ouvertes.

EC2 Réservations de capacité à la demande (ODCRs)

EC2 Les réservations de capacité à la demande (ODCRs) vous permettent de réserver de la capacité de calcul pour vos EC2 instances Amazon dans une zone de disponibilité spécifique, quelle que soit la durée. Lorsque vous utilisez le mode automatique EKS, vous pouvez vouloir contrôler le déploiement de vos charges de travail Kubernetes sur ces instances réservées afin de maximiser l'utilisation de la capacité préachetée ou garantir l'accès des charges de travail critiques à des ressources dédiées.

Par défaut, le mode automatique EKS démarre automatiquement en mode ouvert ODCRs. Cependant, en configurant `capacityReservationSelectorTerms` sur un NodeClass, vous pouvez contrôler explicitement les charges de travail utilisées par ODCRs vos charges de travail. Les nœuds approvisionnés à l'aide de la configuration ODCRs auront `karpenter.sh/capacity-type: reserved` et seront priorisés par rapport à la demande et au spot. Une fois cette fonctionnalité activée, le mode automatique d'EKS n'utilisera plus automatiquement les options ouvertes. ODCRs Elles doivent être explicitement sélectionnées par un NodeClass, ce qui vous permet de contrôler précisément l'utilisation des réservations de capacité au sein de votre cluster.

> **Warning**
>
> Si vous effectuez une configuration `capacityReservationSelectorTerms` NodeClass dans un cluster, le mode automatique d'EKS n'utilisera plus automatiquement l'option open ODCRs pour aucun NodeClass élément du cluster.

Exemple NodeClass

```
apiVersion: eks.amazonaws.com/v1
kind: NodeClass
spec:
  # Optional: Selects upon on-demand capacity reservations and capacity blocks
  # for EKS Auto Mode to prioritize.
  capacityReservationSelectorTerms:
    - id: cr-56fac701cc1951b03
    # Alternative Approaches
    - tags:
        app: "my-app"
      # Optional owning account ID filter
      owner: "012345678901"
```

Cet exemple NodeClass illustre deux approches de sélection ODCRs. La première méthode fait référence directement à une ODCR spécifique à l'aide de son ID (`cr-56fac701cc1951b03`). La seconde méthode utilise la sélection basée sur les balises, le ciblage à l' ODCRs aide de la balise`Name: "targeted-odcr"`. Vous pouvez également éventuellement filtrer en fonction du AWS compte propriétaire de la réservation, ce qui est particulièrement utile dans les scénarios entre comptes ou lorsque vous travaillez avec des réservations à capacité partagée.

EC2 Blocs de capacité pour ML

Les blocs de capacité pour ML vous permettent de réserver à l'avance des instances de calcul accéléré basées sur GPU pour exécuter des charges de travail de machine learning (ML) de courte durée. Les instances qui s'exécutent au sein d'un bloc de capacité sont automatiquement placées à proximité les unes des autres dans Amazon EC2 UltraClusters, pour une mise en réseau non bloquante à faible latence, à l'échelle du pétaoctet.

Pour plus d'informations sur les plateformes et les types d'instances pris en charge, consultez Capacity Blocks for ML dans le guide de EC2 l'utilisateur.

Vous pouvez créer un mode automatique EKS NodeClass qui utilise un bloc de capacité pour le ML, similaire à un ODCR (décrit précédemment).

Les exemples de définitions suivants créent trois ressources :

1. A NodeClass qui fait référence à votre réservation Capacity Block
2. Un NodePool qui utilise le NodeClass et applique une teinte
3. Une spécification de pod tolérant cette balise de rejet et demandant des ressources GPU

Exemple NodeClass

Cela NodeClass fait référence à un bloc de capacité spécifique pour ML par son ID de réservation. Vous pouvez obtenir cet identifiant depuis la EC2 console.

```
apiVersion: eks.amazonaws.com/v1
kind: NodeClass
metadata:
  name: gpu
spec:
  # Specify your Capacity Block reservation ID
  capacityReservationSelectorTerms:
    - id: cr-56fac701cc1951b03
```

Pour de plus amples informations, veuillez consulter the section called "Création d'une classe de nœuds".

Exemple NodePool

Cela NodePool fait référence à la configuration importante gpu NodeClass et précise :

- Il utilise uniquement la capacité réservée en définissant `karpenter.sh/capacity-type: reserved`
- Il demande des familles d'instances GPU spécifiques adaptées aux charges de travail ML
- Il applique une balise de rejet `nvidia.com/gpu` pour s'assurer que seules les charges de travail GPU sont planifiées sur ces nœuds

```
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: gpu
spec:
  template:
    spec:
      nodeClassRef:
        group: eks.amazonaws.com
        kind: NodeClass
        name: gpu
      requirements:
        - key: eks.amazonaws.com/instance-family
          operator: In
          values:
            - g6
            - p4d
            - p4de
            - p5
            - p5e
            - p5en
            - p6
            - p6-b200
        - key: karpenter.sh/capacity-type
          operator: In
          values:
            - reserved
            # Enable other capacity types
```

```
            # - on-demand
            # - spot
      taints:
        - effect: NoSchedule
          key: nvidia.com/gpu
```

Pour de plus amples informations, veuillez consulter the section called "Création d'un groupe de nœuds".

Exemple de pod

Cet exemple de pod montre comment configurer une charge de travail afin qu'elle s'exécute sur vos nœuds de bloc de capacité :

- Il utilise un NodeSelector pour cibler des types de GPU spécifiques (dans ce cas, le H200) GPUs
- Il inclut une tolérance à l'égard de la `nvidia.com/gpu` souillure appliquée par le NodePool
- Il demande explicitement des ressources GPU en utilisant le type de ressource `nvidia.com/gpu`

```
apiVersion: v1
kind: Pod
metadata:
  name: nvidia-smi
spec:
  nodeSelector:
    # Select specific GPU type - uncomment as needed
    # eks.amazonaws.com/instance-gpu-name: l4
    # eks.amazonaws.com/instance-gpu-name: a100
    eks.amazonaws.com/instance-gpu-name: h200
    # eks.amazonaws.com/instance-gpu-name: b200
    eks.amazonaws.com/compute-type: auto
  restartPolicy: OnFailure
  containers:
  - name: nvidia-smi
    image: public.ecr.aws/amazonlinux/amazonlinux:2023-minimal
    args:
    - "nvidia-smi"
    resources:
      requests:
        # Uncomment if needed
        # memory: "30Gi"
        # cpu: "3500m"
```

```
        nvidia.com/gpu: 1
      limits:
        # Uncomment if needed
        # memory: "30Gi"
        nvidia.com/gpu: 1
  tolerations:
  - key: nvidia.com/gpu
    effect: NoSchedule
    operator: Exists
```

Pour plus d'informations, consultez pods dans la documentation Kubernetes.

Ressources connexes

- Blocs de capacité pour le machine learning dans le guide de EC2 l'utilisateur Amazon
- Trouvez et achetez des Capacity Blocks dans le guide de EC2 l'utilisateur Amazon
- Gérez les ressources informatiques pour les AI/ML charges de travail sur Amazon EKS
- Optimisation et gestion des coûts des ressources GPU dans le Guide des bonnes pratiques EKS

Déployer des nœuds EKS Auto Mode sur des Zones Locales

Le mode automatique EKS simplifie la gestion des clusters grâce au provisionnement automatique des nœuds. AWS Les zones locales étendent AWS l'infrastructure à des emplacements géographiques plus proches de vos utilisateurs finaux, réduisant ainsi le temps de latence pour les applications sensibles à la latence. Ce guide explique le processus de déploiement des nœuds EKS Auto Mode sur les Zones AWS Locales, ce qui vous permet d'exécuter des applications conteneurisées avec une latence plus faible pour les utilisateurs de zones géographiques spécifiques.

Ce guide explique également comment utiliser les contraintes et les tolérances de Kubernetes pour garantir que seules des charges de travail spécifiques s'exécutent sur les nœuds de votre zone locale, ce qui vous aide à contrôler les coûts et à optimiser l'utilisation des ressources.

Conditions préalables

Avant de commencer à déployer des nœuds EKS Auto Mode sur des Zones Locales, assurez-vous que les conditions préalables suivantes sont réunies :

- Un cluster de mode automatique EKS existant

- Vous avez activé la zone locale dans votre compte AWS

Étape 1 : créer un sous-réseau de zone locale

La première étape du déploiement des nœuds du mode automatique EKS dans une zone locale consiste à créer un sous-réseau dans cette zone locale. Ce sous-réseau fournit l'infrastructure réseau pour vos nœuds et leur permet de communiquer avec le reste de votre VPC. Suivez les instructions de création d'un sous-réseau de zone locale (dans le guide de l'utilisateur des zones AWS locales) pour créer un sous-réseau dans la zone locale de votre choix.

> **Tip**
>
> Notez le nom du sous-réseau de votre zone locale.

Étape 2 : Création NodeClass pour le sous-réseau de zone locale

Après avoir créé votre sous-réseau de zone locale, vous devez définir un NodeClass qui fait référence à ce sous-réseau. NodeClass Il s'agit d'une ressource personnalisée Kubernetes qui spécifie les attributs d'infrastructure de vos nœuds, notamment les sous-réseaux, les groupes de sécurité et les configurations de stockage à utiliser. Dans l'exemple ci-dessous, nous créons une « zone locale » qui cible un sous-réseau de zone locale en fonction de son NodeClass nom. Vous pouvez également utiliser l'ID de sous-réseau. Vous devrez adapter cette configuration pour cibler votre sous-réseau de zone locale.

Pour de plus amples informations, veuillez consulter the section called “Création d’une classe de nœuds”.

```
apiVersion: eks.amazonaws.com/v1
kind: NodeClass
metadata:
  name: local-zone
spec:
  subnetSelectorTerms:
    - id: <local-subnet-id>
```

Étape 3 : créer NodePool avec NodeClass et teindre

Une fois que vous êtes NodeClass configuré, vous devez maintenant créer un NodePool qui l'utilise NodeClass. A NodePool définit les caractéristiques de calcul de vos nœuds, y compris les types d'instances. Il NodePool utilise le NodeClass comme référence pour déterminer où lancer les instances.

Dans l'exemple ci-dessous, nous créons un NodePool qui fait référence à notre « zone locale ». NodeClass Nous ajoutons également une touche aux nœuds afin de garantir que seuls les pods présentant une tolérance correspondante puissent être planifiés sur ces nœuds de zone locale. Cela est particulièrement important pour les nœuds de zone locale, qui ont généralement des coûts plus élevés et ne doivent être utilisés que par des charges de travail bénéficiant spécifiquement de la réduction de la latence.

Pour de plus amples informations, veuillez consulter the section called “Création d’un groupe de nœuds”.

```
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: my-node-pool
spec:
  template:
    metadata:
      labels:
        node-type: local-zone
    spec:
      nodeClassRef:
        group: eks.amazonaws.com
        kind: NodeClass
        name: local-zone
      taints:
        - key: "aws.amazon.com/local-zone"
          value: "true"
          effect: NoSchedule

      requirements:
        - key: "eks.amazonaws.com/instance-category"
          operator: In
          values: ["c", "m", "r"]
        - key: "eks.amazonaws.com/instance-cpu"
          operator: In
```

```
          values: ["4", "8", "16", "32"]
```

L'odeur associée à la touche `aws.amazon.com/local-zone` et à l'effet `NoSchedule` garantit que les pods sans tolérance correspondante ne seront pas planifiés sur ces nœuds. Cela permet d'éviter que des charges de travail régulières ne s'exécutent accidentellement dans la zone locale, ce qui pourrait entraîner des coûts imprévus.

Étape 4 : Déployer les charges de travail avec tolérance et affinité des nœuds

Pour un contrôle optimal du placement de la charge de travail sur les nœuds de zone locale, utilisez à la fois l'affinité taints/tolerations des nœuds et l'affinité des nœuds. Cette approche combinée offre les avantages suivants :

1. Contrôle des coûts : Cette altération garantit que seuls les pods soumis à des tolérances explicites peuvent utiliser des ressources de zone locale potentiellement coûteuses.
2. Placement garanti : l'affinité des nœuds garantit que vos applications sensibles à la latence s'exécutent exclusivement dans la zone locale, et non sur des nœuds de cluster ordinaires.

Voici un exemple de déploiement configuré pour s'exécuter spécifiquement sur les nœuds de zone locale :

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: low-latency-app
  namespace: default
spec:
  replicas: 2
  selector:
    matchLabels:
      app: low-latency-app
  template:
    metadata:
      labels:
        app: low-latency-app
    spec:
      tolerations:
      - key: "aws.amazon.com/local-zone"
        operator: "Equal"
        value: "true"
        effect: "NoSchedule"
```

```
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: "node-type"
                operator: "In"
                values: ["local-zone"]
      containers:
      - name: application
        image: my-low-latency-app:latest
        resources:
          limits:
            cpu: "1"
            memory: "1Gi"
          requests:
            cpu: "500m"
            memory: "512Mi"
```

Ce déploiement comporte deux configurations de planification clés :

1. La tolérance permet de planifier les capsules sur les nœuds présentant cette odeur`aws.amazon.com/local-zone`.
2. L'exigence d'affinité des nœuds garantit que ces pods ne s'exécuteront que sur les nœuds portant le label`node-type: local-zone`.

Ensemble, ils garantissent que votre application sensible à la latence ne s'exécute que sur les nœuds de zone locale, et que les applications ordinaires ne consomment pas les ressources de la zone locale, sauf si elles sont explicitement configurées pour le faire.

Étape 5 : vérification à l'aide de AWS la console

Après avoir configuré vos déploiements NodeClass NodePool, et,,,,,,,,,,, vous devez vérifier que les nœuds sont approvisionnés dans votre zone locale comme prévu et que vos charges de travail s'exécutent sur eux. Vous pouvez utiliser la console AWS de gestion pour vérifier que les EC2 instances sont lancées dans le sous-réseau de zone locale approprié.

En outre, vous pouvez consulter la liste des nœuds Kubernetes en utilisant `kubectl get nodes -o wide` pour vérifier que les nœuds rejoignent votre cluster avec les étiquettes et les caractéristiques appropriées :

```
kubectl get nodes -o wide
kubectl describe node <node-name> | grep -A 5 Taints
```

Vous pouvez également vérifier que vos pods de charge de travail sont planifiés sur les nœuds de la zone locale :

```
kubectl get pods -o wide
```

Cette approche garantit que seules les charges de travail qui tolèrent spécifiquement l'empreinte de zone locale seront planifiées sur ces nœuds, ce qui vous permet de contrôler les coûts et d'utiliser au mieux les ressources de votre zone locale.

Configuration des paramètres de sécurité avancés pour les nœuds

Cette rubrique explique comment configurer les paramètres de sécurité avancés pour les nœuds Amazon EKS Auto Mode à l'aide des `advancedSecurity` spécifications de votre classe de nœuds.

Conditions préalables

Avant de commencer, assurez-vous de disposer des éléments suivants :

- Un cluster du mode automatique Amazon EKS. Pour de plus amples informations, veuillez consulter the section called "Créer un cluster".
- `kubectl` installé et configuré. Pour de plus amples informations, veuillez consulter Configuration.
- Compréhension de la configuration des classes de nœuds. Pour de plus amples informations, veuillez consulter the section called "Création d'une classe de nœuds".

Configuration des paramètres de sécurité avancés

Pour configurer les paramètres de sécurité avancés pour vos nœuds, définissez les `advancedSecurity` champs dans la spécification de votre classe de nœud :

```
apiVersion: eks.amazonaws.com/v1
kind: NodeClass
metadata:
  name: security-hardened
spec:
```

```
  role: MyNodeRole

  subnetSelectorTerms:
    - tags:
        Name: "private-subnet"

  securityGroupSelectorTerms:
    - tags:
        Name: "eks-cluster-sg"

  advancedSecurity:
    # Enable FIPS-compliant AMIs (US regions only)
    fips: true

    # Configure kernel lockdown mode
    kernelLockdown: "integrity"
```

Appliquez cette configuration :

```
kubectl apply -f nodeclass.yaml
```

Référencez cette classe de nœuds dans la configuration de votre pool de nœuds. Pour de plus amples informations, veuillez consulter the section called “Création d'un groupe de nœuds”.

Descriptions des champs

- `fips`(booléen, facultatif) : lorsqu'il est défini sur`true`, approvisionne les nœuds à l'aide de modules cryptographiques validés AMIs par la norme FIPS 140-2. Ce paramètre sélectionne la conformité à la norme FIPS AMIs ; les clients sont responsables de la gestion de leurs exigences de conformité. Pour plus d'informations, consultez la section Conformité à la AWS norme FIPS. Valeur par défaut : `false`.
- `kernelLockdown`(chaîne, facultatif) : contrôle le mode du module de sécurité du verrouillage du noyau. Valeurs acceptées :
  - `integrity`: bloque les méthodes permettant de remplacer la mémoire du noyau ou de modifier le code du noyau. Empêche le chargement de modules de noyau non signés.
  - `none`: désactive la protection contre le verrouillage du noyau.

    Pour plus d'informations, consultez la documentation sur le verrouillage du noyau Linux.

Considérations

- Les produits conformes à la norme FIPS AMIs sont disponibles dans les AWS régions de l'Est/Ouest des États-Unis et du AWS Canada AWS GovCloud (Centre-Ouest). Pour plus d'informations, consultez la section Conformité à la AWS norme FIPS.
- Lors de l'utilisation`kernelLockdown: "integrity"`, assurez-vous que vos charges de travail ne nécessitent pas le chargement de modules de noyau non signés ou la modification de la mémoire du noyau.

Ressources connexes

- the section called "Création d'une classe de nœuds"- Guide complet de configuration des classes de nœuds
- the section called "Création d'un groupe de nœuds"- Configuration du pool de nœuds

Découvrez comment fonctionne le mode automatique EKS

Ce chapitre explique le fonctionnement des composants des clusters du mode automatique Amazon EKS.

Rubriques

- Informations sur les instances gérées par le mode automatique Amazon EKS
- Informations sur les identités et l'accès dans le mode automatique EKS
- Informations sur le réseau VPC et l'équilibrage de charge dans le mode automatique EKS

Informations sur les instances gérées par le mode automatique Amazon EKS

Cette rubrique explique comment le mode automatique Amazon EKS gère les EC2 instances Amazon dans votre cluster EKS. Lorsque vous activez le mode automatique EKS, les ressources de calcul de votre cluster sont automatiquement provisionnées et gérées par EKS, ce qui modifie la façon dont vous interagissez avec les EC2 instances qui servent de nœuds dans votre cluster.

Il est essentiel de comprendre comment le mode automatique Amazon EKS gère les instances afin de bien planifier votre stratégie de déploiement des charges de travail et vos procédures

opérationnelles. Contrairement aux EC2 instances traditionnelles ou aux groupes de nœuds gérés, ces instances suivent un modèle de cycle de vie différent dans lequel EKS assume la responsabilité de nombreux aspects opérationnels, tout en limitant certains types d'accès et de personnalisation.

Le mode automatique Amazon EKS automatise les tâches de routine pour créer de nouvelles EC2 instances et les attache en tant que nœuds à votre cluster EKS. Le mode automatique EKS détecte lorsqu'une charge de travail ne peut pas s'adapter aux nœuds existants et crée une nouvelle EC2 instance.

Le mode automatique Amazon EKS est chargé de créer, de supprimer et d'appliquer des correctifs aux EC2 instances. Vous restez responsable des conteneurs et des pods déployés sur ces instances.

EC2 Les instances créées par le mode automatique d'EKS sont différentes des autres EC2 instances, ce sont des instances gérées. Ces instances gérées sont détenues par EKS et présentent des restrictions supplémentaires. Vous ne pouvez ni accéder directement aux instances gérées par le mode automatique EKS, ni y installer de logiciels.

AWS suggère d'exécuter le mode automatique EKS ou Karpenter autogéré. Vous pouvez installer les deux durant une phase de migration ou dans une configuration avancée. Si vous avez installé les deux, configurez vos pools de nœuds afin que les charges de travail soient associées soit à Karpenter, soit au mode automatique EKS.

Pour plus d'informations, consultez les instances EC2 gérées par Amazon dans le guide de EC2 l'utilisateur Amazon.

Tableau comparatif

EC2 Instance standard	Instance gérée par le mode automatique EKS
Vous êtes responsable de l'application des correctifs et de la mise à jour de l'instance.	AWS corrige et met à jour automatiquement l'instance.
EKS n'est pas responsable des logiciels installés sur l'instance.	EKS est responsable de certains logiciels sur l'instance, tels que `kubelet`, le moteur d'exécution des conteneurs et le système d'exploitation.
Vous pouvez supprimer l' EC2 instance à l'aide de l' EC2 API.	EKS détermine le nombre d'instances déployées dans votre compte. Si vous

EC2 Instance standard	Instance gérée par le mode automatique EKS
	supprimez une charge de travail, EKS réduira le nombre d'instances dans votre compte.
Vous pouvez utiliser SSH pour accéder à l' EC2 instance.	Vous pouvez déployer des pods et des conteneurs sur l'instance gérée.
Vous déterminez le système d'exploitation et l'image (AMI).	AWS détermine le système d'exploitation et l'image.
Vous pouvez déployer des charges de travail reposant sur les fonctionnalités de Windows ou d'Ubuntu.	Vous pouvez également déployer des conteneurs basés sur Linux, sans dépendances particulières au système d'exploitation.
Vous déterminez le type et la famille d'instance à lancer.	AWS détermine le type et la famille d'instanc es à lancer. Vous pouvez utiliser un groupe de nœuds pour limiter les types d'instances parmi lesquels le mode automatique EKS peut sélectionner.

Les fonctionnalités suivantes fonctionnent à la fois pour les instances gérées et les EC2 instances standard :

- Vous pouvez afficher l'instance dans la AWS console.
- Vous pouvez utiliser le stockage d'instance comme stockage éphémère pour les charges de travail.

Support AMI

Avec le mode automatique EKS, AWS détermine l'image (AMI) utilisée pour vos nœuds de calcul. AWS surveille le déploiement des nouvelles versions de l'AMI en mode automatique d'EKS. Si vous rencontrez des problèmes de charge de travail liés à une version d'AMI, créez un cas de support. Pour plus d'informations, consultez les sections Création de dossiers de support et gestion de dossiers dans le Guide de l'utilisateur du AWS support.

En général, EKS publie chaque semaine une nouvelle AMI contenant des correctifs CVE et de sécurité.

Référence des instances prises en charge par le mode automatique EKS

Le mode automatique EKS crée uniquement des instances de types pris en charge et répondant à une taille minimale requise.

Le mode automatique EKS prend en charge les types d'instances suivants :

Family	Types d'instances
Optimisées pour le calcul (C)	c8i, c8i-flex, c8gd, c8gn, c8g, c7a, c7g, c7gn, c7gd, c7i, c7i-flex, c6a, c6g, c6i, c6gn, c6id, c5ad, c5n, 4
Usage général (M)	m8i, m8i-flex, m8a, m8gn, m8gb, m8gd, m8g, m7i, m7a, m7g, m7gd, m7i-flex, m6a, m6i, m6in, m6g, m6idn, m6id, m5, m5ad, m5n, m5dn, m5d, m5zn, m4
Optimisées pour la mémoire (R)	r8i, r8i-flex, r8gn, r8gb, r8g, r7a, r7iz, r7gd, r7i, r7g, r6a, r6i, r6id, r6in, r6idn, r6g, r6gd, r5, r5dn, r5b, r5b, 5ad, 5d, r4
Capacité extensible (T)	t4g, t3, t3a, t2
Mémoire élevée (Z/X)	z1d, x8g, x2gd
Optimisées pour le stockage (I/D)	i8ge, i7i, i8g, i7ie, i4g, i4i, i3, i3en, is4gen, d3, d3en, im4gn
Informatique accélérée (P/G/Inf/Trn)	p5, p4d, p4de, p3, p3dn, gr6, g6, g6e, g5g, g5, g4dn, inf2, inf1, trn1, trn1n
Calcul haute performance (X2)	x2iezn, x2iedn, x2idn

En outre, le mode automatique d'EKS ne créera que EC2 des instances répondant aux exigences suivantes :

- Plus d'un processeur

- Taille d'instance différente de nano, micro ou petite

Pour plus d'informations, consultez les conventions de dénomination des types d' EC2 instances Amazon.

Service des métadonnées d'instance

- Le mode automatique EKS est IMDSv2 appliqué avec une limite de saut de 1 par défaut, conformément aux meilleures pratiques AWS de sécurité.
- Cette configuration par défaut ne peut pas être modifiée en mode automatique.
- Pour les modules complémentaires qui nécessitent généralement un accès IMDS, fournissez des paramètres (tels que AWS la région) lors de l'installation afin d'éviter les recherches IMDS. Pour de plus amples informations, veuillez consulter the section called "Champs que vous pouvez personnaliser".
- Si un pod doit absolument accéder à IMDS lorsqu'il s'exécute en mode automatique, il doit être configuré pour s'exécuter avec `hostNetwork: true`. Cela permet au pod d'accéder directement au service de métadonnées de l'instance.
- Évaluez toujours les implications de sécurité avant d'autoriser un pod à accéder aux métadonnées de l'instance.

Pour plus d'informations sur le service de métadonnées d' EC2 instance Amazon (IMDS), consultez Configurer les options du service de métadonnées d'instance dans le guide de l' EC2 utilisateur Amazon.

Considérations

- Si le stockage éphémère configuré dans le NodeClass est plus petit que le stockage NVMe local de l'instance, le mode automatique d'EKS élimine le besoin de configuration manuelle en effectuant automatiquement les actions suivantes :
  - Utilise un volume de données Amazon EBS plus petit (20 Gio) afin de réduire les coûts.
  - Formate et configure le stockage NVMe local pour une utilisation éphémère des données. Cela inclut la configuration d'une matrice RAID 0 s'il existe plusieurs NVMe disques.
- Lorsque la NVMe capacité locale `ephemeralStorage.size` est égale ou supérieure, les actions suivantes se produisent :
  - Le mode automatique ignore le petit volume EBS.

  - Le NVMe ou les disques sont exposés directement à votre charge de travail.
- Le mode automatique d'Amazon EKS ne prend pas en charge les actions suivantes du service d'injection de AWS défauts :
  - `ec2:RebootInstances`
  - `ec2:SendSpotInstanceInterruptions`
  - `ec2:StartInstances`
  - `ec2:StopInstances`
  - `ec2:TerminateInstances`
  - `ec2:PauseVolumeIO`
- Le mode automatique d'Amazon EKS prend en charge les actions du Pod EKS du service d'injection de AWS défauts. Pour plus d'informations, consultez les sections Gestion des expériences du service d'injection de défauts et Utilisation des actions AWS FIS aws:eks:pod dans le guide de l' AWS utilisateur de Resilience Hub.
- Vous n'avez pas besoin d'installer le `Neuron Device Plugin` sur les nœuds du mode automatique EKS.

  Cependant, si votre cluster contient d'autres types de nœuds, vous devez configurer le plug-in Neuron Device pour qu'il ne s'exécute pas sur les nœuds du mode automatique. Pour de plus amples informations, veuillez consulter the section called "Contrôle du déploiement".

Informations sur les identités et l'accès dans le mode automatique EKS

Cette rubrique décrit les rôles et autorisations de gestion des identités et des accès (IAM) requis pour utiliser le mode automatique EKS. Le mode automatique EKS utilise deux rôles IAM principaux : un rôle IAM de cluster et un rôle IAM de nœud. Ces rôles fonctionnent conjointement avec l'identité du pod EKS et les entrées d'accès EKS pour assurer une gestion complète des accès aux clusters EKS.

Lorsque vous configurez le mode automatique d'EKS, vous devez configurer ces rôles IAM avec des autorisations spécifiques qui permettent aux AWS services d'interagir avec les ressources de votre cluster. Ces autorisations couvrent la gestion des ressources de calcul, des volumes de stockage, des équilibreurs de charge et des composants réseau. La compréhension de la configuration de ces rôles est essentielle pour garantir le bon fonctionnement et la sécurité du cluster.

En mode automatique d'EKS, les rôles AWS IAM sont automatiquement mappés aux autorisations Kubernetes via les entrées d'accès EKS, ce qui élimine le besoin de configuration manuelle ou de

liaisons personnalisées. `aws-auth` ConfigMaps Lorsque vous créez un nouveau cluster en mode automatique, EKS crée automatiquement les autorisations Kubernetes correspondantes à l'aide des entrées Access, garantissant ainsi que les AWS services et les composants du cluster disposent des niveaux d'accès appropriés au sein du système d'autorisation AWS et du système d'autorisation Kubernetes. Cette intégration automatisée réduit la complexité de la configuration et aide à prévenir les problèmes d'autorisations fréquemment rencontrés lors de la gestion des clusters EKS.

Rôle IAM du cluster

Le rôle Cluster IAM est un rôle AWS Identity and Access Management (IAM) utilisé par Amazon EKS pour gérer les autorisations pour les clusters Kubernetes. Ce rôle accorde à Amazon EKS les autorisations nécessaires pour interagir avec d'autres AWS services au nom de votre cluster et est automatiquement configuré avec des autorisations Kubernetes à l'aide des entrées d'accès EKS.

- Vous devez associer des politiques AWS IAM à ce rôle.
- Le mode automatique EKS attache automatiquement les autorisations Kubernetes à ce rôle via les entrées d'accès EKS.
- Avec le mode automatique d'EKS, AWS suggère de créer un seul rôle IAM de cluster par AWS compte.
- AWS suggère de nommer ce rôle`AmazonEKSAutoClusterRole`.
- Ce rôle nécessite des autorisations pour plusieurs AWS services afin de gérer les ressources, notamment les volumes EBS, les Elastic Load Balancers et EC2 les instances.
- La configuration suggérée pour ce rôle inclut plusieurs politiques IAM AWS gérées, liées aux différentes fonctionnalités du mode automatique d'EKS.
  - `AmazonEKSComputePolicy`
  - `AmazonEKSBlockStoragePolicy`
  - `AmazonEKSLoadBalancingPolicy`
  - `AmazonEKSNetworkingPolicy`
  - `AmazonEKSClusterPolicy`

Pour plus d'informations sur le rôle IAM du cluster et les politiques IAM AWS gérées, voir :

- the section called " AWS politiques gérées"
- the section called "Rôle IAM du cluster"

Pour plus d'informations sur les autorisations d'accès Kubernetes, consultez :

- the section called "Vérification des stratégies d'accès"

Rôle IAM de nœud

Le rôle Node IAM est un rôle AWS Identity and Access Management (IAM) utilisé par Amazon EKS pour gérer les autorisations des nœuds de travail dans les clusters Kubernetes. Ce rôle accorde aux EC2 instances exécutées en tant que nœuds Kubernetes les autorisations nécessaires pour interagir avec les AWS services et les ressources, et est automatiquement configuré avec les autorisations RBAC Kubernetes à l'aide des entrées d'accès EKS.

- Vous devez associer des politiques AWS IAM à ce rôle.
- Le mode automatique EKS attache automatiquement ces autorisations Kubernetes RBAC à ce rôle via les entrées d'accès EKS.
- AWS suggère de nommer ce rôle`AmazonEKSAutoNodeRole`.
- Avec le mode automatique d'EKS, AWS suggère de créer un rôle IAM de nœud unique par AWS compte.
- Ce rôle dispose d'autorisations limitées. Les principales autorisations incluent la possibilité d'assumer un rôle d'identité du pod et l'autorisation d'extraire des images depuis ECR.
- AWS suggère les politiques IAM AWS gérées suivantes :
  - `AmazonEKSWorkerNodeMinimalPolicy`
  - `AmazonEC2ContainerRegistryPullOnly`

Pour plus d'informations sur le rôle IAM du cluster et les politiques IAM AWS gérées, voir :

- the section called " AWS politiques gérées"
- the section called "Rôle IAM de nœud"

Pour plus d'informations sur les autorisations d'accès Kubernetes, consultez :

- the section called "Vérification des stratégies d'accès"

Rôle lié à un service

Amazon EKS utilise un rôle lié à un service (Service-Linked Role, SLR) pour certaines opérations. Un rôle lié à un service est un type unique de rôle IAM lié directement à Amazon EKS. Les rôles liés à un service sont prédéfinis par Amazon EKS et incluent toutes les autorisations requises par le service pour appeler d'autres AWS services en votre nom.

AWS crée et configure automatiquement le reflex. Vous ne pouvez supprimer un SLR qu'après avoir supprimé les ressources associées. Vos ressources Amazon EKS sont ainsi protégées, car vous ne pouvez pas involontairement supprimer l'autorisation d'accéder aux ressources.

La politique SLR accorde à Amazon EKS l'autorisation d'observer et de supprimer les principaux composants de l'infrastructure : EC2 ressources (instances, interfaces réseau, groupes de sécurité), ressources ELB (équilibreurs de charge, groupes cibles), CloudWatch fonctionnalités (journalisation et métriques) et rôles IAM avec le préfixe « eks ». Il permet également la mise en réseau de terminaux privés via l'association de VPC/hosted zones et inclut des autorisations pour la EventBridge surveillance et le nettoyage des ressources étiquetées EKS.

Pour en savoir plus, consultez :

- the section called "AWS politique gérée : Amazon EKSService RolePolicy"
- the section called "Autorisations du rôle lié à un service pour Amazon EKS"

AWS Balises personnalisées pour les ressources EKS Auto

Par défaut, les politiques gérées liées au mode automatique d'EKS n'autorisent pas l'application de balises définies par l'utilisateur aux AWS ressources provisionnées en mode automatique. Si vous souhaitez appliquer des balises définies par l'utilisateur aux AWS ressources, vous devez attribuer des autorisations supplémentaires au rôle IAM du cluster avec des autorisations suffisantes pour créer et modifier des balises sur les AWS ressources. Voici un exemple de politique qui autorisera un accès illimité au balisage :

Voir un exemple de stratégie de balisage personnalisée

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

```
            "Action": [
                "ec2:CreateFleet",
                "ec2:RunInstances",
                "ec2:CreateLaunchTemplate"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/eks:eks-cluster-name": "${aws:PrincipalTag/eks:eks-
cluster-name}"
                },
                "StringLike": {
                    "aws:RequestTag/eks:kubernetes-node-class-name": "*",
                    "aws:RequestTag/eks:kubernetes-node-pool-name": "*"
                }
            }
        },
        {
            "Sid": "Storage",
            "Effect": "Allow",
            "Action": [
                "ec2:CreateVolume",
                "ec2:CreateSnapshot"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:volume/*",
                "arn:aws:ec2:*:*:snapshot/*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/eks:eks-cluster-name": "${aws:PrincipalTag/eks:eks-
cluster-name}"
                }
            }
        },
        {
            "Sid": "Networking",
            "Effect": "Allow",
            "Action": "ec2:CreateNetworkInterface",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/eks:eks-cluster-name": "${aws:PrincipalTag/eks:eks-
cluster-name}"
```

```
                },
                "StringLike": {
                    "aws:RequestTag/eks:kubernetes-cni-node-name": "*"
                }
            }
        },
        {
            "Sid": "LoadBalancer",
            "Effect": "Allow",
            "Action": [
                "elasticloadbalancing:CreateLoadBalancer",
                "elasticloadbalancing:CreateTargetGroup",
                "elasticloadbalancing:CreateListener",
                "elasticloadbalancing:CreateRule",
                "ec2:CreateSecurityGroup"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/eks:eks-cluster-name": "${aws:PrincipalTag/eks:eks-
cluster-name}"
                }
            }
        },
        {
            "Sid": "ShieldProtection",
            "Effect": "Allow",
            "Action": [
                "shield:CreateProtection"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/eks:eks-cluster-name": "${aws:PrincipalTag/eks:eks-
cluster-name}"
                }
            }
        },
        {
            "Sid": "ShieldTagResource",
            "Effect": "Allow",
            "Action": [
                "shield:TagResource"
            ],
```

```
            "Resource": "arn:aws:shield::*:protection/*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/eks:eks-cluster-name": "${aws:PrincipalTag/eks:eks-
cluster-name}"
                }
            }
        }
    ]
}
```

Référence de stratégie d'accès

Pour plus d'informations sur les autorisations Kubernetes utilisées par le mode automatique EKS, consultez the section called "Vérification des stratégies d'accès".

Informations sur le réseau VPC et l'équilibrage de charge dans le mode automatique EKS

Cette rubrique explique comment configurer les fonctionnalités de réseau du cloud privé virtuel (VPC) et d'équilibrage de charge dans le mode automatique EKS. Même si le mode automatique EKS gère automatiquement la plupart des composants réseau, vous pouvez tout de même personnaliser certains aspects de la configuration réseau de votre cluster à l'aide de ressources `NodeClass` et d'annotations d'équilibreur de charge.

Lorsque vous utilisez le mode automatique EKS, AWS gère la configuration de l'interface réseau de conteneurs VPC (CNI) et le provisionnement de l'équilibreur de charge pour votre cluster. Vous pouvez influencer le comportement du réseau en définissant des objets `NodeClass` et en appliquant des annotations spécifiques à vos ressources Service et Ingress, tout en conservant le modèle opérationnel automatisé fourni par le mode automatique EKS.

Fonctionnalité de réseau

Le mode automatique EKS introduit une nouvelle fonctionnalité de réseau qui gère le réseau des nœuds et des pods. Vous pouvez la configurer en créant un objet Kubernetes `NodeClass`.

Les options de configuration du AWS VPC CNI précédent ne s'appliqueront pas au mode automatique EKS.

Configuration du réseau avec une `NodeClass`

La ressource `NodeClass` dans le mode automatique EKS permet de personnaliser certains aspects de la fonctionnalité de réseau. Grâce à `NodeClass`, vous pouvez définir la sélection des groupes de sécurité, contrôler le placement des nœuds dans les sous-réseaux du VPC ,définir des politiques SNAT, configurer des politiques réseau et activer la journalisation des événements réseau. Cette approche maintient le modèle opérationnel automatisé du mode automatique EKS, tout en offrant une flexibilité de personnalisation du réseau.

Vous pouvez utiliser une `NodeClass` pour :

- Sélectionner un groupe de sécurité pour les nœuds
- Contrôler la manière dont les nœuds sont placés sur les sous-réseaux VPC
- Définisser la politique SNAT du nœud sur `random` ou `disabled`
- Activer les politiques réseau Kubernetes, notamment :
  - Définir la politique réseau sur Refuser ou Autoriser par défaut
  - Activer la journalisation des événements réseau dans un fichier.
- Isoler le trafic des pods du trafic des nœuds en attachant des pods à différents sous-réseaux.

Découvrez comment créer un Amazon EKS NodeClass.

Considérations

Le mode automatique EKS prend en charge :

- Politiques réseau EKS.
- Les options `HostPort` et `HostNetwork` pour les pods Kubernetes.
- Nœuds et pods dans des sous-réseaux publics ou privés.
- Mise en cache des requêtes DNS sur le nœud.

Le mode automatique EKS ne prend pas en charge :

- Groupes de sécurité par pod (SGPP).
- Réseau personnalisé dans la `ENIConfig`. Possibilité de placer des pods dans plusieurs sous-réseaux ou de les isoler complètement du trafic des nœuds via the section called "Sélection des sous-réseaux pour les pods".

- Configurations Warm IP, Warm Prefix et Warm ENI.
- Configuration minimale des cibles IP.
- Autres configurations prises en charge par le AWS VPC CNI open source.
- Configurations de politique réseau telles que la personnalisation du temporisateur Conntrack (la valeur par défaut est 300 s).
- Exportation des journaux d'événements réseau vers CloudWatch.

Gestion des ressources réseau

Le mode automatique EKS gère les préfixes, l'adressage IP et la gestion de l'interface réseau en surveillant les NodeClass ressources pour les configurations réseau. Le service exécute automatiquement plusieurs opérations essentielles :

Délégation de préfixes

Le mode automatique d'EKS utilise par défaut la délégation de préfixes (préfixes /28) pour la mise en réseau des pods et gère un pool prédéfini de ressources IP qui évolue en fonction du nombre de pods planifiés. Lorsque la fragmentation du sous-réseau du pod est détectée, le mode automatique provisionne les adresses IP secondaires (/32). Grâce à cet algorithme de mise en réseau de pods par défaut, le mode automatique calcule le nombre maximum de pods par nœud en fonction du nombre de ENIs pods IPs pris en charge par type d'instance (en supposant le pire des cas de fragmentation). Pour plus d'informations sur Max ENIs et IPs par type d'instance, consultez la section Nombre maximal d'adresses IP par interface réseau dans le Guide de EC2 l'utilisateur. Les familles d'instances de nouvelle génération (Nitro v6 et versions ultérieures) ont généralement augmenté ENIs et IPs par type d'instance, et le mode automatique ajuste le calcul du nombre maximum de pods en conséquence.

Pour les IPv6 clusters, seule la délégation de préfixes est utilisée, et le mode automatique utilise toujours une limite maximale de 110 pods par nœud.

Gestion de la stabilisation

Le service implémente un pool de recharge pour les préfixes ou IPv4 les adresses secondaires qui ne sont plus utilisés. Une fois le temps de stabilisation expiré, ces ressources sont renvoyées au VPC. Cependant, si des pods réutilisent ces ressources pendant le temps de stabilisation, elles sont restaurées à partir du groupe de stabilisation.

IPv6 Support

Pour les IPv6 clusters, le mode automatique EKS fournit un `/80` IPv6 préfixe par nœud sur l'interface réseau principale.

Le service assure également une gestion et une récupération de mémoire appropriées de toutes les interfaces réseau.

Équilibrage de charge

Vous configurez les équilibreurs de charge AWS élastiques fournis par EKS Auto Mode à l'aide d'annotations sur les ressources de service et d'entrée.

Pour plus d'informations, consultez the section called "Création d'une classe de ressources Ingress" ou the section called "Créer un service".

Considérations relatives à l'équilibrage de charge avec le mode automatique EKS

- Le mode de ciblage par défaut est le mode IP, et non le mode instance.
- Le mode automatique EKS prend uniquement en charge le mode groupe de sécurité pour les équilibreurs de charge réseau Network Load Balancer.
- AWS ne prend pas en charge la migration des équilibreurs de charge du contrôleur d'équilibrage de AWS charge autogéré vers le mode automatique EKS.
- Le champ `networking.ingress.ipBlock` dans la spécification `TargetGroupBinding` n'est pas pris en charge.
- Si vos composants master utilisent des groupes de sécurité personnalisés (et non ceux suivant le modèle de nommage `eks-cluster-sg-`), le rôle de votre cluster doit disposer d'autorisations IAM supplémentaires. La politique gérée par EKS par défaut permet uniquement à EKS de modifier les groupes de sécurité nommés `eks-cluster-sg-`. Sans autorisation de modifier vos groupes de sécurité personnalisés, EKS ne peut pas ajouter les règles d'entrée requises permettant au ALB/NLB trafic d'atteindre vos modules.

Considérations relatives au CoreDNS

Le mode automatique EKS n'utilise pas le déploiement CoreDNS traditionnel pour fournir une résolution DNS au sein du cluster. Au lieu de cela, les nœuds en mode automatique utilisent CoreDNS s'exécutant en tant que service système directement sur chaque nœud. Si vous passez d'un cluster traditionnel en mode automatique, vous pouvez supprimer le déploiement CoreDNS de votre cluster une fois que vos charges de travail ont été déplacées vers les nœuds du mode automatique.

> **Important**
>
> Si vous prévoyez de gérer un cluster avec des nœuds en mode automatique et en mode non automatique, vous devez conserver le déploiement CoreDNS. Les nœuds non en mode automatique s'appuient sur les modules CoreDNS traditionnels pour la résolution DNS, car ils ne peuvent pas accéder au service DNS au niveau des nœuds fourni par le mode automatique.

Observabilité pour le mode automatique EKS

Consultez ce chapitre pour en savoir plus sur les options d'observabilité pour les clusters en mode automatique Amazon EKS.

Rubriques

- Journaux AWS de composants gérés par accès pour EKS Auto

Journaux AWS de composants gérés par accès pour EKS Auto

Vous pouvez accéder aux journaux des composants AWS gérés depuis le mode automatique d'EKS pour améliorer l'observabilité des opérations de votre cluster. Le mode automatique EKS prend en charge les journaux pour les sources suivantes :

- Mise à l'échelle automatique des ordinateurs - Karpenter
- Stockage par blocs - EBS CSI
- Équilibrage de AWS charge - Load Balancer Controller
- Mise en réseau des pods - Gestion des adresses IP VPC CNI

Les journaux peuvent être livrés à la destination de livraison de votre choix.

Lorsque vous créez un cluster EKS Auto, vous avez la possibilité d'activer la journalisation du plan de contrôle (serveur API, audit, authentificateur, gestionnaire de contrôleurs, planificateur). Les journaux des composants gérés par EKS Auto (tels que le calcul, le stockage par blocs, l'équilibrage de charge et l'IPAM) nécessitent une configuration séparée via la livraison des journaux.

Configuration de la diffusion des journaux

Pour configurer la livraison des journaux des composants AWS gérés pour votre cluster EKS Auto Mode, utilisez l'API Amazon CloudWatch Logs. Pour obtenir des instructions de configuration détaillées, consultez la section Activation de la journalisation à partir AWS des services dans le guide de l'utilisateur Amazon CloudWatch Logs. Chaque fonctionnalité du mode automatique peut être configurée en tant que source de diffusion individuelle de CloudWatch Vended Logs, ce qui vous permet de sélectionner les journaux auxquels vous souhaitez avoir accès.

Le mode automatique EKS prend en charge les types de journaux suivants :

- AUTO_MODE_COMPUTE_LOGS
- MODE AUTOMATIQUE BLOQUE LES JOURNAUX DE STOCKAGE
- LOAD_BALANCING_LOAD_LOGS EN MODE AUTOMATIQUE
- MODE AUTOMATIQUE IPAM LOGS

Utilisation d'Amazon CloudWatch APIs

La configuration de la journalisation nécessite trois étapes :

1. Créez une source de diffusion pour la fonctionnalité à l'aide de l' CloudWatch PutDeliverySource API
2. Créez une destination de livraison à l'aide de PutDeliveryDestination
3. Créez une livraison pour connecter la source et la destination à l'aide de CreateDelivery

Vous pouvez configurer les détails de la destination des journaux du mode automatique à l'aide de l' deliveryDestinationConfiguration objet de l' CloudWatch PutDeliveryDestination API. Il utilise l'ARN d'un groupe de CloudWatch logs, d'un bucket S3 ou d'un flux de diffusion Kinesis Data Firehose.

Vous pouvez configurer une seule fonctionnalité du mode automatique (source de livraison) pour envoyer des journaux vers plusieurs destinations en créant plusieurs livraisons. Vous pouvez également créer plusieurs livraisons pour configurer plusieurs sources de livraison afin d'envoyer des journaux vers la même destination de livraison.

Autorisations IAM

En fonction de la destination sélectionnée, vous devrez peut-être configurer des politiques ou des rôles IAM pour le groupe de CloudWatch journaux, le compartiment S3 et Kinesis Data Firehose afin

de garantir la transmission réussie des journaux. En outre, si vous envoyez des journaux entre AWS comptes, vous devez utiliser l' PutDeliveryDestinationPolicy API pour configurer une politique IAM autorisant la livraison à destination. Consultez la documentation sur les autorisations de CloudWatch Vended Logs pour plus d'informations.

Afficher vos journaux

Une fois la livraison des journaux configurée, les journaux seront livrés à la destination que vous avez spécifiée. La méthode d'accès aux journaux dépend du type de destination que vous avez choisi :

- CloudWatch Journaux : affichez les journaux dans la console CloudWatch Logs, utilisez les commandes AWS CLI ou effectuez des requêtes avec CloudWatch Logs Insights
- Amazon S3 : accédez aux journaux sous forme d'objets dans votre compartiment S3 via la console S3, la AWS CLI ou des outils d'analyse tels qu'Amazon Athena
- Amazon Data Firehose - Les journaux sont diffusés vers votre cible Firehose configurée (telle que S3, Service, Redshift OpenSearch , etc.)

Tarification

CloudWatch Les frais de livraison et de stockage des journaux s'appliquent en fonction de la destination de livraison que vous avez choisie. CloudWatch Vended Logs permet une livraison fiable et sécurisée des journaux avec AWS authentification et autorisation intégrées à un prix réduit par rapport aux CloudWatch journaux standard. Consultez la section Vended Logs de la page de CloudWatch tarification pour plus de détails.

Ressources connexes

- Journalisation de plan de contrôle d'Amazon EKS
- PutDeliverySource API dans la référence de l'API CloudWatch Logs
- PutDeliveryDestination API dans la référence de l'API CloudWatch Logs
- CreateDelivery API dans la référence de l'API CloudWatch Logs

Résolution des problèmes avec le mode automatique EKS

Avec le mode automatique d'EKS, vous AWS assumez une plus grande responsabilité pour les instances EC2 de votre AWS compte. EKS assume la responsabilité de l'exécution de conteneur

sur les nœuds, du système d'exploitation sur les nœuds et de certains contrôleurs. Cela inclut un contrôleur de stockage par blocs, un contrôleur d'équilibrage de charge et un contrôleur de calcul.

Vous devez utiliser AWS Kubernetes APIs pour dépanner les nœuds. Vous pouvez effectuer les actions suivantes :

- Utilisez une ressource `NodeDiagnostic` Kubernetes pour extraire les journaux des nœuds à l'aide de the section called "Agent de surveillance des nœuds". Pour les étapes détaillées, consultez the section called "Extraction des journaux de nœuds".
- Utilisez la commande AWS EC2 CLI `get-console-output` pour récupérer la sortie de console à partir des nœuds. Pour les étapes détaillées, consultez the section called "Obtenez la sortie de console d'une instance gérée EC2 à l'aide de la CLI AWS EC2".
- Utilisez les conteneurs de débogage Kubernetes pour extraire les journaux des nœuds. Pour les étapes détaillées, consultez the section called "Extraction des journaux des nœuds à l'aide des conteneurs de débogage et de la CLI `kubectl`".

> **Note**
>
> Le mode automatique EKS utilise des instances gérées EC2. Vous ne pouvez pas accéder directement à ces instances EC2, y compris par SSH.

Vous pouvez rencontrer les problèmes suivants, qui disposent de solutions spécifiques aux composants du mode automatique EKS :

- Pods bloqués dans l'état `Pending`, qui ne sont pas planifiés sur les nœuds du mode automatique. Pour les solutions, consultez the section called "Résolution des problèmes liés à l'échec de la planification des pods sur un nœud du mode automatique".
- Instances gérées EC2 qui ne rejoignent pas le cluster en tant que nœuds Kubernetes. Pour les solutions, consultez the section called "Résolution problèmes liés au fait qu'un nœud ne rejoint pas le cluster".
- Erreurs et problèmes liés aux `NodePools`, `PersistentVolumes` et `Services` utilisant les contrôleurs intégrés au mode automatique EKS. Pour les solutions, consultez the section called "Résolution des problèmes liés aux contrôleurs inclus dans le mode automatique".
- La sécurité renforcée des pods empêche le partage de volumes entre pods. Pour les solutions, consultez the section called "Partage des volumes entre les pods".

Vous pouvez utiliser les méthodes suivantes pour diagnostiquer et résoudre les problèmes liés aux composants du mode automatique EKS :

- the section called "Obtenez la sortie de console d'une instance gérée EC2 à l'aide de la CLI AWS EC2"
- the section called "Extraction des journaux des nœuds à l'aide des conteneurs de débogage et de la CLI `kubectl`"
- the section called "Afficher les ressources associées au mode automatique EKS dans la AWS console"
- the section called "Afficher les erreurs IAM dans votre AWS compte"
- the section called "Détection des problèmes de connectivité des nœuds avec l'outil VPC `Reachability Analyzer`"

Agent de surveillance des nœuds

Le mode automatique EKS inclut l'agent de surveillance des nœuds Amazon EKS. Vous pouvez utiliser cet agent pour consulter les informations de diagnostic et de débogage relatives aux nœuds. L'agent de surveillance des nœuds publie les `events` Kubernetes et les `conditions` du nœud. Pour de plus amples informations, veuillez consulter the section called "État du nœud".

Obtenez la sortie de console d'une instance gérée EC2 à l'aide de la CLI AWS EC2

Cette procédure permet de résoudre les problèmes survenant au démarrage ou au niveau du noyau.

Tout d'abord, déterminez l'ID de l'instance EC2 associée à votre charge de travail. Ensuite, utilisez la AWS CLI pour récupérer la sortie de la console.

1. Confirmez que `kubectl` est installé et connecté à votre cluster
2. (Facultatif) Utilisez le nom d'un déploiement Kubernetes pour afficher la liste des pods associés.

```
kubectl get pods -l app=<deployment-name>
```

3. Utilisez le nom du pod Kubernetes pour déterminer l'ID de l'instance EC2 du nœud associé.

```
kubectl get pod <pod-name> -o wide
```

4. Utilisez l'ID de l'instance EC2 pour extraire le résultat de la console.

```
aws ec2 get-console-output --instance-id <instance id> --latest --output text
```

Extraction des journaux des nœuds à l'aide des conteneurs de débogage et de la CLI **kubectl**

La méthode recommandée pour extraire les journaux d'un nœud du mode automatique EKS consiste à utiliser la ressource `NodeDiagnostic`. Pour les étapes, consultez the section called "Extraction des journaux de nœuds".

Cependant, vous pouvez également diffuser les journaux en temps réel à partir d'une instance en utilisant la commande `kubectl debug node`. Cette commande lance un nouveau pod sur le nœud à déboguer, avec lequel vous pouvez ensuite interagir.

1. Lancez un conteneur de débogage. La commande suivante utilise `i-01234567890123456` pour l'ID d'instance du nœud, `-it` pour allouer un `tty` et attacher `stdin` pour un usage interactif, et le profil `sysadmin` défini dans le fichier kubeconfig.

   ```
   kubectl debug node/i-01234567890123456 -it --profile=sysadmin --image=public.ecr.aws/
   amazonlinux/amazonlinux:2023
   ```

   L'exemple qui suit illustre un résultat.

   ```
   Creating debugging pod node-debugger-i-01234567890123456-nxb9c with container
    debugger on node i-01234567890123456.
   If you don't see a command prompt, try pressing enter.
   bash-5.2#
   ```

2. Depuis le shell, vous pouvez maintenant installer le `util-linux-core` qui fournit la commande `nsenter`. Utilisez `nsenter` pour entrer dans l'espace de noms de montage du PID 1 (`init`) sur l'hôte, puis exécutez la commande `journalctl` pour diffuser les journaux depuis `kubelet` :

   ```
   yum install -y util-linux-core
   nsenter -t 1 -m journalctl -f -u kubelet
   ```

Pour des raisons de sécurité, l'image de conteneur Amazon Linux n'installe pas de nombreux binaires par défaut. Vous pouvez utiliser la commande `yum whatprovides` pour identifier le package à installer afin d'obtenir le binaire correspondant.

```
yum whatprovides ps
```

```
Last metadata expiration check: 0:03:36 ago on Thu Jan 16 14:49:17 2025.
procps-ng-3.3.17-1.amzn2023.0.2.x86_64 : System and process monitoring utilities
Repo        : @System
Matched from:
Filename    : /usr/bin/ps
Provide    : /bin/ps

procps-ng-3.3.17-1.amzn2023.0.2.x86_64 : System and process monitoring utilities
Repo        : amazonlinux
Matched from:
Filename    : /usr/bin/ps
Provide    : /bin/ps
```

Afficher les ressources associées au mode automatique EKS dans la AWS console

Vous pouvez utiliser la AWS console pour afficher l'état des ressources associées à votre cluster EKS Auto Mode.

- Volumes EBS
  - Affichage des volumes du mode automatique EKS en recherchant la clé de balise `eks:eks-cluster-name`
- Équilibreurs de charge
  - Affichage des équilibreurs de charge du mode automatique EKS en recherchant la clé de balise `eks:eks-cluster-name`
- Instances EC2
  - Affichage des instances du mode automatique EKS en recherchant la clé de balise `eks:eks-cluster-name`

Afficher les erreurs IAM dans votre AWS compte

1. Accédez à la CloudTrail console
2. Choisissez « Historique des événements » dans le volet de navigation de gauche
3. Appliquez les filtres de codes d'erreur suivants :
   - AccessDenied
   - UnauthorizedOperation
   - InvalidClientTokenId

Recherchez les erreurs associées à votre cluster EKS. Utilisez les messages d'erreur pour mettre à jour les entrées d'accès EKS, le rôle IAM du cluster ou le rôle IAM du nœud. Vous devrez peut-être attacher une nouvelle politique à ces rôles pour leur accorder les autorisations nécessaires au mode automatique EKS.

Résolution des problèmes liés à l'échec de la planification des pods sur un nœud du mode automatique

Si des pods restent dans l'état `Pending` et ne sont pas planifiés sur un nœud du mode automatique, vérifiez si le manifeste de pod ou de déploiement contient un `nodeSelector`. Si un `nodeSelector` est présent, assurez-vous qu'il utilise `eks.amazonaws.com/compute-type: auto` pour être planifié sur les nœuds gérés par le mode automatique EKS. Pour plus d'informations sur les étiquettes de nœud utilisées par le mode automatique EKS, consultez the section called "Contrôle du déploiement".

Résolution problèmes liés au fait qu'un nœud ne rejoint pas le cluster

Le mode automatique EKS configure automatiquement les nouvelles instances EC2 avec les informations nécessaires pour rejoindre le cluster, notamment l'adresse de point de terminaison du cluster et l'autorité de certification (CA) du cluster. Cependant, certaines instances peuvent ne pas rejoindre le cluster EKS en tant que nœuds. Exécutez les commandes suivantes pour identifier les instances qui n'ont pas rejoint le cluster :

1. Exécutez `kubectl get nodeclaim` pour vérifier les `NodeClaims` dont l'état est `Ready = False`.

```
kubectl get nodeclaim
```

2. Exécutez `kubectl describe nodeclaim <node_claim>` et consultez la section État pour détecter tout problème empêchant le nœud de rejoindre le cluster.

```
kubectl describe nodeclaim <node_claim>
```

Messages d'erreur courants :

`Error getting launch template configs`

Cette erreur peut s'afficher si vous définissez des balises personnalisées dans `NodeClass` en utilisant les autorisations par défaut du rôle IAM du cluster. Consultez the section called "Identités et accès".

`Error creating fleet`

Il peut s'agir d'un problème d'autorisation lié à l'appel `RunInstances` effectué via l'API EC2. Vérifiez AWS CloudTrail les erreurs et the section called "Rôle IAM du cluster du mode automatique" les autorisations IAM requises.

Détection des problèmes de connectivité des nœuds avec l'outil **`VPC Reachability Analyzer`**

> **Note**
>
> Vous êtes facturé pour chaque analyse exécutée avec le VPC Reachability Analyzer. Pour les détails de tarification, consultez Tarification Amazon VPC.

L'une des raisons pour lesquelles une instance n'a pas rejoint le cluster est un problème de connectivité réseau qui l'empêche d'atteindre le serveur d'API. Pour diagnostiquer ce problème, vous pouvez utiliser le VPC Reachability Analyzer afin d'effectuer une analyse de la connectivité entre un nœud qui ne parvient pas à rejoindre le cluster et le serveur d'API. Vous aurez besoin de deux informations :

- ID d'instance du nœud qui ne peut pas rejoindre le cluster
- Adresse IP du point de terminaison du serveur d'API Kubernetes

Pour obtenir l'ID d'instance, vous devez créer une charge de travail sur le cluster afin que le mode automatique EKS lance une instance EC2. Cela crée également un objet `NodeClaim` dans votre cluster qui aura l'ID d'instance. Exécutez `kubectl get nodeclaim -o yaml` pour imprimer tous les éléments `NodeClaims` de votre cluster. Chaque `NodeClaim` contient l'ID d'instance en tant que champ et à nouveau dans le providerID :

```
kubectl get nodeclaim -o yaml
```

L'exemple qui suit illustre un résultat.

```
    nodeName: i-01234567890123456
    providerID: aws:///us-west-2a/i-01234567890123456
```

Vous pouvez déterminer votre point de terminaison de serveur d'API Kubernetes en exécutant `kubectl get endpoint kubernetes -o yaml`. Les adresses se trouvent dans le champ des adresses :

```
kubectl get endpoints kubernetes -o yaml
```

L'exemple qui suit illustre un résultat.

```
apiVersion: v1
kind: Endpoints
metadata:
  name: kubernetes
  namespace: default
subsets:
- addresses:
  - ip: 10.0.143.233
  - ip: 10.0.152.17
  ports:
  - name: https
    port: 443
    protocol: TCP
```

Avec ces deux informations, vous pouvez effectuer l'analyse de connectivité. Tout d'abord, accédez à VPC Reachability Analyzer dans la AWS Management Console.

1. Cliquez sur « Créer et analyser le chemin »

2. Donnez un nom à l'analyse (par exemple « Échec de rejoindre le nœud »)
3. Pour le « Type de source », sélectionnez « Instances »
4. Saisissez l'ID d'instance du nœud défaillant comme « Source »
5. Pour « Destination du chemin », sélectionnez « Adresse IP »
6. Saisissez l'une des adresses IP du serveur d'API comme « Adresse de destination »
7. Développez la section « Configuration d'en-têtes de paquets supplémentaires »
8. Saisissez un « Port de destination » de 443
9. Sélectionnez « Protocole » comme TCP, s'il n'est pas déjà sélectionné
10. Cliquez sur « Créer et analyser le chemin »
11. L'analyse peut prendre quelques minutes. Si les résultats de l'analyse indiquent une défaillance de connectivité, ils préciseront l'endroit exact où l'échec s'est produit dans le chemin réseau, afin que vous puissiez corriger le problème.

Partage des volumes entre les pods

Les nœuds en mode automatique EKS sont configurés SELinux en mode d'application, ce qui permet une meilleure isolation entre les pods exécutés sur le même nœud. Lorsque cette option SELinux est activée, la plupart des pods non privilégiés se verront automatiquement appliquer leur propre étiquette de sécurité multi-catégories (MCS). Cette étiquette MCS est unique pour chaque pod et vise à garantir qu'un processus dans un pod ne puisse pas interagir avec un processus dans un autre pod ou sur l'hôte. Même si un pod étiqueté s'exécute en tant que root et dispose d'un accès au système de fichiers de l'hôte, il ne pourra pas modifier des fichiers sur l'hôte, exécuter des appels système sensibles, accéder à l'exécution du conteneur, ni obtenir les clés secrètes du kubelet.

En conséquence, vous pouvez rencontrer des problèmes lors du partage de données entre pods. Par exemple, un `PersistentVolumeClaim` avec un mode d'accès `ReadWriteOnce` ne permettra toujours pas à plusieurs pods d'accéder simultanément au volume.

Pour autoriser le partage entre pods, vous pouvez utiliser les `seLinuxOptions` du pod pour configurer la même étiquette MCS sur ces pods. Dans cet exemple, nous attribuons les trois catégories `c123,c456,c789` au pod. Cela n'entre pas en conflit avec les catégories attribuées automatiquement aux pods sur le nœud, car ces derniers ne reçoivent que deux catégories.

```
securityContext:
```

```
  seLinuxOptions:
    level: "s0:c123,c456,c789"
```

Affichage des événements Karpenter dans les journaux du plan de contrôle

Pour les clusters EKS disposant des journaux du plan de contrôle activés, vous pouvez analyser les actions et les décisions de Karpenter en interrogeant les journaux. Cela est particulièrement utile pour la résolution des problèmes du mode automatique EKS liés à la mise en service des nœuds, la mise à l'échelle ou la suppression des nœuds. Pour consulter les événements liés à Karpenter, utilisez la requête CloudWatch Logs Insights suivante :

```
fields @timestamp, @message
| filter @logStream like /kube-apiserver-audit/
| filter @message like 'DisruptionBlocked'
or @message like 'DisruptionLaunching'
or @message like 'DisruptionTerminating'
or @message like 'DisruptionWaitingReadiness'
or @message like 'Unconsolidatable'
or @message like 'FailedScheduling'
or @message like 'NoCompatibleInstanceTypes'
or @message like 'NodeRepairBlocked'
or @message like 'Disrupted'
or @message like 'Evicted'
or @message like 'FailedDraining'
or @message like 'TerminationGracePeriodExpiring'
or @message like 'TerminationFailed'
or @message like 'FailedConsistencyCheck'
or @message like 'InsufficientCapacityError'
or @message like 'UnregisteredTaintMissing'
or @message like 'NodeClassNotReady'
| sort @timestamp desc
```

Cette requête filtre les événements spécifiques liés à Karpenter dans les journaux d'audit de kube-apiserver. Les événements incluent différents états d'interruption, des échecs de planification, des problèmes de capacité et des anomalies liées aux nœuds. En analysant ces journaux, vous pouvez mieux comprendre :

- Pourquoi Karpenter exécute certaines actions.
- Tout problème empêchant le provisionnement, la mise à l'échelle ou la terminaison correcte des nœuds.

- Problèmes potentiels de capacité ou de compatibilité avec les types d'instances.
- Événements liés au cycle de vie des nœuds, tels que les interruptions, les évictions ou les terminaisons.

Pour utiliser cette requête, procédez comme suit :

1. Accédez à la CloudWatch console
2. Sélectionnez « Logs Insights » dans le volet de navigation de gauche
3. Sélectionnez le groupe de journaux pour les journaux du plan de contrôle de votre cluster EKS
4. Collez la requête dans l'éditeur de requêtes
5. Ajustez la plage de temps selon les besoins
6. Exécuter la requête

Les résultats afficheront une chronologie des événements liés à Karpenter, ce qui vous aidera à résoudre les problèmes et à comprendre le comportement du mode automatique EKS dans votre cluster. Pour examiner les actions de Karpenter sur un nœud particulier, vous pouvez ajouter un filtre de ligne indiquant l'ID de l'instance à la requête mentionnée précédemment :

```
|filter @message like /[.replaceable]`i-1234567891012345 6`/
```

> **Note**
>
> Pour exécuter cette requête, la journalisation du plan de contrôle doit être activée sur votre cluster EKS. Si ce n'est pas encore le cas, consultez the section called "journaux de plan de contrôle ".

Résolution des problèmes liés aux contrôleurs inclus dans le mode automatique

Si vous rencontrez un problème avec un contrôleur, vérifiez les points suivants :

- Si les ressources associées à ce contrôleur sont correctement formatées et valides.
- Si les ressources AWS IAM et Kubernetes RBAC sont correctement configurées pour votre cluster. Pour de plus amples informations, veuillez consulter the section called "Identités et accès".

Ressources connexes

Utilisez ces articles de AWS Re:Post pour connaître les étapes de dépannage avancées :

- Comment résoudre les problèmes de mise à l'échelle courants dans le mode automatique EKS ?
- Comment résoudre les problèmes de provisionnement des groupes de nœuds et des classes de nœuds personnalisés dans le mode automatique Amazon EKS ?
- Comment résoudre les problèmes liés aux groupes de nœuds intégrés au mode automatique EKS dont l'état est inconnu ?

Consultation des notes de mise à jour du mode automatique EKS

Cette page présente les mises à jour du mode automatique Amazon EKS. Vous pouvez la consulter régulièrement pour suivre les annonces concernant les nouvelles fonctionnalités, les correctifs, les problèmes connus et les fonctionnalités obsolètes.

Pour recevoir des notifications concernant toutes les modifications apportées aux fichiers sources de cette page de documentation spécifique, vous pouvez vous abonner à l'URL suivante à l'aide d'un lecteur RSS :

```
https://github.com/awsdocs/amazon-eks-user-guide/commits/mainline/latest/ug/automode/auto-change.adoc.atom
```

2 février 2026

Fonctionnalité : Ajout du support pour désactiver le trafic V4egress provenant IPv6 des pods dans les clusters en mode automatique EKS. IPv6 Pour de plus amples informations, veuillez consulter the section called "Désactivez la IPv4 sortie des IPv6 pods en IPv6 clusters.".

19 décembre 2025

Fonctionnalité : Ajout de la prise en charge du mode IP secondaire qui fournit des adresses IP secondaires au lieu du préfixe des nœuds automatiques. Ce mode permet de conserver une adresse IP secondaire minimale IPTarget et d'économiser des ressources IP pour les clients qui n'ont pas besoin de préfixer d'autres adresses secondaires IPs ou de préfixes. Pour plus d'informations, consultez the section called "Spécification de la classe de nœuds" et the section called "Mode IP secondaire pour les pods".

19 novembre 2025

Fonctionnalité : Activation du pull et du déballage en parallèle Seekable OCI (SOCI) pour les instances des familles G, P et Trn avec stockage local. NVMe Le pull and unpack en parallèle SOCI est toujours utilisé pour ces familles d'instances avec le mode automatique EKS et aucune modification de configuration n'est requise pour l'activer. Pour plus d'informations sur SOCI, consultez le blog de lancement.

19 novembre 2025

Fonctionnalité : Ajout de la prise en charge des pools de nœuds à capacité statique qui maintiennent un nombre fixe de nœuds. Pour de plus amples informations, veuillez consulter the section called "Pools de nœuds à capacité statique".

23 octobre 2025

Fonctionnalité : Les utilisateurs disposant de clusters dans les régions des États-Unis peuvent désormais demander à utiliser la compatibilité FIPS en AMIs le spécifiant `spec.advancedSecurity.fips` dans leur NodeClass définition.

1er octobre 2025

Fonctionnalité : Le mode automatique d'EKS prend désormais en charge le déploiement de nœuds dans les Zones AWS Locales. Pour de plus amples informations, veuillez consulter the section called "Utiliser les Zones Locales".

30 septembre 2025

Fonctionnalité : Ajout de la prise en charge d'InstanceProfile, qui s'exclut mutuellement NodeClass `spec.instanceProfile` du champ. `spec.role`

29 septembre 2025

Le DRA n'est actuellement pas pris en charge par le mode automatique EKS.

10 septembre 2025

Tâche d'entretien : les événements déclenchés par le contrôleur de calcul en mode automatique utiliseront désormais le nom `eks-auto-mode/compute` au lieu de `karpenter`.

24 août 2025

Correction de bogue : VPCs l'utilisation d'un ensemble d'options DHCP avec un nom de domaine personnalisé contenant des majuscules empêchait les nœuds de rejoindre le cluster en raison de la génération d'un nom d'hôte non valide. Ce problème est désormais résolu : les noms de domaine contenant des majuscules fonctionnent correctement.

15 août 2025

Correction de bogue : le Pod Identity Agent n'écoute désormais que l'adresse IPv4 Link Local dans un cluster IPv4 EKS afin d'éviter que le Pod ne puisse pas atteindre l' IPv6 adresse.

6 août 2025

Fonctionnalité : Ajout d'une nouvelle configuration NodeClass `spec.advancedNetworking.associatePublicIPAddress` qui peut être utilisée pour empêcher l'attribution d'adresses IP publiques aux nœuds en mode automatique EKS

30 juin 2025

Fonctionnalité : Le mode automatique utilise NodeClass désormais la clé KMS personnalisée configurée pour chiffrer le volume racine en lecture seule de l'instance, en plus du read/write volume de données. Auparavant, la clé KMS personnalisée ne servait qu'au chiffrement du volume de données.

20 juin 2025

Fonctionnalité : Support pour contrôler le déploiement des charges de travail dans les réservations de capacité à la demande EC2 ()ODCRs. Cela ajoute la clé facultative `capacityReservationSelectorTerms` au NodeClass, vous permettant de contrôler explicitement les charges de travail utilisées par ODCRs vos charges de travail. Pour de plus amples informations, veuillez consulter the section called “Contrôle des réserves de capacité”.

13 juin 2025

Caractéristique : prise en charge de sous-réseaux de pods distincts dans la `NodeClass`. Cela ajoute les clés facultatives `` `podSubnetSelectorTerms `` et `podSecurityGroupSelectorTerms` permettant de définir les sous-réseaux et les groupes de sécurité pour les pods. Pour de plus amples informations, veuillez consulter the section called “Sélection des sous-réseaux pour les pods”.

30 avril 2025

Caractéristique : prise en charge des serveurs proxy réseau sortant dans la `NodeClass`. Cela ajoute la clé facultative `advancedNetworking` permettant de configurer votre proxy HTTPS. Pour de plus amples informations, veuillez consulter the section called "Spécification de la classe de nœuds".

18 avril 2025

Caractéristique : prise en charge de la résolution des domaines .local (habituellement réservés au Multicast DNS) via DNS unicast.

11 avril 2025

Caractéristique : ajout des paramètres `certificateBundles` et `ephemeralStorage.kmsKeyID` à `NodeClass`. Pour de plus amples informations, veuillez consulter the section called "Spécification de la classe de nœuds".

Caractéristique : amélioration de la vitesse d'extraction des images, en particulier pour les types d'instances disposant d'un stockage local permettant une décompression plus rapide des images.

Correction de bogue : résolution d'un problème de course à l'origine FailedCreatePodSandBox d'une erreur lors de la numérotation : dial tcp 127.0.0. 1:50051 : connexion : la connexion était parfois refusée pour les pods planifiés vers un nœud immédiatement au démarrage.

4 avril 2025

Caractéristique : augmentation de `registryPullQPS` de 5 à 25 et de `registryBurst` de 10 à 50 pour réduire la limitation de débit appliquée côté client lors de l'extraction des images (`Failed to pull image xyz: pull QPS exceeded`)

31 mars 2025

Correction de bogue : résolution d'un problème où, si un pod CoreDNS s'exécutait sur un nœud du mode automatique, les requêtes DNS des pods de ce nœud étaient envoyées à ce pod CoreDNS au lieu du serveur DNS local du nœud. Désormais, les requêtes DNS provenant de pods sur un nœud du mode automatique sont systématiquement dirigées vers le DNS local du nœud.

21 mars 2025

Correction de bogue : les nœuds du mode automatique résolvent désormais correctement le domaine `kube-dns.kube-system.svc.cluster.local` lorsqu'aucun service `kube-dns` n'est installé dans le cluster. Résout GitHub le problème #2546.

14 mars 2025

Caractéristique : activation de la sortie `IPv4` dans les clusters `IPv6`. Le trafic `IPv4` sortant depuis les clusters du mode automatique `IPv6` est désormais automatiquement traduit vers l'adresse `v4` de l'ENI primaire du nœud.

Cycle de vie et configuration des clusters Amazon EKS

Cette section fournit des conseils détaillés sur la gestion complète du cycle de vie des clusters Kubernetes et sur les différentes manières de les configurer. Elle couvre la création d'un cluster, la surveillance de son état, la mise à jour des versions Kubernetes et la suppression du cluster. Il inclut également des sujets avancés sur la façon de configurer l'accès aux terminaux, le support enable/disable Windows, la configuration de clusters privés, la mise en œuvre de l'autoscaling et la manière d'améliorer la résilience grâce au changement de zone pour la redirection du trafic dans les configurations multi-AZ.

Rubriques

- Création d'un cluster du mode automatique Amazon EKS
- Création d'un cluster Amazon EKS
- Plan de contrôle provisionné Amazon EKS
- Préparation aux mises à niveau des versions Kubernetes et résolution des erreurs de configuration grâce aux informations sur les clusters
- Mettre à jour un cluster existant vers une nouvelle version de Kubernetes
- Supprimer un cluster
- Point de terminaison du serveur d'API du cluster
- Déployer des nœuds Windows sur des clusters EKS
- Désactiver la prise en charge Windows
- Déployer des clusters privés avec un accès Internet limité
- Mise à l'échelle du calcul du cluster avec Karpenter et Cluster Autoscaler
- Découvrez le changement de zone du contrôleur de récupération d'application (ARC) Amazon dans Amazon EKS
- Activer le changement de zone EKS pour éviter les zones de disponibilité défaillantes

Création d'un cluster du mode automatique Amazon EKS

Cette rubrique fournit des instructions détaillées pour créer un cluster du mode automatique Amazon EKS à l'aide d'options de configuration avancées. Il couvre les prérequis, les options de mise en réseau et la configuration des modules complémentaires. Le processus comprend la création des rôles IAM, la configuration des paramètres du cluster, la spécification des paramètres réseau et la

sélection des modules complémentaires. Les utilisateurs peuvent créer des clusters à l'aide de la CLI AWS Management Console ou de la AWS CLI, avec step-by-step des instructions fournies pour les deux méthodes.

Pour les utilisateurs recherchant un processus d'installation moins complexe, consultez la section correspondante pour connaître les étapes simplifiées de création de cluster :

- the section called "CLI eksctl"
- the section called " AWS CLI"
- the section called "Console de gestion"

Ce guide de configuration avancée s'adresse aux utilisateurs souhaitant un contrôle plus précis sur la configuration de leur cluster du mode automatique EKS et ayant une bonne connaissance des concepts et exigences Amazon EKS. Avant de poursuivre la configuration avancée, assurez-vous d'avoir satisfait à tous les prérequis et de bien comprendre les exigences en matière de mise en réseau et d'IAM pour les clusters du mode automatique EKS.

Le mode automatique EKS nécessite des autorisations IAM supplémentaires. Pour en savoir plus, consultez :

- the section called "Rôles IAM des clusters du mode automatique EKS"
- the section called "Identités et accès"

> **Note**
>
> Si vous souhaitez créer un cluster sans le mode automatique EKS, consultez the section called "Créer un cluster".
> Cette rubrique traite de la configuration avancée. Si vous souhaitez simplement commencer avec le mode automatique EKS, consultez the section called "Créer un cluster".

Conditions préalables

- Un VPC et des sous-réseaux existants qui répondent aux exigences Amazon EKS. Avant de déployer un cluster pour une utilisation en production, nous vous recommandons de bien connaître les exigences du VPC et du sous-réseau. Si vous n'avez pas de VPC ni de sous-réseaux, vous pouvez les créer à l'aide d'un modèle fourni par Amazon EKS. AWS CloudFormation

- L'outil de ligne de commande `kubectl` est installé sur votre appareil ou AWS CloudShell. La version peut correspondre à celle utilisée par votre cluster Kubernetes, ou différer d'au plus une version mineure, qu'elle soit antérieure ou plus récente. Par exemple, si la version de votre cluster est `1.29`, vous pouvez utiliser la version `kubectl` `1.28`, `1.29` ou `1.30`. Pour installer ou mettre à niveau `kubectl`, veuillez consulter the section called "Configurer kubectl et eksctl".
- Version `2.12.3` ou version ultérieure `1.27.160` ou version ultérieure de l'interface de ligne de AWS commande (AWS CLI) installée et configurée sur votre appareil ou AWS CloudShell. Pour vérifier votre version actuelle, utilisez `aws --version`. Pour installer la dernière version, consultez la section Installation et configuration rapide avec aws configure dans le Guide de l'utilisateur de l'interface de ligne de AWS commande.
- Un principal IAM disposant des autorisations nécessaires pour créer et modifier les ressources EKS et IAM est requis.

Créer un cluster - AWS console

1. Ouvrez la console Amazon EKS.
2. Choisissez Add cluster (Ajouter un cluster), puis choisissez Create (Créer).
3. Sous Options de configuration, sélectionnez Configuration personnalisée.
   - Cette rubrique traite de la configuration personnalisée. Pour plus d'informations sur la configuration rapide, consultez the section called "Console de gestion".
4. Vérifiez que l'option Utiliser le mode automatique EKS est activée.
   - Cette rubrique décrit la création de clusters avec le mode automatique EKS. Pour plus d'informations sur la création de clusters sans le mode automatique EKS, consultez the section called "Créer un cluster".
5. Sur la page Configurer le cluster, renseignez les champs suivants :
   - Nom : nom de votre cluster. Le nom ne peut contenir que des caractères alphanumériques (sensibles à la casse), des tirets et des traits de soulignement. Il doit commencer par un caractère alphanumérique et ne peut pas dépasser 100 caractères. Le nom doit être unique dans la AWS région et le AWS compte dans lesquels vous créez le cluster.
   - Rôle IAM du cluster : choisissez le rôle IAM du cluster Amazon EKS que vous avez créé pour permettre au plan de contrôle Kubernetes de gérer AWS les ressources en votre nom. Si vous n'avez pas encore créé de rôle IAM de cluster pour le mode automatique EKS, sélectionnez le bouton Créer le rôle recommandé pour créer le rôle avec les autorisations requises dans la console IAM.

- Version Kubernetes : version de Kubernetes à utiliser pour votre cluster. Nous vous recommandons de sélectionner la dernière version, sauf si vous avez besoin d'une version antérieure.
- Politique de mise à niveau : politique de version Kubernetes que vous souhaitez définir pour votre cluster. Si vous souhaitez que votre cluster fonctionne uniquement sur une version en prise en charge standard, choisissez Standard. Si vous souhaitez que votre cluster passe en prise en charge étendue à la fin de la période de prise en charge standard, choisissez Étendue. Si vous sélectionnez une version de Kubernetes déjà en prise en charge étendue, vous ne pouvez pas choisir la prise en charge standard comme option.

6. Dans la section Calcul en mode automatique de la page de configuration du cluster, renseignez les champs suivants :
   - Groupes de nœuds : déterminez si vous souhaitez utiliser les groupes de nœuds intégrés. Pour de plus amples informations, veuillez consulter the section called "Vérifier les groupes de nœuds créés".
   - Rôle IAM de nœud : si vous activez un ou plusieurs groupes de nœuds intégrés, vous devez sélectionner un rôle IAM de nœud. Le mode automatique EKS attribuera ce rôle aux nouveaux nœuds. Vous ne pourrez plus modifier cette valeur après la création du cluster. Si vous n'avez pas encore créé de rôle IAM de nœud pour le mode automatique EKS, sélectionnez le bouton Créer le rôle recommandé pour créer le rôle avec les autorisations nécessaires. Pour plus d'informations sur ce rôle, consultez the section called "Identités et accès".
7. Dans la section Accès au cluster de la page de configuration du cluster, renseignez les champs suivants :
   - Amorcer l'accès administrateur au cluster : le créateur du cluster devient automatiquement administrateur Kubernetes. Si vous souhaitez désactiver cet accès automatique, sélectionnez Refuser l'accès administrateur au cluster.
   - Mode d'authentification du cluster : le mode automatique EKS nécessite des entrées d'accès EKS, correspondant au mode d'authentification de l'API EKS. Vous pouvez éventuellement activer le mode `ConfigMap` d'authentification en sélectionnant EKS API et ConfigMap.
8. Saisissez les autres champs figurant sur la page de configuration du cluster :
   - Chiffrement des secrets : (facultatif) choisissez d'activer le chiffrement des secrets Kubernetes à l'aide d'une clé KMS. Vous pouvez également activer cette option une fois que vous avez créé votre cluster. Avant d'activer cette fonctionnalité, assurez-vous d'avoir pris connaissance des informations mentionnées dans the section called "Activer le chiffrement secret".

- Changement de zone ARC : le mode automatique EKS ne prend pas en charge le changement de zone ARC.
- Identifications◦: (facultatif) ajoutez des identifications à votre cluster. Pour de plus amples informations, veuillez consulter the section called “Balisage de vos ressources ”.

  Lorsque vous avez terminé d'utiliser cette page, choisissez Suivant.

9. Sur la page Spécifier les réseaux sélectionnez des valeurs pour les champs suivants :

- VPC : choisissez un VPC existant qui répond aux exigences relatives au VPC Amazon EKS pour y créer votre cluster. Avant de sélectionner un VPC, nous vous recommandons de bien connaître toutes les exigences et considérations de the section called “Exigences requises pour le VPC et les sous-réseaux”. Vous ne pouvez pas modifier le VPC que vous souhaitez utiliser après la création du cluster. Si aucun VPCs n'est répertorié, vous devez d'abord en créer un. Pour de plus amples informations, veuillez consulter the section called “Création d'un VPC”.
- Sous-réseaux : par défaut, tous les sous-réseaux disponibles dans le VPC spécifié dans le champ précédent sont présélectionnés. Vous devez en sélectionner au moins deux.

  Les sous-réseaux que vous choisissez doivent respecter les exigences relatives aux sous-réseaux d'Amazon EKS. Avant de sélectionner les sous-réseaux, assurez-vous de bien comprendre toutes les exigences et considérations relatives aux VPC et sous-réseaux Amazon EKS.

  Groupes de sécurité : (facultatif) spécifiez un ou plusieurs groupes de sécurité créant des interfaces réseau auxquelles vous souhaitez associer Amazon EKS.

  Que vous choisissiez ou non un groupe de sécurité, Amazon EKS crée un groupe de sécurité qui permet la communication entre votre cluster et votre VPC. Amazon EKS associe ce groupe de sécurité, et tout ce que vous choisissez, aux interfaces réseau qu'il crée. Pour plus d'informations sur le groupe de sécurité de cluster créé par Amazon EKS, consultez the section called “Exigences de groupe de sécurité”. Vous pouvez modifier les règles dans le groupe de sécurité de cluster créé par Amazon EKS.
- Choisissez la famille d'adresses IP du cluster : vous pouvez choisir l'une ou l'autre IPv4et IPv6.

  Kubernetes attribue par défaut des adresses `IPv4` aux pods et aux services. Avant de décider d'utiliser la famille `IPv6`, assurez-vous de bien connaître toutes les considérations et exigences mentionnées dans les rubriques Exigences et considérations relatives au VPC, the section called “Exigences et considérations requises pour les sous-réseaux”, the section called “Exigences de groupe de sécurité” et the section called “IPv6”. Si vous choisissez la

famille `IPv6`, vous ne pouvez pas spécifier une plage d'adresses à partir de laquelle Kubernetes doit attribuer les adresses de service `IPv6`, comme c'est possible avec la famille `IPv4`. Kubernetes attribue les adresses de service à partir de la plage d'adresses locales uniques (`fc00::/7`).

- (Facultatif) Choisissez Configurer la plage d'adresses IP du service Kubernetes et spécifiez une gamme de services **`IPv4`**.

  La spécification de votre propre plage peut aider à prévenir les conflits entre les services Kubernetes et d'autres réseaux appairés ou connectés à votre VPC. Entrez une plage en notation CIDR. Par exemple : `10.2.0.0/16`.

  Le bloc d'adresse CIDR doit répondre aux critères suivants :

  - Être situé dans l'une des plages suivantes : `10.0.0.0/8`, `172.16.0.0/12`, ou `192.168.0.0/16`.
  - Avoir une taille minimale de `/24` et une taille maximale de `/12`.
  - Ne pas chevaucher la plage du VPC pour vos ressources Amazon EKS.

  Vous ne pouvez spécifier cette option que lorsque vous utilisez la famille d'adresses `IPv4` et uniquement lors de la création du cluster. Si vous ne spécifiez pas cela, alors Kubernetes attribue des adresses IP de service à partir des blocs CIDR `10.100.0.0/16` ou `172.20.0.0/16`.

- Pour Accès au point de terminaison de cluster, sélectionnez une option. Une fois votre cluster créé, vous pouvez modifier cette option. Avant de sélectionner une option autre que par défaut, assurez-vous de vous familiariser avec les options et leurs implications. Pour de plus amples informations, veuillez consulter the section called "Accès au point de terminaison de cluster".

  Lorsque vous avez terminé d'utiliser cette page, choisissez Suivant.

10. (Facultatif) Sur la page Configurer l'observabilité, choisissez quelles options de Métriques et de Journalisation du plan de contrôle doivent être activées. Par défaut, chaque type de journal est désactivé.
    - Pour plus d'informations sur l'option de métriques Prometheus, consultez the section called "Étape 1 : activer les métriques Prometheus".
    - Pour plus d'informations sur les options de Journalisation du plan de contrôle, consultez the section called "journaux de plan de contrôle ".
    - Lorsque vous avez terminé d'utiliser cette page, choisissez Suivant.
11. Sur la page Select add-ons (Sélectionner les modules complémentaires), choisissez les modules complémentaires que vous souhaitez ajouter à votre cluster. Vous pouvez choisir autant de

modules complémentaires Amazon EKS et de modules complémentaires AWS Marketplace que vous le souhaitez. Si les modules complémentaires AWS Marketplace que vous souhaitez installer ne figurent pas dans la liste, vous pouvez cliquer sur la numérotation des pages pour afficher des résultats supplémentaires ou rechercher les modules complémentaires AWS Marketplace disponibles en saisissant du texte dans le champ de recherche. Vous pouvez également filtrer les résultats par catégorie, fournisseur ou modèle de tarification, puis sélectionner les modules complémentaires affichés dans les résultats de recherche. Lorsque vous créez un cluster, vous pouvez afficher, sélectionner et installer tout module complémentaire compatible avec les identités de pod EKS, comme indiqué dans the section called "Pod Identity".

- Le mode automatique EKS automatise les fonctionnalités de certains modules complémentaires. Si vous prévoyez de déployer des groupes de nœuds gérés par EKS sur votre cluster du mode automatique EKS, sélectionnez Modules complémentaires Amazon EKS supplémentaires et consultez les options. Vous devrez peut-être installer des modules complémentaires tels que CoreDNS et kube-proxy. EKS installera les modules complémentaires de cette section uniquement sur des nœuds et des groupes de nœuds autogérés.
- Lorsque vous avez terminé d'utiliser cette page, choisissez Suivant.

12. Sur la page Configurer les paramètres de modules complémentaires sélectionnés, sélectionnez la version que vous voulez installer. Vous pourrez toujours effectuer une mise à jour vers une version ultérieure après la création du cluster.

    Pour les modules complémentaires compatibles avec EKS Pod Identities, vous pouvez utiliser la console pour générer automatiquement le rôle avec le nom, la politique AWS gérée et la politique de confiance préremplis spécifiquement pour le module complémentaire. Vous pouvez réutiliser des rôles existants ou créer de nouveaux rôles pour les modules complémentaires pris en charge. Pour les étapes permettant d'utiliser la console afin de créer des rôles pour les modules complémentaires qui prennent en charge les identités de pod EKS, consultez the section called "Créer un module complémentaire (AWS console)". Si un module complémentaire ne prend pas en charge l'identité du pod EKS, un message s'affiche avec des instructions pour utiliser l'assistant afin de créer les rôles IAM pour comptes de service (IRSA) après la création du cluster.

    Vous pourrez mettre à jour la configuration de chaque module complémentaire après la création du cluster. Pour plus d'informations sur la configuration des modules complémentaires, consultez the section called "Mettre à jour un module complémentaire". Lorsque vous avez terminé d'utiliser cette page, choisissez Suivant.

13. Sur la page Vérifier et créer, passez en revue les informations que vous avez saisies ou sélectionnées sur les pages précédentes. Si vous devez apporter des modifications, choisissez

Modifier. Quand vous êtes satisfait, choisissez Créer. Le champ État affiche EN COURS DE CRÉATION pendant que le cluster est provisionné.

> **Note**
>
> Il est possible que vous receviez un message d'erreur indiquant que l'une des zones de disponibilité de votre demande ne dispose pas d'une capacité suffisante pour créer un cluster Amazon EKS. Si cela se produit, la sortie de l'erreur contient les zones de disponibilité qui peuvent prendre en charge un nouveau cluster. Essayez à nouveau de créer votre cluster avec au moins deux sous-réseaux situés dans les zones de disponibilité prises en charge pour votre compte. Pour de plus amples informations, veuillez consulter the section called "Capacité insuffisante".

L'approvisionnement de cluster dure plusieurs minutes.

Créer un cluster - AWS CLI

Les instructions CLI suivantes concernent la création de ressources IAM et la création du cluster.

Création d'un rôle IAM de cluster du mode automatique Amazon EKS

Étape 1 : créer la politique d'approbation

Créez une politique d'approbation qui permet au service Amazon EKS d'assumer le rôle. Enregistrez la politique sous `trust-policy.json` :

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

```
  ]
}
```

Étape 2 : créer le rôle IAM

Utilisez la politique d'approbation pour créer le rôle IAM du cluster :

```
aws iam create-role \
    --role-name AmazonEKSAutoClusterRole \
    --assume-role-policy-document file://trust-policy.json
```

Étape 3 : noter l'ARN du rôle

Extrayez et enregistrez l'ARN du nouveau rôle pour les étapes ultérieures :

```
aws iam get-role --role-name AmazonEKSAutoClusterRole --query "Role.Arn" --output text
```

Étape 4 : attacher les politiques requises

Associez les politiques AWS gérées suivantes au rôle IAM du cluster pour accorder les autorisations nécessaires :

EKSClusterPolitique d'Amazon :

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSClusterPolicy
```

EKSComputePolitique d'Amazon :

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSComputePolicy
```

Amazon EKSBlock StoragePolicy :

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSBlockStoragePolicy
```

Amazon EKSLoad BalancingPolicy :

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSLoadBalancingPolicy
```

EKSNetworkingPolitique d'Amazon :

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSNetworkingPolicy
```

Création d'un rôle IAM de nœud du mode automatique EKS

Étape 1 : créer la politique d'approbation

Créez une politique d'approbation qui permet au service Amazon EKS d'assumer le rôle. Enregistrez la politique sous `node-trust-policy.json` :

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Étape 2 : créer le rôle IAM du nœud

Utilisez le fichier node-trust-policy.json de l'étape précédente pour définir les entités qui peuvent assumer le rôle. Exécutez la commande suivante pour créer le rôle IAM du nœud :

```
aws iam create-role \
    --role-name AmazonEKSAutoNodeRole \
    --assume-role-policy-document file://node-trust-policy.json
```

Étape 3 : noter l'ARN du rôle

Après la création du rôle, extrayez et enregistrez l'ARN du rôle IAM du nœud. Cet ARN sera nécessaire dans les étapes suivantes. Utilisez la commande suivante pour extraire l'ARN :

```
aws iam get-role --role-name AmazonEKSAutoNodeRole --query "Role.Arn" --output text
```

Étape 4 : attacher les politiques requises

Associez les politiques AWS gérées suivantes au rôle Node IAM pour fournir les autorisations nécessaires :

Amazon EKSWorker NodeMinimalPolicy :

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoNodeRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSWorkerNodeMinimalPolicy
```

Amazon EC2 ContainerRegistryPullOnly :

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoNodeRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEC2ContainerRegistryPullOnly
```

Créer un cluster

1. Créez votre cluster à l'aide de la commande suivante. Avant d'exécuter la commande, effectuez les remplacements suivants :
   - *region-code*Remplacez-le par la AWS région dans laquelle vous souhaitez créer votre cluster.
   - Remplacez *my-cluster* par un nom pour votre cluster. Le nom ne peut contenir que des caractères alphanumériques (sensibles à la casse), des tirets et des traits de soulignement. Il doit commencer par un caractère alphanumérique et ne peut pas dépasser 100 caractères. Le nom doit être unique dans la AWS région et le AWS compte dans lesquels vous créez le cluster.
   - Remplacez la version *1.30* par n'importe quelle version prise en charge par Amazon EKS.
   - Remplacez *111122223333* par votre identifiant de compte
   - Si vous avez créé des rôles IAM nommés différemment pour les rôles de cluster et de nœud, remplacez le ARNs.

- Remplacez les valeurs de `subnetIds` par vos propres valeurs. Vous pouvez également en ajouter d'autres IDs. Vous devez spécifier au moins deux sous-réseaux IDs.

  Les sous-réseaux que vous choisissez doivent respecter les exigences relatives aux sous-réseaux d'Amazon EKS. Avant de sélectionner les sous-réseaux, assurez-vous de bien comprendre toutes les exigences et considérations relatives aux VPC et sous-réseaux Amazon EKS.

- Si vous ne voulez pas spécifier un ID de groupe de sécurité, supprimez `,securityGroupIds=sg-<ExampleID1>` depuis la commande. Si vous souhaitez spécifier un ou plusieurs groupes de sécurité IDs, remplacez les valeurs de par `securityGroupIds` les vôtres. Vous pouvez également en ajouter d'autres IDs.

  Que vous choisissiez ou non un groupe de sécurité, Amazon EKS crée un groupe de sécurité qui permet la communication entre votre cluster et votre VPC. Amazon EKS associe ce groupe de sécurité, et tout ce que vous choisissez, aux interfaces réseau qu'il crée. Pour plus d'informations sur le groupe de sécurité de cluster créé par Amazon EKS, consultez the section called “Exigences de groupe de sécurité”. Vous pouvez modifier les règles dans le groupe de sécurité de cluster créé par Amazon EKS.

```
aws eks create-cluster \
  --region region-code \
  --name my-cluster \
  --kubernetes-version 1.30 \
  --role-arn arn:aws: iam::111122223333:role/AmazonEKSAutoClusterRole \
  --resources-vpc-config '{"subnetIds": ["subnet-ExampleID1","subnet-ExampleID2"],
 "securityGroupIds": ["sg-ExampleID1"], "endpointPublicAccess": true,
 "endpointPrivateAccess": true}' \
  --compute-config '{"enabled": true, "nodeRoleArn": "arn:aws:
 iam::111122223333:role/AmazonEKSAutoNodeRole", "nodePools": ["general-purpose",
 "system"]}' \
  --kubernetes-network-config '{"elasticLoadBalancing": {"enabled": true}}' \
  --storage-config '{"blockStorage": {"enabled": true}}' \
  --access-config '{"authenticationMode": "API"}'
```

> **Note**
>
> Il est possible que vous receviez un message d'erreur indiquant que l'une des zones de disponibilité de votre demande ne dispose pas d'une capacité suffisante pour créer un cluster Amazon EKS. Si cela se produit, la sortie de l'erreur contient les zones de

disponibilité qui peuvent prendre en charge un nouveau cluster. Essayez à nouveau de créer votre cluster avec au moins deux sous-réseaux situés dans les zones de disponibilité prises en charge pour votre compte. Pour de plus amples informations, veuillez consulter the section called "Capacité insuffisante".

Les paramètres suivants sont des paramètres facultatifs qui, si nécessaire, doivent être ajoutés à la commande précédente. Vous pouvez activer ces options uniquement lorsque vous créez le cluster, et non après.

- Si vous souhaitez spécifier à partir de quel bloc de routage inter-domaines sans classe (CIDR) `IPv4` Kubernetes attribue des adresses IP de service, vous devez le spécifier en ajoutant le `--kubernetes-network-config serviceIpv4Cidr=<cidr-block>` à la commande suivante.

  La spécification de votre propre plage peut aider à prévenir les conflits entre les services Kubernetes et d'autres réseaux appairés ou connectés à votre VPC. Entrez une plage en notation CIDR. Par exemple : `10.2.0.0/16`.

  Le bloc d'adresse CIDR doit répondre aux critères suivants :

  - Être situé dans l'une des plages suivantes : `10.0.0.0/8`, `172.16.0.0/12`, ou `192.168.0.0/16`.
  - Avoir une taille minimale de `/24` et une taille maximale de `/12`.
  - Ne pas chevaucher la plage du VPC pour vos ressources Amazon EKS.

  Vous ne pouvez spécifier cette option que lorsque vous utilisez la famille d'adresses `IPv4` et uniquement lors de la création du cluster. Si vous ne spécifiez pas cela, alors Kubernetes attribue des adresses IP de service à partir des blocs CIDR `10.100.0.0/16` ou `172.20.0.0/16`.

- Si vous créez un cluster et que vous souhaitez qu'il attribue des adresses `IPv6` aux pods et aux services au lieu d'adresses `IPv4`, ajoutez `--kubernetes-network-config ipFamily=ipv6` à la commande suivante.

  Kubernetes attribue par défaut des adresses `IPv4` aux pods et aux services. Avant de décider d'utiliser la famille `IPv6`, assurez-vous de bien connaître toutes les considérations et exigences mentionnées dans les rubriques Exigences et considérations relatives au VPC, the section called "Exigences et considérations requises pour les sous-réseaux", the section called "Exigences de groupe de sécurité" et the section called "IPv6". Si vous choisissez la

famille `IPv6`, vous ne pouvez pas spécifier une plage d'adresses à partir de laquelle Kubernetes doit attribuer les adresses de service `IPv6`, comme c'est possible avec la famille `IPv4`. Kubernetes attribue les adresses de service à partir de la plage d'adresses locales uniques (`fc00::/7`).

2. Le provisionnement du cluster prend quelques minutes. Vous pouvez vérifier le statut de votre cluster avec la commande suivante.

```
aws eks describe-cluster --region region-code --name my-cluster --query
 "cluster.status"
```

Étapes suivantes

- the section called "Accès au cluster avec kubectl"
- the section called "Entrées d'accès"
- Activer le chiffrement des secrets pour votre cluster.
- Configurer la journalisation de votre cluster.
- Ajouter des nœuds à votre cluster.

Création d'un cluster Amazon EKS

> **Note**
>
> Cette rubrique traite de la création de clusters Amazon EKS sans le mode automatique EKS. Pour obtenir des instructions détaillées sur la création d'un cluster du mode automatique EKS, consultez the section called "Création d'un cluster Auto".
> Pour démarrer avec le mode automatique EKS, consultez the section called "Création d'un cluster (mode automatique EKS)".

Cette rubrique fournit une vue d'ensemble des options disponibles et décrit les éléments à prendre en compte lorsque vous créez un cluster Amazon EKS. Si vous devez créer un cluster en utilisant votre infrastructure sur site comme capacité de calcul pour les nœuds, consultez the section called "Créer un cluster". Si vous créez un cluster Amazon EKS pour la première fois, nous vous recommandons de suivre l'un des guides dans Démarrer. Ces guides vous aident à créer un cluster par défaut simple sans développer toutes les options disponibles.

Conditions préalables

- Un VPC et des sous-réseaux existants qui répondent aux exigences Amazon EKS. Avant de déployer un cluster pour une utilisation en production, nous vous recommandons de bien connaître les exigences du VPC et du sous-réseau. Si vous n'avez pas de VPC ni de sous-réseaux, vous pouvez les créer à l'aide d'un modèle fourni par Amazon EKS. AWS CloudFormation
- L'outil de ligne de commande `kubectl` est installé sur votre appareil ou AWS CloudShell. La version peut correspondre à celle utilisée par votre cluster Kubernetes, ou différer d'au plus une version mineure, qu'elle soit antérieure ou plus récente. Pour installer ou mettre à niveau `kubectl`, veuillez consulter the section called "Configurer kubectl et eksctl".
- Version `2.12.3` ou version ultérieure `1.27.160` ou version ultérieure de l'interface de ligne de AWS commande (AWS CLI) installée et configurée sur votre appareil ou AWS CloudShell. Pour vérifier votre version actuelle, utilisez `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Les gestionnaires de packages tels que `yum` Homebrew pour macOS ont souvent plusieurs versions de retard sur la dernière version de la AWS CLI. `apt-get` Pour installer la dernière version, consultez la section Installation et configuration rapide avec aws configure dans le Guide de l'utilisateur de l'interface de ligne de AWS commande. La version de la AWS CLI installée AWS CloudShell peut également avoir plusieurs versions de retard par rapport à la dernière version. Pour le mettre à jour, consultez la section Installation de la AWS CLI dans votre répertoire de base dans le guide de AWS CloudShell l'utilisateur.
- Un principal IAM disposant des autorisations `create` et `describe` sur un cluster Amazon EKS. Pour plus d'informations, consultez the section called "Créer un cluster Kubernetes local sur un Outpost" et the section called "Affichage de la liste ou description de tous les clusters".

Étape 1 : créer un rôle IAM de cluster

1. Si vous avez déjà un rôle IAM de cluster, ou si vous allez créer votre cluster avec `eksctl`, vous pouvez ignorer cette étape. Par défaut, `eksctl` crée un rôle pour vous.
2. Exécutez la commande suivante pour créer un fichier JSON de politique d'approbation IAM.

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

```
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

3. Créez le rôle IAM de cluster Amazon EKS. Si nécessaire, faites précéder *eks-cluster-role-trust-policy.json* par le chemin d'accès sur votre ordinateur sur lequel vous avez enregistré le fichier lors de l'étape précédente. La commande associe la politique d'approbation que vous avez créée lors de l'étape précédente à ce rôle. Pour créer un rôle IAM, le principal IAM qui crée le rôle doit se voir attribuer l'action (autorisation) IAM `iam:CreateRole`.

```
aws iam create-role --role-name myAmazonEKSClusterRole --assume-role-policy-document
 file://"eks-cluster-role-trust-policy.json"
```

4. Vous pouvez attribuer la politique gérée par Amazon EKS ou créer votre propre politique personnalisée. Pour connaître les autorisations minimales que vous devez utiliser dans votre politique personnalisée, consultez the section called "Rôle IAM du cluster".

   Associez la politique gérée par Amazon EKS nommée Amazon EKSCluster Policy au rôle. Pour attacher une politique IAM à un principal IAM, le principal qui attache la politique doit se voir attribuer l'une des actions (autorisations) IAM `iam:AttachUserPolicy` ou `iam:AttachRolePolicy`.

```
aws iam attach-role-policy --policy-arn arn:aws: iam::aws:policy/
AmazonEKSClusterPolicy --role-name myAmazonEKSClusterRole
```

Rôle lié à un service

Amazon EKS crée automatiquement un rôle lié à un service appelé `AWSServiceRoleForAmazonEKS`.

Ce rôle s'ajoute au rôle IAM du cluster. Un rôle lié à un service est un type unique de rôle IAM lié directement à Amazon EKS. Ce rôle permet à Amazon EKS de gérer les clusters de votre compte. Pour de plus amples informations, veuillez consulter the section called "Rôle de cluster".

L'identité IAM que vous utilisez pour créer le cluster EKS doit disposer de l'autorisation nécessaire pour créer ce rôle lié à un service. Cela inclut l'autorisation `iam:CreateServiceLinkedRole`.

Si le rôle lié à un service n'existe pas déjà et que votre rôle IAM actuel ne dispose pas des autorisations nécessaires pour le créer, l'opération de création du cluster échouera.

Étape 2 : créer un cluster

Vous pouvez créer un cluster à l'aide de :

- `eksctl`
- le AWS Management Console
- la AWS CLI

Créer un cluster – eksctl

1. Vous avez besoin d'`0.215.0`une version ou d'une version ultérieure de l'outil de ligne de `eksctl` commande installée sur votre appareil ou AWS CloudShell. Pour installer ou mettre à jour `eksctl`, veuillez consulter Installation dans la documentation de `eksctl`.
2. Créez un `IPv4` cluster Amazon EKS avec la version par défaut d'Amazon EKS Kubernetes dans votre région par défaut. AWS Avant d'exécuter la commande, effectuez les remplacements suivants :
3. *`region-code`*Remplacez-le par la AWS région dans laquelle vous souhaitez créer votre cluster.
4. Remplacez *`my-cluster`* par un nom pour votre cluster. Un nom ne peut contenir que des caractères alphanumériques (sensibles à la casse) et des traits d'union. Il doit commencer par un caractère alphanumérique et ne peut pas dépasser 100 caractères. Le nom doit être unique dans la AWS région et le AWS compte dans lesquels vous créez le cluster.
5. Remplacez la version *`1.33`* par n'importe quelle version prise en charge par Amazon EKS.
6. Modifiez les valeurs de `vpc-private-subnets` pour répondre à vos besoins. Vous pouvez également en ajouter d'autres IDs. Vous devez spécifier au moins deux sous-réseaux IDs. Si vous préférez spécifier des sous-réseaux publics, vous pouvez remplacer `--vpc-private-subnets` par `--vpc-public-subnets`. Les sous-réseaux publics ont une table de routage associée avec un routage vers une passerelle Internet, mais les sous-réseaux privés n'ont pas de table de routage associée. Nous recommandons d'utiliser des sous-réseaux privés dans la mesure du possible.

   Les sous-réseaux que vous choisissez doivent respecter les exigences relatives aux sous-réseaux d'Amazon EKS. Avant de sélectionner les sous-réseaux, assurez-vous de bien comprendre toutes les exigences et considérations relatives aux VPC et sous-réseaux Amazon EKS.

7. Exécutez la commande suivante :

```
eksctl create cluster --name my-cluster --region region-code --version 1.33 --vpc-
private-subnets subnet-ExampleID1,subnet-ExampleID2 --without-nodegroup
```

L'approvisionnement de cluster dure plusieurs minutes. Pendant la création du cluster, plusieurs lignes de sortie apparaissent. La dernière ligne de sortie est similaire à celle de l'exemple suivant.

```
[#]  EKS cluster "my-cluster" in "region-code" region is ready
```

8. Continuez avec the section called "Étape 3 : mettre à jour kubeconfig"

Paramètres facultatifs

Pour afficher la plupart des options qui peuvent être spécifiées lors de la création d'un cluster avec `eksctl`, utilisez la commande `eksctl create cluster --help`. Pour consulter toutes les options disponibles, vous pouvez utiliser un fichier `config`. Pour plus d'informations, consultez Utilisation des fichiers de configuration et du schéma du fichier de configuration dans la documentation `eksctl`. Vous pouvez trouver des exemples de fichiers de configuration sur GitHub.

Les paramètres suivants sont des paramètres facultatifs qui, si nécessaire, doivent être ajoutés à la commande précédente. Vous pouvez activer ces options uniquement lorsque vous créez le cluster, et non après. Si vous devez spécifier ces options, vous devez créer le cluster à l'aide d'un fichier de configuration eksctl et définir les paramètres dans ce fichier plutôt qu'en utilisant la commande précédente.

- Si vous souhaitez spécifier un ou plusieurs groupes de sécurité qu'Amazon EKS doit attribuer aux interfaces réseau qu'il crée, utilisez l'option securityGroup.

  Que vous choisissiez ou non un groupe de sécurité, Amazon EKS crée un groupe de sécurité qui permet la communication entre votre cluster et votre VPC. Amazon EKS associe ce groupe de sécurité, et tout ce que vous choisissez, aux interfaces réseau qu'il crée. Pour plus d'informations sur le groupe de sécurité de cluster créé par Amazon EKS, consultez the section called "Exigences de groupe de sécurité". Vous pouvez modifier les règles dans le groupe de sécurité de cluster créé par Amazon EKS.
- Si vous souhaitez spécifier le bloc de routage interdomaine `IPv4` sans classe (CIDR) à partir duquel Kubernetes attribue les adresses IP de service, spécifiez l'option CIDR du service. IPv4

La spécification de votre propre plage peut aider à prévenir les conflits entre les services Kubernetes et d'autres réseaux appairés ou connectés à votre VPC. Entrez une plage en notation CIDR. Par exemple : `10.2.0.0/16`.

Le bloc d'adresse CIDR doit répondre aux critères suivants :

- Être situé dans l'une des plages suivantes : `10.0.0.0/8`, `172.16.0.0/12`, ou `192.168.0.0/16`.
- Avoir une taille minimale de `/24` et une taille maximale de `/12`.
- Ne pas chevaucher la plage du VPC pour vos ressources Amazon EKS.

  Vous ne pouvez spécifier cette option que lorsque vous utilisez la famille d'adresses `IPv4` et uniquement lors de la création du cluster. Si vous ne spécifiez pas cela, alors Kubernetes attribue des adresses IP de service à partir des blocs CIDR `10.100.0.0/16` ou `172.20.0.0/16`.

- Si vous créez un cluster et souhaitez que le cluster attribue des adresses `IPv6` aux pods et aux services au lieu d'adresses `IPv4`, utilisez l'option ipFamily.

  Kubernetes attribue par défaut des adresses `IPv4` aux pods et aux services. Avant de décider d'utiliser la famille `IPv6`, assurez-vous de bien connaître toutes les considérations et exigences mentionnées dans les rubriques the section called "Exigences et considérations requises pour le VPC", the section called "Exigences et considérations requises pour les sous-réseaux", the section called "Exigences de groupe de sécurité" et the section called "IPv6". Si vous choisissez la famille `IPv6`, vous ne pouvez pas spécifier une plage d'adresses à partir de laquelle Kubernetes doit attribuer les adresses de service `IPv6`, comme c'est possible avec la famille `IPv4`. Kubernetes attribue les adresses de service à partir de la plage d'adresses locales uniques (`fc00::/7`).

Créer un cluster - AWS console

1. Ouvrez la console Amazon EKS.
2. Choisissez Add cluster (Ajouter un cluster), puis choisissez Create (Créer).
3. Sous Options de configuration, sélectionnez Configuration personnalisée
   - Pour plus d'informations sur la création rapide d'un cluster avec le mode automatique EKS, consultez the section called "Console de gestion".
4. Sous Mode automatique EKS, désactivez l'option Utiliser le mode automatique EKS.

- Pour plus d'informations sur la création d'un cluster du mode automatique EKS avec une configuration personnalisée, consultez the section called "Création d'un cluster Auto".

5. Sur la page Configurer le cluster, renseignez les champs suivants :
   - Nom : nom de votre cluster. Le nom ne peut contenir que des caractères alphanumériques (sensibles à la casse), des tirets et des traits de soulignement. Il doit commencer par un caractère alphanumérique et ne peut pas dépasser 100 caractères. Le nom doit être unique dans la AWS région et le AWS compte dans lesquels vous créez le cluster.
   - Rôle IAM du cluster : choisissez le rôle IAM du cluster Amazon EKS que vous avez créé pour permettre au plan de contrôle Kubernetes de gérer AWS les ressources en votre nom.
   - Version Kubernetes : version de Kubernetes à utiliser pour votre cluster. Nous vous recommandons de sélectionner la dernière version, sauf si vous avez besoin d'une version antérieure.
   - Type de prise en charge : la politique de version Kubernetes que vous souhaitez définir pour votre cluster. Si vous souhaitez que votre cluster fonctionne uniquement sur une version en prise en charge standard, sélectionnez Prise en charge standard. Si vous souhaitez que votre cluster bénéficie d'une prise en charge étendue à la fin de la période de prise en charge standard, sélectionnez Prise en charge étendue. Si vous sélectionnez une version de Kubernetes déjà en prise en charge étendue, vous ne pouvez pas choisir la prise en charge standard comme option.
   - Chiffrement des secrets : (facultatif) choisissez d'activer le chiffrement des secrets Kubernetes à l'aide d'une clé KMS. Vous pouvez également activer cette option une fois que vous avez créé votre cluster. Avant d'activer cette fonctionnalité, assurez-vous d'avoir pris connaissance des informations mentionnées dans the section called "Activer le chiffrement secret".
   - Identifications◦: (facultatif) ajoutez des identifications à votre cluster. Pour de plus amples informations, veuillez consulter the section called "Balisage de vos ressources ".
   - Changement de zone ARC : (facultatif) vous pouvez utiliser le contrôleur de récupération d'application Route 53 pour atténuer les interruptions dans les zones de disponibilité défaillantes. Pour de plus amples informations, veuillez consulter the section called "En savoir plus sur le changement de zone".
6. Dans la section Accès au cluster de la page de configuration du cluster, renseignez les champs suivants :
   - Amorcer l'accès administrateur au cluster : le créateur du cluster devient automatiquement administrateur Kubernetes. Si vous souhaitez désactiver cet accès automatique, sélectionnez Refuser l'accès administrateur au cluster.

- Mode d'authentification du cluster : déterminez comment vous souhaitez accorder aux utilisateurs et aux rôles IAM l'accès à APIs Kubernetes. Pour de plus amples informations, veuillez consulter the section called “Définition du mode d’authentification du cluster”.

  Lorsque vous avez terminé d'utiliser cette page, choisissez Suivant.

7. Sur la page Spécifier les réseaux sélectionnez des valeurs pour les champs suivants :

- VPC : choisissez un VPC existant qui répond aux exigences relatives au VPC Amazon EKS pour y créer votre cluster. Avant de sélectionner un VPC, nous vous recommandons de bien connaître toutes les exigences et considérations de the section called “Exigences requises pour le VPC et les sous-réseaux”. Vous ne pouvez pas modifier le VPC que vous souhaitez utiliser après la création du cluster. Si aucun VPCs n'est répertorié, vous devez d'abord en créer un. Pour de plus amples informations, veuillez consulter the section called “Création d’un VPC”.
- Sous-réseaux : par défaut, tous les sous-réseaux disponibles dans le VPC spécifié dans le champ précédent sont présélectionnés. Vous devez en sélectionner au moins deux.

  Les sous-réseaux que vous choisissez doivent respecter les exigences relatives aux sous-réseaux d'Amazon EKS. Avant de sélectionner les sous-réseaux, assurez-vous de bien comprendre toutes les exigences et considérations relatives aux VPC et sous-réseaux Amazon EKS.

  Groupes de sécurité : (facultatif) spécifiez un ou plusieurs groupes de sécurité créant des interfaces réseau auxquelles vous souhaitez associer Amazon EKS.

  Que vous choisissiez ou non un groupe de sécurité, Amazon EKS crée un groupe de sécurité qui permet la communication entre votre cluster et votre VPC. Amazon EKS associe ce groupe de sécurité, et tout ce que vous choisissez, aux interfaces réseau qu'il crée. Pour plus d'informations sur le groupe de sécurité de cluster créé par Amazon EKS, consultez the section called “Exigences de groupe de sécurité”. Vous pouvez modifier les règles dans le groupe de sécurité de cluster créé par Amazon EKS.
- Choisissez la famille d'adresses IP du cluster : vous pouvez choisir l'une ou l'autre IPv4et IPv6.

  Kubernetes attribue par défaut des adresses `IPv4` aux pods et aux services. Avant de décider d’utiliser la famille `IPv6`, assurez-vous de bien connaître toutes les considérations et exigences mentionnées dans les rubriques Exigences et considérations relatives au VPC, the section called “Exigences et considérations requises pour les sous-réseaux”, the section called “Exigences de groupe de sécurité” et the section called “IPv6”. Si vous choisissez la famille `IPv6`, vous ne pouvez pas spécifier une plage d’adresses à partir de laquelle Kubernetes

doit attribuer les adresses de service IPv6, comme c'est possible avec la famille IPv4. Kubernetes attribue les adresses de service à partir de la plage d'adresses locales uniques (fc00::/7).

- (Facultatif) Choisissez Configurer la plage d'adresses IP du service Kubernetes et spécifiez une gamme de services **IPv4**.

  La spécification de votre propre plage peut aider à prévenir les conflits entre les services Kubernetes et d'autres réseaux appairés ou connectés à votre VPC. Entrez une plage en notation CIDR. Par exemple : 10.2.0.0/16.

  Le bloc d'adresse CIDR doit répondre aux critères suivants :

  - Être situé dans l'une des plages suivantes : 10.0.0.0/8, 172.16.0.0/12, ou 192.168.0.0/16.
  - Avoir une taille minimale de /24 et une taille maximale de /12.
  - Ne pas chevaucher la plage du VPC pour vos ressources Amazon EKS.

Vous ne pouvez spécifier cette option que lorsque vous utilisez la famille d'adresses IPv4 et uniquement lors de la création du cluster. Si vous ne spécifiez pas cela, alors Kubernetes attribue des adresses IP de service à partir des blocs CIDR 10.100.0.0/16 ou 172.20.0.0/16.

- Pour Accès au point de terminaison de cluster, sélectionnez une option. Une fois votre cluster créé, vous pouvez modifier cette option. Avant de sélectionner une option autre que par défaut, assurez-vous de vous familiariser avec les options et leurs implications. Pour de plus amples informations, veuillez consulter the section called "Accès au point de terminaison de cluster".

  Lorsque vous avez terminé d'utiliser cette page, choisissez Suivant.

8. (Facultatif) Sur la page Configurer l'observabilité, choisissez quelles options de Métriques et de Journalisation du plan de contrôle doivent être activées. Par défaut, chaque type de journal est désactivé.

   - Pour plus d'informations sur l'option de métriques Prometheus, consultez the section called "Étape 1 : activer les métriques Prometheus".
   - Pour plus d'informations sur les options de Journalisation du plan de contrôle, consultez the section called "journaux de plan de contrôle ".

   Lorsque vous avez terminé d'utiliser cette page, choisissez Suivant.

9. Sur la page Select add-ons (Sélectionner les modules complémentaires), choisissez les modules complémentaires que vous souhaitez ajouter à votre cluster. Certains modules complémentaires

sont présélectionnés. Vous pouvez choisir autant de modules complémentaires Amazon EKS et de modules complémentaires AWS Marketplace que vous le souhaitez. Si les modules complémentaires AWS Marketplace que vous souhaitez installer ne figurent pas dans la liste, vous pouvez cliquer sur la numérotation des pages pour afficher des résultats supplémentaires ou rechercher les modules complémentaires AWS Marketplace disponibles en saisissant du texte dans le champ de recherche. Vous pouvez également filtrer les résultats par catégorie, fournisseur ou modèle de tarification, puis sélectionner les modules complémentaires affichés dans les résultats de recherche. Lorsque vous créez un cluster, vous pouvez afficher, sélectionner et installer tout module complémentaire compatible avec les identités de pod EKS, comme indiqué dans the section called "Pod Identity".

Lorsque vous avez terminé d'utiliser cette page, choisissez Suivant.

Certains modules complémentaires, tels qu'Amazon VPC, CoreDNS et kube-proxy, sont installés par défaut. Si vous désactivez l'un des modules complémentaires par défaut, cela peut affecter votre capacité à exécuter des applications Kubernetes.

10. Sur la page Configurer les paramètres de modules complémentaires sélectionnés, sélectionnez la version que vous voulez installer. Vous pourrez toujours effectuer une mise à jour vers une version ultérieure après la création du cluster.

    Pour les modules complémentaires compatibles avec EKS Pod Identities, vous pouvez utiliser la console pour générer automatiquement le rôle avec le nom, la politique AWS gérée et la politique de confiance préremplis spécifiquement pour le module complémentaire. Vous pouvez réutiliser des rôles existants ou créer de nouveaux rôles pour les modules complémentaires pris en charge. Pour les étapes permettant d'utiliser la console afin de créer des rôles pour les modules complémentaires qui prennent en charge les identités de pod EKS, consultez the section called "Créer un module complémentaire (AWS console)". Si un module complémentaire ne prend pas en charge l'identité du pod EKS, un message s'affiche avec des instructions pour utiliser l'assistant afin de créer les rôles IAM pour comptes de service (IRSA) après la création du cluster.

    Vous pourrez mettre à jour la configuration de chaque module complémentaire après la création du cluster. Pour plus d'informations sur la configuration des modules complémentaires, consultez the section called "Mettre à jour un module complémentaire". Lorsque vous avez terminé d'utiliser cette page, choisissez Suivant.

11. Sur la page Vérifier et créer, passez en revue les informations que vous avez saisies ou sélectionnées sur les pages précédentes. Si vous devez apporter des modifications, choisissez

Modifier. Quand vous êtes satisfait, choisissez Créer. Le champ État affiche EN COURS DE CRÉATION pendant que le cluster est provisionné.

> **Note**
>
> Il est possible que vous receviez un message d'erreur indiquant que l'une des zones de disponibilité de votre demande ne dispose pas d'une capacité suffisante pour créer un cluster Amazon EKS. Si cela se produit, la sortie de l'erreur contient les zones de disponibilité qui peuvent prendre en charge un nouveau cluster. Essayez à nouveau de créer votre cluster avec au moins deux sous-réseaux situés dans les zones de disponibilité prises en charge pour votre compte. Pour de plus amples informations, veuillez consulter the section called "Capacité insuffisante".

L'approvisionnement de cluster dure plusieurs minutes.

12. Continuez avec the section called "Étape 3 : mettre à jour kubeconfig"

Créer un cluster - AWS CLI

1. Créez votre cluster à l'aide de la commande suivante. Avant d'exécuter la commande, effectuez les remplacements suivants :
   - *region-code* Remplacez-le par la AWS région dans laquelle vous souhaitez créer votre cluster.
   - Remplacez *my-cluster* par un nom pour votre cluster. Le nom ne peut contenir que des caractères alphanumériques (sensibles à la casse), des tirets et des traits de soulignement. Il doit commencer par un caractère alphanumérique et ne peut pas dépasser 100 caractères. Le nom doit être unique dans la AWS région et le AWS compte dans lesquels vous créez le cluster.
   - Remplacez la version *1.33* par n'importe quelle version prise en charge par Amazon EKS.
   - Remplacez *111122223333* par l'ID de votre compte et *myAmazonEKSClusterRole* par le nom de votre rôle IAM de cluster.
   - Remplacez les valeurs de `subnetIds` par vos propres valeurs. Vous pouvez également en ajouter d'autres IDs. Vous devez spécifier au moins deux sous-réseaux IDs.

     Les sous-réseaux que vous choisissez doivent respecter les exigences relatives aux sous-réseaux d'Amazon EKS. Avant de sélectionner les sous-réseaux, assurez-vous de bien comprendre toutes les exigences et considérations relatives aux VPC et sous-réseaux Amazon EKS.

- Si vous ne voulez pas spécifier un ID de groupe de sécurité, supprimez `,securityGroupIds=sg-<ExampleID1>` depuis la commande. Si vous souhaitez spécifier un ou plusieurs groupes de sécurité IDs, remplacez les valeurs de par `securityGroupIds` les vôtres. Vous pouvez également en ajouter d'autres IDs.

  Que vous choisissiez ou non un groupe de sécurité, Amazon EKS crée un groupe de sécurité qui permet la communication entre votre cluster et votre VPC. Amazon EKS associe ce groupe de sécurité, et tout ce que vous choisissez, aux interfaces réseau qu'il crée. Pour plus d'informations sur le groupe de sécurité de cluster créé par Amazon EKS, consultez the section called “Exigences de groupe de sécurité”. Vous pouvez modifier les règles dans le groupe de sécurité de cluster créé par Amazon EKS.

  ```
  aws eks create-cluster --region region-code --name my-cluster --kubernetes-version
   1.33 \
     --role-arn arn:aws: iam::111122223333:role/myAmazonEKSClusterRole \
     --resources-vpc-config subnetIds=subnet-ExampleID1,subnet-
  ExampleID2,securityGroupIds=sg-ExampleID1
  ```

  > **Note**
  >
  > Il est possible que vous receviez un message d'erreur indiquant que l'une des zones de disponibilité de votre demande ne dispose pas d'une capacité suffisante pour créer un cluster Amazon EKS. Si cela se produit, la sortie de l'erreur contient les zones de disponibilité qui peuvent prendre en charge un nouveau cluster. Essayez à nouveau de créer votre cluster avec au moins deux sous-réseaux situés dans les zones de disponibilité prises en charge pour votre compte. Pour de plus amples informations, veuillez consulter the section called “Capacité insuffisante”.

  Les paramètres suivants sont des paramètres facultatifs qui, si nécessaire, doivent être ajoutés à la commande précédente. Vous pouvez activer ces options uniquement lorsque vous créez le cluster, et non après.

- Par défaut, EKS installe plusieurs modules complémentaires réseau lors de la création du cluster. Cela inclut le plug-in Amazon VPC CNI, CoreDNS et kube-proxy.

Si vous souhaitez désactiver l'installation de ces modules complémentaires réseau par défaut, utilisez le paramètre ci-dessous. Cela peut être utilisé comme alternative CNIs, comme le cilium. Pour plus d'informations, consultez la Référence de l'API EKS.

```
aws eks create-cluster --bootstrapSelfManagedAddons false
```

- Si vous souhaitez spécifier à partir de quel bloc de routage inter-domaines sans classe (CIDR) IPv4 Kubernetes attribue des adresses IP de service, vous devez le spécifier en ajoutant le `--kubernetes-network-config serviceIpv4Cidr=<cidr-block>` à la commande suivante.

  La spécification de votre propre plage peut aider à prévenir les conflits entre les services Kubernetes et d'autres réseaux appairés ou connectés à votre VPC. Entrez une plage en notation CIDR. Par exemple : `10.2.0.0/16`.

  Le bloc d'adresse CIDR doit répondre aux critères suivants :

  - Être situé dans l'une des plages suivantes : `10.0.0.0/8`, `172.16.0.0/12`, ou `192.168.0.0/16`.
  - Avoir une taille minimale de `/24` et une taille maximale de `/12`.
  - Ne pas chevaucher la plage du VPC pour vos ressources Amazon EKS.

Vous ne pouvez spécifier cette option que lorsque vous utilisez la famille d'adresses IPv4 et uniquement lors de la création du cluster. Si vous ne spécifiez pas cela, alors Kubernetes attribue des adresses IP de service à partir des blocs CIDR `10.100.0.0/16` ou `172.20.0.0/16`.

- Si vous créez un cluster et que vous souhaitez qu'il attribue des adresses IPv6 aux pods et aux services au lieu d'adresses IPv4, ajoutez `--kubernetes-network-config ipFamily=ipv6` à la commande suivante.

  Kubernetes attribue par défaut des adresses IPv4 aux pods et aux services. Avant de décider d'utiliser la famille IPv6, assurez-vous de bien connaître toutes les considérations et exigences mentionnées dans les rubriques Exigences et considérations relatives au VPC, the section called "Exigences et considérations requises pour les sous-réseaux", the section called "Exigences de groupe de sécurité" et the section called "IPv6". Si vous choisissez la famille IPv6, vous ne pouvez pas spécifier une plage d'adresses à partir de laquelle Kubernetes doit attribuer les adresses de service IPv6, comme c'est possible avec la famille IPv4. Kubernetes attribue les adresses de service à partir de la plage d'adresses locales uniques (`fc00::/7`).

2. Le provisionnement du cluster prend quelques minutes. Vous pouvez vérifier le statut de votre cluster avec la commande suivante.

```
aws eks describe-cluster --region region-code --name my-cluster --query
 "cluster.status"
```

   Ne passez pas à l'étape suivante tant que le résultat renvoyé n'est pas `ACTIVE`.

3. Continuez avec the section called "Étape 3 : mettre à jour kubeconfig"

Étape 3 : mettre à jour kubeconfig

1. Si vous avez créé votre cluster à l'aide de `eksctl`, vous pouvez ignorer cette étape. Cela est dû au fait que `eksctl` a déjà terminé cette étape pour vous. Activez `kubectl` pour communiquer avec votre cluster en ajoutant un nouveau contexte au fichier `kubectl config`. Pour plus d'informations sur la création et la mise à jour du fichier, consultez the section called "Accès au cluster avec kubectl".

```
aws eks update-kubeconfig --region region-code --name my-cluster
```

   L'exemple qui suit illustre un résultat.

```
Added new context arn:aws: eks:region-code:111122223333:cluster/my-cluster to /home/
username/.kube/config
```

2. Confirmez la communication avec votre cluster en exécutant la commande suivante.

```
kubectl get svc
```

   L'exemple qui suit illustre un résultat.

```
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.100.0.1   <none>        443/TCP   28h
```

Étape 4 : configuration du cluster

1. (Recommandé) Pour utiliser certains modules complémentaires Amazon EKS ou pour permettre à des charges de travail Kubernetes individuelles de disposer d'autorisations spécifiques de gestion des AWS identités et des accès (IAM), créez un fournisseur IAM OpenID Connect (OIDC) pour votre cluster. Vous n'avez besoin de créer un fournisseur OIDC IAM pour votre cluster qu'une seule fois. Pour en savoir plus sur les modules complémentaires Amazon EKS, consultez the section called "Modules complémentaires Amazon EKS". Pour en savoir plus sur l'attribution des autorisations IAM spécifiques à vos applications, consultez the section called "Informations d'identification auprès de l'IRSA".
2. (Recommandé) Configurez votre cluster pour le plugin Amazon VPC CNI pour Kubernetes avant de déployer des nœuds Amazon sur EC2 votre cluster. Par défaut, le plugin a été installé avec votre cluster. Lorsque vous ajoutez EC2 des nœuds Amazon à votre cluster, le plugin est automatiquement déployé sur chaque EC2 nœud Amazon que vous ajoutez. Ce plug-in nécessite l'ajout de l'une des politiques IAM suivantes à un rôle IAM. Si votre cluster utilise la famille `IPv4`, utilisez la politique IAM gérée AmazonEKS_CNI_Policy. Si votre cluster utilise la famille `IPv6`, utilisez une politique IAM que vous créez.

   Le rôle IAM auquel vous attachez la politique peut être le rôle IAM du nœud ou un rôle dédié utilisé uniquement pour le plugin. Nous vous recommandons d'associer la politique à ce rôle. Pour plus d'informations sur la création du rôle, consultez the section called "Configuration pour IRSA" ou Rôle IAM du nœud Amazon EKS.
3. Si vous avez déployé votre cluster à l'aide du AWS Management Console, vous pouvez ignorer cette étape. AWS Management Console Déploie le plug-in Amazon VPC CNI pour les modules complémentaires Kubernetes, CoreDNS et Amazon EKS, par défaut. `kube-proxy`

   Si vous déployez votre cluster à l'aide de l'une `eksctl` ou de l'autre ou de la AWS CLI, le plug-in Amazon VPC CNI pour Kubernetes, CoreDNS et les modules complémentaires autogérés sont déployés. `kube-proxy` Vous pouvez migrer le plug-in Amazon VPC CNI pour Kubernetes, CoreDNS, et les modules complémentaires autogérés `kube-proxy` déployés avec votre cluster vers des modules complémentaires Amazon EKS. Pour de plus amples informations, veuillez consulter the section called "Modules complémentaires Amazon EKS".
4. (Facultatif) Si ce n'est pas déjà fait, vous pouvez activer les métriques Prometheus pour votre cluster. Pour plus d'informations, consultez Création d'un scraper dans le Guide de l'utilisateur Amazon Managed Service for Prometheus.

5. Si vous prévoyez de déployer des charges de travail utilisant des volumes Amazon EBS, vous devez installer le module complémentaire Amazon EBS CSI dans votre cluster avant de déployer ces charges de travail.

Étapes suivantes

- Le principal IAM qui a créé le cluster est le seul principal à avoir accès au cluster. Accordez des autorisations à d'autres principaux IAM afin qu'ils puissent accéder à votre cluster.
- Si le principal IAM qui a créé le cluster ne dispose que des autorisations IAM minimales référencées dans les prérequis, vous pouvez alors ajouter des autorisations Amazon EKS supplémentaires pour ce principal. Pour plus d'informations sur l'octroi d'autorisations Amazon EKS aux principaux IAM, consultez la rubrique the section called "Référence IAM".
- Si vous souhaitez que le principal IAM qui a créé le cluster, ou tout autre principal, puisse visualiser les ressources Kubernetes dans la console Amazon EKS, accordez les autorisations requises à ces entités.
- Si vous souhaitez que les nœuds et les principaux IAM accèdent à votre cluster depuis votre VPC, activez le point de terminaison privé pour votre cluster. Le point de terminaison public est activé par défaut. Vous pouvez désactiver le point de terminaison public après avoir activé le point de terminaison privé, si vous le souhaitez. Pour de plus amples informations, veuillez consulter the section called "Accès au point de terminaison de cluster".
- Activer le chiffrement des secrets pour votre cluster.
- Configurer la journalisation de votre cluster.
- Ajouter des nœuds à votre cluster.

Plan de contrôle provisionné Amazon EKS

Présentation de

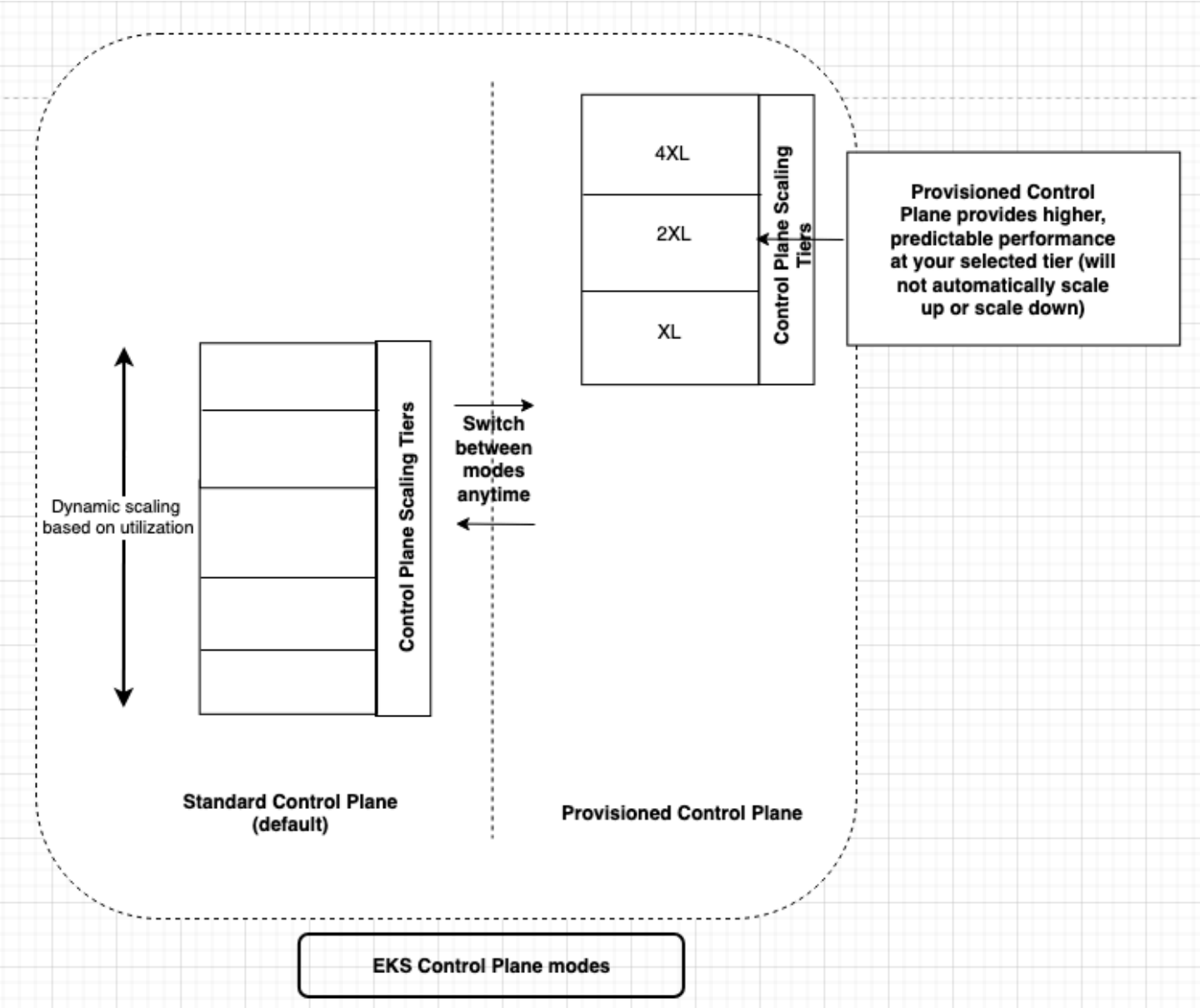
Le plan de contrôle provisionné Amazon EKS est une fonctionnalité qui permet aux administrateurs de clusters de choisir parmi un ensemble de niveaux de dimensionnement et de désigner le niveau de leur choix pour des performances très élevées et prévisibles depuis le plan de contrôle du cluster. Cela permet aux administrateurs de clusters de s'assurer que le plan de contrôle est toujours approvisionné avec la capacité spécifiée.

Amazon EKS propose deux modes de fonctionnement pour le plan de contrôle de votre cluster. Par défaut, les clusters Amazon EKS utilisent le mode standard, dans lequel le plan de contrôle augmente ou diminue automatiquement en fonction des exigences de votre charge de travail. Le mode standard alloue de manière dynamique une capacité suffisante du plan de contrôle pour répondre aux besoins de votre charge de travail et constitue la solution recommandée dans la plupart des cas d'utilisation. Toutefois, pour les charges de travail spécialisées qui ne peuvent tolérer aucune variabilité des performances due à la mise à l'échelle du plan de contrôle ou celles qui nécessitent une très grande capacité du plan de contrôle, vous pouvez éventuellement utiliser le mode provisionné. Le mode provisionné vous permet de préallouer une capacité du plan de contrôle toujours prête à répondre aux exigences de charge de travail exigeantes.

> **Note**
>
> Le mode provisionné est un mode de fonctionnement du plan de contrôle supplémentaire en plus du mode standard par défaut. L'introduction du mode provisionné ne modifie pas le comportement du mode standard.

Avec le plan de contrôle provisionné EKS, les administrateurs du cluster peuvent préprovisionner la capacité du plan de contrôle souhaitée à l'avance, offrant ainsi des performances prévisibles et élevées à partir du plan de contrôle du cluster, qui est toujours disponible. Le plan de contrôle provisionné EKS permet également aux administrateurs de clusters de fournir la même capacité de plan de contrôle dans tous les environnements, qu'il s'agisse de sites de préparation, de production ou de reprise après sinistre. Cela est important pour garantir que les performances du plan de contrôle obtenues dans tous les environnements sont cohérentes et prévisibles. Enfin, EKS Provisioned Control Plane vous donne accès à de très hauts niveaux de performance du plan de contrôle, ce qui permet d'exécuter des charges de travail d'IA extrêmement évolutives, des calculs à haute performance et des charges de travail de traitement de données à grande échelle sur Kubernetes.

Tous les clusters Amazon EKS existants et nouveaux fonctionnent en mode standard par défaut. Pour les clusters nécessitant des performances élevées et prévisibles de la part du plan de contrôle, vous pouvez choisir d'utiliser la fonction EKS Provisioned Control Plane. Vous serez facturé au taux horaire correspondant au niveau de mise à l'échelle du plan de contrôle concerné, en plus des frais horaires EKS pour le support standard ou prolongé. Pour plus d'informations sur la tarification, consultez la section Tarification d'Amazon EKS.



Cas d’utilisation

Le plan de contrôle provisionné EKS est conçu pour répondre à des scénarios spécifiques dans lesquels des performances élevées et prévisibles du plan de contrôle sont essentielles à vos opérations. La compréhension de ces cas d'utilisation peut vous aider à déterminer si EKS Provisioned Control Plane est la solution adaptée à vos charges de travail.

Charges de travail critiques pour les performances : pour les charges de travail qui nécessitent une latence minimale et des performances maximales de la part du plan de contrôle Kubernetes, le plan de contrôle provisionné EKS fournit une capacité qui élimine la variabilité des performances grâce à la mise à l'échelle du plan de contrôle.

Charges de travail extrêmement évolutives — Si vous exécutez des charges de travail hautement évolutives telles que la formation et l'inférence basées sur l'IA, le calcul haute performance ou le traitement de données à grande échelle nécessitant l'exécution d'un grand nombre de nœuds dans le cluster, le plan de contrôle provisionné fournit la capacité de plan de contrôle nécessaire pour prendre en charge ces charges de travail exigeantes.

Événements à forte demande prévus — Lorsque vous vous attendez à une augmentation soudaine du nombre de demandes de plans de contrôle en raison d'un événement à venir, tel que des ventes ou des promotions sur le commerce électronique, des lancements de produits, des périodes de shopping des fêtes ou des événements sportifs ou de divertissement majeurs, le plan de contrôle provisionné vous permet d'augmenter la capacité de votre plan de contrôle à l'avance. Cette approche proactive garantit que votre plan de contrôle est prêt à faire face à l'augmentation de la charge sans attendre le dimensionnement automatique pour répondre à la demande.

Cohérence de l'environnement : le plan de contrôle provisionné vous permet de faire correspondre la capacité et les performances du plan de contrôle entre les environnements de préparation et de production, ce qui vous aide à identifier les problèmes potentiels à un stade précoce avant le déploiement en production. En conservant le même niveau de plan de contrôle dans tous les environnements, vous pouvez vous assurer que les résultats des tests reflètent précisément le comportement de production, réduisant ainsi le risque de surprises liées aux performances lors du déploiement.

Reprise après sinistre et continuité des activités : pour les scénarios de reprise après sinistre, le plan de contrôle provisionné vous permet de fournir des environnements de basculement avec le même niveau de capacité que votre environnement principal. Cela garantit une interruption minimale et une restauration rapide en cas de basculement, car votre cluster de reprise après sinistre aura des caractéristiques de performance de plan de contrôle identiques à celles de votre cluster de production dès son activation.

Niveaux de dimensionnement du plan de contrôle

EKS Provisioned Control Plane propose des niveaux de scalabilité nommés selon les tailles de t-shirt (XL, 2XL, 4XL). Chaque niveau définit sa capacité par le biais de trois attributs Kubernetes clés qui déterminent les caractéristiques de performance du plan de contrôle de votre cluster. La compréhension de ces attributs vous permet de sélectionner le niveau adapté aux exigences de votre charge de travail.

La simultanéité des demandes d'API mesure le nombre de demandes que le serveur d'API du plan de contrôle Kubernetes peut traiter simultanément, ce qui est essentiel pour les charges de travail à haut débit.

Le taux de planification des pods indique la rapidité avec laquelle le planificateur Kubernetes par défaut peut planifier les pods sur les nœuds, mesurée en pods par seconde.

La taille de la base de données du cluster indique l'espace de stockage alloué à etcd, la base de données qui contient l'état et les métadonnées du cluster.

Lorsque vous configurez le plan de contrôle de votre cluster sur un certain niveau de scalabilité à l'aide du plan de contrôle provisionné, EKS garantit que le plan de contrôle de votre cluster maintient les limites correspondant à ce niveau. Les limites des niveaux de dimensionnement du plan de contrôle varient selon la version de Kubernetes, comme indiqué dans les tableaux suivants.

EKS v1.28 et v1.29

Niveau de mise à l'échelle du plan de contrôle provisionné	Concurrence des demandes d'API (sièges)	Débit de planification des pods (pods/sec)	Taille de la base de données du cluster (Go)
XL	1700	100	16
2XL	3400	100	16
4XL	6800	100	16

EKS v1.30 et versions ultérieures

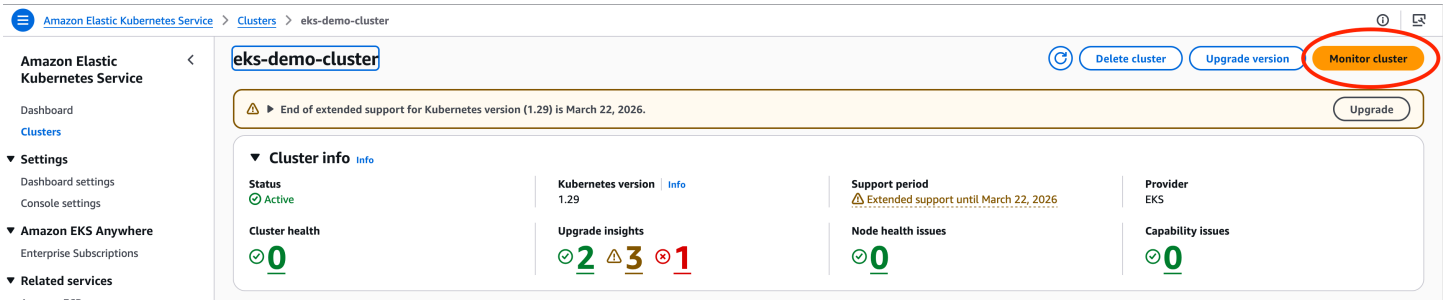
Niveau de mise à l'échelle du plan de contrôle provisionné	Concurrence des demandes d'API (sièges)	Débit de planification des pods (pods/sec)	Taille de la base de données du cluster (Go)
XL	1700	167	16
2XL	3400	283	16
4XL	6800	400	16

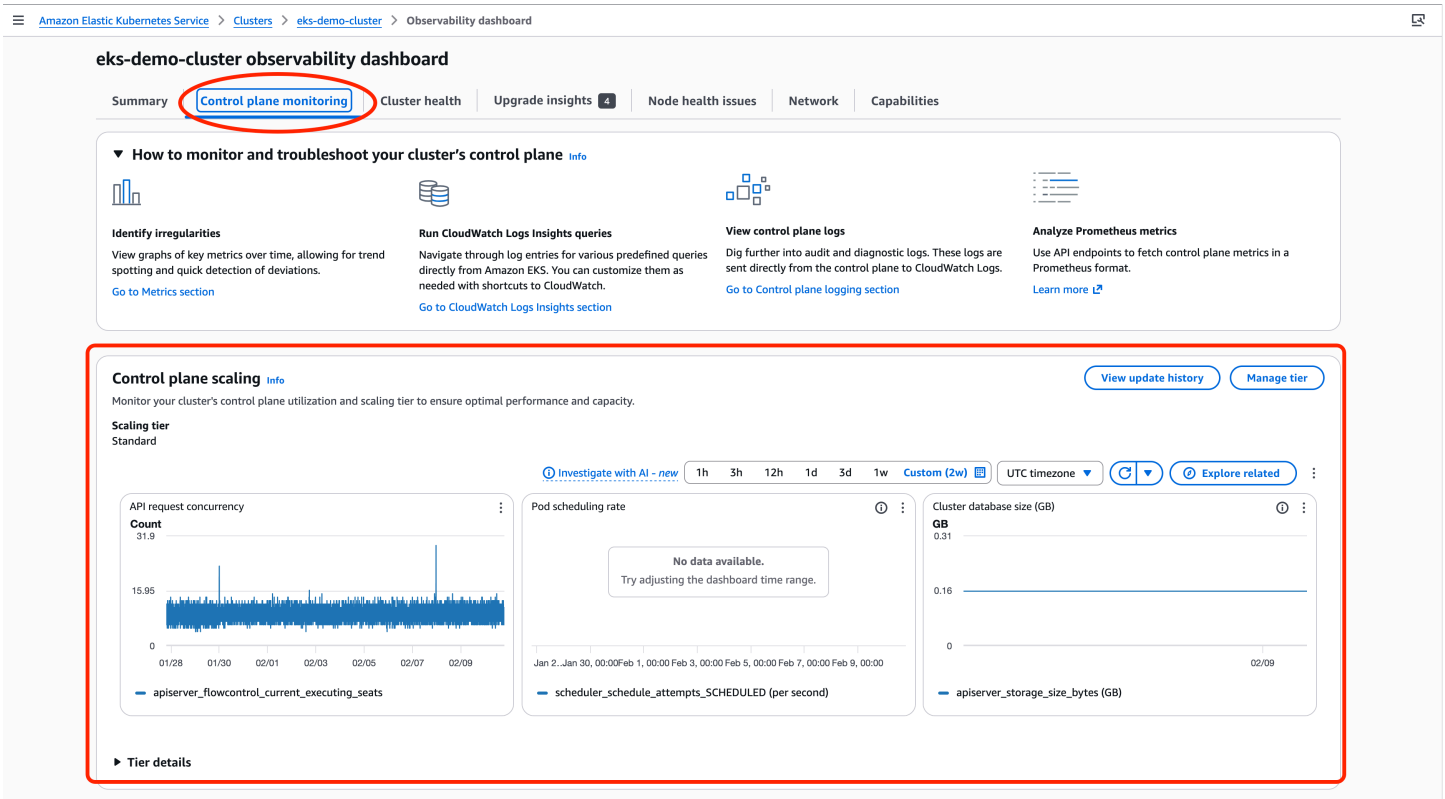
Surveillance du plan de contrôle, échelonnement de l'utilisation des niveaux

Amazon EKS fournit plusieurs indicateurs pour vous aider à surveiller l'utilisation des niveaux de votre plan de contrôle. Ces métriques sont publiées sous forme de CloudWatch métriques Amazon et sont accessibles via la console EKS CloudWatch et EKS. De plus, ces métriques peuvent être extraites du point de terminaison Prometheus de votre cluster EKS (voir ici).

	Prometheus Metric	CloudWatch Métrique
Concurrence des demandes d'API	apiserver_flowcontrol_current_executing_seats	apiserver_flowcontrol_current_executing_seats
Taux de planification des pods	scheduler_schedule_attempts_total	scheduler_schedule_attempts total, Scheduler_Schedule_Attempts_Scheduled, Scheduler_Schedule_Attempts_Unschedulable
Taille de base de données du cluster	apiserver_storage_size_bytes	apiserver_storage_size_bytes

Vous pouvez consulter l'utilisation du plan de contrôle dans la console Amazon EKS. Sur la page de présentation de votre cluster, choisissez Surveiller le cluster pour accéder au tableau de bord d'observabilité, puis sélectionnez l'onglet Surveillance du plan de contrôle pour afficher l'utilisation du plan de contrôle dans la section Mise à l'échelle du plan de contrôle.





Comprendre la capacité du niveau Tier par rapport aux performances réelles

Lorsque vous sélectionnez un niveau de dimensionnement du plan de contrôle provisionné, les attributs du niveau représentent les configurations sous-jacentes qu'Amazon EKS applique à votre plan de contrôle. Cependant, les performances réelles que vous obtenez dépendent de vos modèles de charge de travail spécifiques, de vos configurations et du respect des meilleures pratiques de Kubernetes. Par exemple, alors qu'un niveau 4XL configure la priorité et l'équité des API (APF) avec 6 800 postes de requêtes simultanés, le débit de demandes réel que vous obtenez à partir du plan de contrôle dépend des types d'opérations effectuées. Par exemple, Kubernetes pénalise davantage les requêtes de liste que les requêtes de type get, de sorte que le nombre effectif de demandes de liste traitées simultanément par le plan de contrôle est inférieur à celui des demandes get (voir ici). De même, bien que le planificateur par défaut QPS soit défini sur 400 pour un niveau 4XL, le taux réel de planification de vos pods dépend de facteurs tels que la disponibilité des nœuds et leur état de santé pour la planification. Pour obtenir des performances optimales, assurez-vous que vos applications respectent les meilleures pratiques de Kubernetes (voir ici) et qu'elles sont correctement configurées en fonction des caractéristiques de votre charge de travail.

Considérations

- Capacité du plan de contrôle standard — Le mode plan de contrôle standard d'EKS offre le meilleur rapport qualité/prix et constitue l'option recommandée pour la grande majorité des cas d'utilisation. Toutefois, pour les charges de travail spécialisées qui ne peuvent tolérer aucune variabilité des performances due à la mise à l'échelle du plan de contrôle ou celles qui nécessitent une très grande capacité du plan de contrôle, vous pouvez éventuellement envisager d'utiliser le mode provisionné.
- Inscription requise : les clusters existants ne passeront pas automatiquement du plan de contrôle standard à un niveau plus onéreux du plan de contrôle provisionné EKS. Vous devez explicitement adhérer à l'un des nouveaux niveaux de dimensionnement du plan de contrôle provisionné par EKS.
- Restriction de sortie : le mode plan de contrôle standard prend en charge jusqu'à 8 Go de taille de base de données de cluster (etcd). Si la taille de la base de données de votre cluster dépasse 8 Go en mode provisionné, vous ne pouvez pas revenir en mode standard tant que vous n'avez pas réduit la taille de la base de données à moins de 8 Go. Par exemple, si vous utilisez 14 Go de stockage de base de données en mode provisionné, vous devez d'abord réduire l'utilisation de votre base de données à moins de 8 Go avant de revenir en mode standard.
- Pas de mise à l'échelle automatique des niveaux : le plan de contrôle provisionné EKS ne passe pas automatiquement d'un niveau à l'autre. Une fois que vous avez sélectionné un niveau de scalabilité, le plan de contrôle de votre cluster reste attaché à ce niveau, garantissant ainsi des performances constantes et prévisibles. Cependant, vous avez la possibilité de mettre en œuvre votre propre solution de dimensionnement automatique en surveillant les indicateurs d'utilisation des niveaux et en utilisant le plan de contrôle provisionné EKS APIs pour augmenter ou diminuer lorsque ces indicateurs dépassent les seuils que vous définissez, vous donnant ainsi un contrôle total sur votre stratégie de dimensionnement et l'optimisation des coûts.
- Affichage du niveau actuel : vous pouvez utiliser la console Amazon EKS, la CLI Amazon Web Services ou l'API pour afficher le niveau de dimensionnement actuel du plan de contrôle. Dans la CLI, vous pouvez exécuter la `describe-cluster` commande suivante : `aws eks describe-cluster --name cluster-name`
- Temps de transition entre les niveaux : vous pouvez utiliser la console Amazon EKS, Amazon EKS APIs ou la CLI pour quitter ou passer d'un niveau de dimensionnement à un autre. Amazon EKS a introduit un nouveau type de mise à jour de cluster appelé`ScalingTierConfigUpdate`, que vous pouvez inspecter pour suivre la progression de la transition. Après avoir exécuté une commande de changement de niveau, vous pouvez répertorier les mises à jour sur le cluster pour

voir une nouvelle mise à jour de type `ScalingTierConfigUpdate` avec statut`Updating`. Le statut passe à `Successful` la fin de la mise à jour ou en `Failed` cas d'erreur. Le champ d'erreur de la mise à jour indique la raison de l'échec. Il n'y a aucune restriction quant à la fréquence à laquelle vous pouvez passer d'un niveau à l'autre. La modification du niveau du plan de contrôle prend plusieurs minutes.

- Sélection du niveau optimal : pour déterminer le niveau de mise à l'échelle du plan de contrôle provisionné optimal pour votre cluster, vous pouvez effectuer des tests de charge en provisionnant votre cluster sur le niveau le plus élevé (4XL). Effectuez ensuite un test de charge pour simuler le pic de demande sur le plan de contrôle de votre cluster. Observez les mesures d'utilisation du niveau du plan de contrôle au pic de charge et utilisez ces observations comme facteur directeur pour sélectionner le niveau approprié pour le mode provisionné.
- Tarification du plan de contrôle provisionné : vous serez facturé au taux horaire correspondant au niveau de dimensionnement du plan de contrôle provisionné auquel se trouve votre cluster. Cela s'ajoute aux frais horaires de support standard ou prolongé. Consultez la page de tarification d'Amazon EKS pour plus de détails.
- Niveau de mise à l'échelle supérieur : si vous avez l'intention d'exécuter votre cluster sur un niveau de mise à l'échelle supérieur à 4XL, contactez l'équipe chargée de votre compte Amazon Web Services pour obtenir des informations supplémentaires sur les tarifs.
- Support des versions et des régions de Kubernetes : le plan de contrôle provisionné par EKS est pris en charge dans toutes les régions commerciales d'Amazon Web Services GovCloud et en Chine. Le plan de contrôle provisionné fonctionne sur EKS v1.28 et versions ultérieures.

Commencer à utiliser le plan de contrôle provisionné Amazon EKS

Ce guide vous explique les étapes de configuration et d'utilisation du plan de contrôle provisionné EKS à l'aide de la AWS CLI et AWS Management Console.

Conditions préalables

Avant de commencer, assurez-vous de disposer des éléments suivants :

- AWS CLI : outil de ligne de commande permettant de travailler avec AWS des services, notamment Amazon EKS. Pour plus d'informations, consultez la section Installation dans le guide de l'utilisateur de l'interface de ligne de AWS commande. Après avoir installé la AWS CLI, nous vous recommandons de la configurer également. Pour plus d'informations, consultez la section Configuration rapide avec aws configure dans le Guide de l'utilisateur de l'interface de ligne de

AWS commande. Notez que la AWS CLI v2 est requise pour utiliser l'option update-kubeconfig présentée sur cette page.

- Autorisations IAM requises : le principal de sécurité IAM que vous utilisez doit disposer des autorisations nécessaires pour utiliser les rôles IAM Amazon EKS, les rôles liés à un service, AWS CloudFormation un VPC et les ressources associées. Pour plus d'informations, consultez les sections Actions et utilisation de rôles liés à un service dans le guide de l'utilisateur IAM. Vous devez effectuer toutes les étapes de ce guide avec le même utilisateur. Exécutez la commande suivante pour vérifier l'utilisateur actuel :

```
aws sts get-caller-identity
```

> **Note**
>
> Nous vous recommandons de terminer les étapes de cette rubrique dans un shell Bash. Si vous n'utilisez pas de shell Bash, certaines commandes de script telles que les caractères de continuation de ligne et la façon dont les variables sont définies et utilisées nécessitent un ajustement pour votre shell. En outre, les règles de votre shell en matière de guillemets peuvent être différentes. Pour plus d'informations, consultez la section Utilisation de guillemets avec des chaînes dans la AWS CLI du Guide de l'utilisateur de l'interface de ligne de AWS commande.

Plan de contrôle provisionné EKS - CLI AWS

Créez un cluster avec le niveau de mise à l'échelle du plan de contrôle provisionné par EKS

```
aws eks create-cluster --name prod-cluster \
--role-arn arn:aws:iam::012345678910:role/eks-service-role-AWSServiceRoleForAmazonEKS-
J7ONKE3BQ4PI \
--resources-vpc-config subnetIds=subnet-6782e71e,subnet-
e7e761ac,securityGroupIds=sg-6979fe18 \
--control-plane-scaling-config tier=tier-xl
```

Réponse :

```
{
    "cluster": {
        "name": "my-eks-cluster",
```

```
        "arn": "arn:aws:eks:us-east-2:111122223333:cluster/my-eks-cluster",
        "createdAt": "2024-03-14T11:31:44.348000-04:00",
        "version": "1.26",
        "endpoint": "https://JSA79429HJDASKJDJ8223829MNDNASW.yl4.us-
east-2.eks.amazonaws.com",
        "roleArn": "arn:aws:iam::111122223333:role/eksctl-my-eks-cluster-cluster-
ServiceRole-zMF6CBakwwbW",
        "resourcesVpcConfig": {
            "subnetIds": [
                "subnet-0fb75d2d8401716e7",
                "subnet-02184492f67a3d0f9",
                "subnet-04098063527aab776",
                "subnet-0e2907431c9988b72",
                "subnet-04ad87f71c6e5ab4d",
                "subnet-09d912bb63ef21b9a"
            ],
            "securityGroupIds": [
                "sg-0c1327f6270afbb36"
            ],
            "clusterSecurityGroupId": "sg-01c84d09d70f39a7f",
            "vpcId": "vpc-0012b8e1cc0abb17d",
            "endpointPublicAccess": true,
            "endpointPrivateAccess": true,
            "publicAccessCidrs": [
                "22.19.18.2/32"
            ]
        },
        "controlPlaneScalingConfig": {
            "tier": "tier-xl"
        },
        "kubernetesNetworkConfig": {
            "serviceIpv4Cidr": "10.100.0.0/16",
            "ipFamily": "ipv4"
        },
        "logging": {
            "clusterLogging": [
                {
                    "types": [
                        "api",
                        "audit",
                        "authenticator",
                        "controllerManager",
                        "scheduler"
                    ],
```

```
                    "enabled": true
                }
            ]
        },
        "identity": {
            "oidc": {
                "issuer": "https://oidc.eks.us-east-2.amazonaws.com/id/
JSA79429HJDASKJDJ8223829MNDNASW"
            }
        },
        "status": "CREATING",
        "certificateAuthority": {
            "data": "CA_DATA_STRING..."
        },
        "platformVersion": "eks.14",
        "tags": {
            "aws:cloudformation:stack-name": "eksctl-my-eks-cluster-cluster",
            "alpha.eksctl.io/cluster-name": "my-eks-cluster",
            "karpenter.sh/discovery": "my-eks-cluster",
            "aws:cloudformation:stack-id": "arn:aws:cloudformation:us-
east-2:111122223333:stack/eksctl-my-eks-cluster-cluster/e752ea00-e217-11ee-
beae-0a9599c8c7ed",
            "auto-delete": "no",
            "eksctl.cluster.k8s.io/v1alpha1/cluster-name": "my-eks-cluster",
            "EKS-Cluster-Name": "my-eks-cluster",
            "alpha.eksctl.io/cluster-oidc-enabled": "true",
            "aws:cloudformation:logical-id": "ControlPlane",
            "alpha.eksctl.io/eksctl-version": "0.173.0-dev
+a7ee89342.2024-03-01T03:40:57Z",
            "Name": "eksctl-my-eks-cluster-cluster/ControlPlane"
        },
        "health": {
            "issues": []
        },
        "accessConfig": {
            "authenticationMode": "API_AND_CONFIG_MAP"
        }
    }
}
```

Afficher le niveau de mise à l'échelle du plan de contrôle du cluster

```
aws eks describe-cluster --name prod-cluster
```

Réponse :


```
{
    "cluster": {
        "name": "my-eks-cluster",
        "arn": "arn:aws:eks:us-east-2:111122223333:cluster/my-eks-cluster",
        "createdAt": "2024-03-14T11:31:44.348000-04:00",
        "version": "1.26",
        "endpoint": "https://JSA79429HJDASKJDJ8223829MNDNASW.yl4.us-
east-2.eks.amazonaws.com",
        "roleArn": "arn:aws:iam::111122223333:role/eksctl-my-eks-cluster-cluster-
ServiceRole-zMF6CBakwwbW",
        "resourcesVpcConfig": {
            "subnetIds": [
                "subnet-0fb75d2d8401716e7",
                "subnet-02184492f67a3d0f9",
                "subnet-04098063527aab776",
                "subnet-0e2907431c9988b72",
                "subnet-04ad87f71c6e5ab4d",
                "subnet-09d912bb63ef21b9a"
            ],
            "securityGroupIds": [
                "sg-0c1327f6270afbb36"
            ],
            "clusterSecurityGroupId": "sg-01c84d09d70f39a7f",
            "vpcId": "vpc-0012b8e1cc0abb17d",
            "endpointPublicAccess": true,
            "endpointPrivateAccess": true,
            "publicAccessCidrs": [
                "22.19.18.2/32"
            ]
        },
        "controlPlaneScalingConfig": {
            "tier": "tier-xl"
        },
        "kubernetesNetworkConfig": {
            "serviceIpv4Cidr": "10.100.0.0/16",
            "ipFamily": "ipv4"
        },
        "logging": {
            "clusterLogging": [
                {
                    "types": [
                        "api",
```

```
                        "audit",
                        "authenticator",
                        "controllerManager",
                        "scheduler"
                    ],
                    "enabled": true
                }
            ]
        },
        "identity": {
            "oidc": {
                "issuer": "https://oidc.eks.us-east-2.amazonaws.com/id/
JSA79429HJDASKJDJ8223829MNDNASW"
            }
        },
        "status": "ACTIVE",
        "certificateAuthority": {
            "data": "CA_DATA_STRING..."
        },
        "platformVersion": "eks.14",
        "tags": {
            "aws:cloudformation:stack-name": "eksctl-my-eks-cluster-cluster",
            "alpha.eksctl.io/cluster-name": "my-eks-cluster",
            "karpenter.sh/discovery": "my-eks-cluster",
            "aws:cloudformation:stack-id": "arn:aws:cloudformation:us-
east-2:111122223333:stack/eksctl-my-eks-cluster-cluster/e752ea00-e217-11ee-
beae-0a9599c8c7ed",
            "auto-delete": "no",
            "eksctl.cluster.k8s.io/v1alpha1/cluster-name": "my-eks-cluster",
            "EKS-Cluster-Name": "my-eks-cluster",
            "alpha.eksctl.io/cluster-oidc-enabled": "true",
            "aws:cloudformation:logical-id": "ControlPlane",
            "alpha.eksctl.io/eksctl-version": "0.173.0-dev
+a7ee89342.2024-03-01T03:40:57Z",
            "Name": "eksctl-my-eks-cluster-cluster/ControlPlane"
        },
        "health": {
            "issues": []
        },
        "accessConfig": {
            "authenticationMode": "API_AND_CONFIG_MAP"
        }
    }
```

```
}
```

Mettre à jour le cluster pour utiliser le plan de contrôle provisionné EKS

```
aws eks update-cluster-config --name prod-cluster \
--control-plane-scaling-config tier=tier-2xl
```

Réponse :

```
{
    "update": {
        "id": "7551c64b-1d27-4b1e-9f8e-c45f056eb6fd",
        "status": "InProgress",
        "type": "ScalingTierConfigUpdate",
        "params": [
            {
                "type": "UpdatedTier",
                "value": "tier-2xl"
            },
            {
                "type": "PreviousTier",
                "value": "tier-xl"
            }
        ],
        "createdAt": 1565807210.37,
        "errors": []
    }
}
```

Afficher la mise à jour du dimensionnement du plan de contrôle

```
aws eks list-updates --name example
```

Réponse :

```
{
    "updateIds": [
        "7551c64b-1d27-4b1e-9f8e-c45f056eb6fd1"
    ]
}
```

Passer du plan de contrôle provisionné au plan de contrôle standard

```
aws eks update-cluster-config --name prod-cluster \
--control-plane-scaling-config tier=standard
```

Réponse :

```
{
    "update": {
        "id": "7551c64b-1d27-4b1e-9f8e-c45f056eb6fd",
        "status": "InProgress",
        "type": "ScalingTierConfigUpdate",
        "params": [
            {
                "type": "UpdatedTier",
                "value": "standard"
            },
            {
                "type": "PreviousTier",
                "value": "tier-2xl"
            }
        ],
        "createdAt": 1565807210.37,
        "errors": []
    }
}
```

Plan de contrôle provisionné EKS - AWS Management Console

1. Ouvrez la console Amazon EKS.
2. Choisissez Créer un cluster.
3. Sous Options de configuration, sélectionnez Configuration personnalisée.
4. Faites défiler l'écran vers le bas jusqu'au niveau de dimensionnement du plan de contrôle. Sélectionnez Utiliser un niveau de dimensionnement pour activer le plan de contrôle provisionné.
5. Sélectionnez le niveau de mise à l'échelle du plan de contrôle que vous souhaitez configurer pour le cluster parmi les différentes options de niveau de mise à l'échelle, telles que XL, 2XL et 4XL.
6. Sélectionnez d'autres options de configuration de cluster selon vos besoins. À la dernière étape, sélectionnez Créer un cluster. Notez que la création du cluster peut prendre plusieurs minutes.

Préparation aux mises à niveau des versions Kubernetes et résolution des erreurs de configuration grâce aux informations sur les clusters

Les informations sur les clusters Amazon EKS détectent les problèmes et fournissent des recommandations pour vous aider à gérer votre cluster. Chaque cluster Amazon EKS est soumis à des contrôles automatiques et récurrents par rapport à une liste d'informations organisée par Amazon EKS. Ces vérifications d'informations sont entièrement gérées par Amazon EKS et incluent des recommandations pour résoudre les problèmes détectés.

Types d'informations sur le cluster

- Informations sur la configuration : identifient les erreurs de configuration dans la configuration de vos nœuds hybrides EKS susceptibles d'affecter les fonctionnalités de votre cluster ou de vos charges de travail.
- Informations sur la mise à niveau : identifient les problèmes susceptibles d'affecter votre capacité à effectuer la mise à niveau vers les nouvelles versions Kubernetes.

Considérations

- Fréquence : Amazon EKS actualise les informations sur les clusters toutes les 24 heures, ou vous pouvez les actualiser manuellement pour voir l'état le plus récent. Par exemple, vous pouvez actualiser manuellement les informations sur les clusters après avoir résolu un problème pour voir s'il a été résolu.
- Autorisations : Amazon EKS crée automatiquement une entrée d'accès au cluster pour obtenir des informations sur les clusters dans chaque cluster EKS. Cette entrée autorise EKS à afficher des informations sur votre cluster. Amazon EKS utilise ces informations pour générer d'autres informations. Pour de plus amples informations, consultez the section called "Amazon EKSCluster InsightsPolicy".

Cas d'utilisation

Les informations sur les clusters d'Amazon EKS offrent des vérifications automatisées pour maintenir la santé, la fiabilité et une configuration optimale de vos clusters Kubernetes. Vous trouverez ci-

dessous les principaux cas d'utilisation pour obtenir des informations sur les clusters, notamment la préparation à la mise à niveau et la résolution des problèmes de configuration.

Informations sur la mise à niveau

Les informations sur la mise à niveau constituent un type spécifique de vérifications d'informations au sein des informations sur les clusters. Ces vérifications renvoient des informations relatives à l'état de préparation de la mise à niveau des versions Kubernetes. Amazon EKS effectue des vérifications des informations sur la mise à niveau sur chaque cluster EKS.

> **Important**
>
> Amazon EKS a temporairement supprimé une fonctionnalité qui vous obligeait à utiliser un indicateur `--force` pour mettre à niveau votre cluster en cas de problèmes liés aux informations sur le cluster. Pour plus d'informations, consultez la section Annulation temporaire de l'application des informations sur la mise à niveau lors de la mise à jour de la version du cluster sur Github.
> Pour plus d'informations sur la mise à niveau de votre cluster, consultez the section called "Étape 3 : mettre à jour le plan de contrôle du cluster".

Avant de mettre à jour la version Kubernetes de votre cluster, utilisez l'onglet Informations sur la mise à niveau du tableau de bord d'observabilité dans la console Amazon EKS. Si votre cluster présente des problèmes identifiés, examinez-les et apportez les correctifs nécessaires. Chaque problème est accompagné de liens vers la documentation Amazon EKS et Kubernetes. Une fois le problème résolu, actualisez les informations sur les clusters à la demande pour obtenir les informations les plus récentes. Si tous les problèmes ont été résolus, mettez à jour votre cluster.

Amazon EKS renvoie des informations relatives à l'état de préparation de la mise à niveau des versions Kubernetes. Les informations sur la mise à niveau identifient les problèmes potentiels susceptibles d'avoir un impact sur les mises à niveau des clusters Kubernetes. Cela réduit au minimum le temps de préparation des administrateurs et améliore la fiabilité des applications sur les nouvelles versions Kubernetes. Les clusters sont automatiquement analysés par Amazon EKS sur la base d'une liste de problèmes connus pouvant impacter les mises à niveau Kubernetes. Amazon EKS met fréquemment à jour la liste des vérifications des informations en fonction des révisions des modifications apportées à chaque version Kubernetes.

Les informations de mise à niveau d'Amazon EKS accélèrent le processus de test et de vérification des nouvelles versions. Elles permettent également aux administrateurs de cluster et aux

développeurs d'applications d'adopter plus rapidement les nouvelles fonctionnalités Kubernetes en identifiant les points d'attention et en fournissant des recommandations de remédiation.

Informations sur la configuration

EKS Cluster Insights analyse automatiquement les clusters Amazon EKS dotés de nœuds hybrides afin d'identifier les problèmes de configuration qui entravent la communication entre le plan de contrôle Kubernetes et le webhook, les commandes kubectl telles que exec et logs, etc. Les informations sur la configuration permettent de détecter les problèmes et de fournir des recommandations pour y remédier, accélérant ainsi le délai de configuration de nœuds hybrides entièrement fonctionnels.

Mise en route

Pour consulter la liste des vérifications effectuées et les éventuels problèmes identifiés par Amazon EKS, vous pouvez utiliser la AWS Management Console, la CLI AWS, les kits SDK AWS et l'opération d'API `ListInsights` Amazon EKS correspondante. Consultez the section called "Afficher les informations sur le cluster" pour démarrer.

Afficher les informations sur le cluster

Amazon EKS fournit deux types d'informations : Informations sur la configuration et Informations sur la mise à niveau. Informations sur la configuration identifie les erreurs de configuration dans votre configuration de nœuds hybrides Amazon EKS qui pourraient nuire au fonctionnement de votre cluster ou de vos charges de travail. Informations sur la mise à niveau identifie les problèmes qui pourraient affecter votre capacité à mettre à niveau vers de nouvelles versions de Kubernetes.

Pour consulter la liste des vérifications d'informations effectuées et les éventuels problèmes pertinents identifiés par Amazon EKS, vous pouvez appeler le fonctionnement de look in the AWS Management Console, de la AWS CLI et de l'`ListInsights`API Amazon EKS. AWS SDKs

Afficher les informations de configuration (console)

1. Ouvrez la console Amazon EKS.
2. Dans la liste de clusters, choisissez le nom du cluster Amazon EKS pour lequel vous souhaitez obtenir des informations.
3. Sélectionnez Surveiller le cluster.
4. Sélectionnez l'onglet État du cluster.

5. Dans le tableau Informations de configuration, vous verrez les colonnes suivantes :
   - Nom : vérification effectuée par Amazon EKS par rapport au cluster.
   - Statut de l'information : une information dont le statut est `Error` signifie qu'il existe une erreur de configuration susceptible d'affecter le fonctionnement du cluster. Une information dont le statut est `Warning` signifie que la configuration ne correspond pas à l'approche documentée, mais que le cluster peut fonctionner si vous l'avez configuré intentionnellement. Une information dont le statut est `Passing` signifie qu'Amazon EKS n'a détecté aucun problème associé à ce contrôle de statut dans votre cluster.
   - Version : la version applicable.
   - Dernière heure d'actualisation : heure à laquelle le statut de l'information a été actualisé pour la dernière fois pour ce cluster.
   - Description : informations provenant de la vérification des informations, qui incluent l'alerte et les mesures correctives recommandées.

Afficher les informations de mise à niveau (console)

1. Ouvrez la console Amazon EKS.
2. Dans la liste de clusters, choisissez le nom du cluster Amazon EKS pour lequel vous souhaitez obtenir des informations.
3. Sélectionnez Surveiller le cluster.
4. Sélectionnez l'onglet Informations de mise à niveau.
5. Pour afficher les dernières données, cliquez sur le bouton Actualiser les informations et attendez que l'opération d'actualisation soit terminée.
6. Dans le tableau Informations de mise à niveau, vous verrez les colonnes suivantes :
   - Nom : vérification effectuée par Amazon EKS par rapport au cluster.
   - Statut de l'information : une information dont le statut est « Erreur » signifie généralement que la version Kubernetes concernée est N+1 de la version actuelle du cluster, tandis qu'un statut « Avertissement » signifie que l'information s'applique à une future version Kubernetes N+2 ou supérieure. Une information dont le statut est « Réussi » signifie qu'Amazon EKS n'a détecté aucun problème associé à cette vérification d'informations dans votre cluster. Une information dont le statut est « Inconnu » signifie qu'Amazon EKS n'est pas en mesure de déterminer si votre cluster est concerné par cette vérification des informations.
   - Version : la version Kubernetes pour laquelle l'information a vérifié les problèmes éventuels.

- Dernière heure d'actualisation : heure à laquelle le statut de l'information a été actualisé pour la dernière fois pour ce cluster.
- Dernière transition : heure à laquelle le statut de cette information a été modifié pour la dernière fois.
- Description : informations provenant de la vérification des informations, qui incluent l'alerte et les mesures correctives recommandées.

Afficher les informations sur le cluster (AWS CLI)

1. Pour afficher les dernières données, actualisez les informations pour un cluster spécifié. Apportez les modifications suivantes à la commande si nécessaire, puis exécutez la commande modifiée.
   - *region-code*Remplacez-le par le code de votre AWS région.
   - Remplacez *my-cluster* par le nom de votre cluster.

   ```
   aws eks start-insights-refresh --region region-code --cluster-name my-cluster
   ```

2. Pour suivre l'état d'actualisation des informations, exécutez la commande suivante. Remplacez *my-cluster* par le nom de votre cluster.

   ```
   aws eks describe-insights-refresh --cluster-name my-cluster
   ```

   L'exemple qui suit illustre un résultat.

   ```
   {
       "message": "Insights refresh is in progress",
       "status": "IN_PROGRESS",
       "startedAt": "2025-07-30T13:36:09-07:00"
   }
   ```

3. Répertoriez les informations pour un cluster spécifié. Apportez les modifications suivantes à la commande si nécessaire, puis exécutez la commande modifiée.
   - *region-code*Remplacez-le par le code de votre AWS région.
   - Remplacez *my-cluster* par le nom de votre cluster.

   ```
   aws eks list-insights --region region-code --cluster-name my-cluster
   ```

L'exemple qui suit illustre un résultat.


```
{
"insights":
    [
        {
            "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
            "name": "Deprecated APIs removed in Kubernetes vX.XX",
            "category": "UPGRADE_READINESS",
            "kubernetesVersion": "X.XX",
            "lastRefreshTime": 1734557315.000,
            "lastTransitionTime": 1734557309.000,
            "description": "Checks for usage of deprecated APIs that are scheduled
 for removal in Kubernetes vX.XX. Upgrading your cluster before migrating to the
 updated APIs supported by vX.XX could cause application impact.",
            "insightStatus":
            {
                "status": "PASSING",
                "reason": "No deprecated API usage detected within the last 30
 days.",
            },
        },
        {
            "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
            "name": "Kubelet version skew",
            "category": "UPGRADE_READINESS",
            "kubernetesVersion": "X.XX",
            "lastRefreshTime": 1734557309.000,
            "lastTransitionTime": 1734557309.000,
            "description": "Checks for kubelet versions of worker nodes in the
 cluster to see if upgrade would cause non compliance with supported Kubernetes
 kubelet version skew policy.",
            "insightStatus":
            {
                "status": "UNKNOWN",
                "reason": "Unable to determine status of node kubelet versions.",
            },
        },
        {
            "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
            "name": "Deprecated APIs removed in Kubernetes vX.XX",
```

```
            "category": "UPGRADE_READINESS",
            "kubernetesVersion": "X.XX",
            "lastRefreshTime": 1734557315.000,
            "lastTransitionTime": 1734557309.000,
            "description": "Checks for usage of deprecated APIs that are scheduled
for removal in Kubernetes vX.XX. Upgrading your cluster before migrating to the
updated APIs supported by vX.XX could cause application impact.",
            "insightStatus":
            {
                "status": "PASSING",
                "reason": "No deprecated API usage detected within the last 30
days.",
            },
        },
        {
            "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa",
            "name": "Cluster health issues",
            "category": "UPGRADE_READINESS",
            "kubernetesVersion": "X.XX",
            "lastRefreshTime": 1734557314.000,
            "lastTransitionTime": 1734557309.000,
            "description": "Checks for any cluster health issues that prevent
successful upgrade to the next Kubernetes version on EKS.",
            "insightStatus":
            {
                "status": "PASSING",
                "reason": "No cluster health issues detected.",
            },
        },
        {
            "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLEbbbbb",
            "name": "EKS add-on version compatibility",
            "category": "UPGRADE_READINESS",
            "kubernetesVersion": "X.XX",
            "lastRefreshTime": 1734557314.000,
            "lastTransitionTime": 1734557309.000,
            "description": "Checks version of installed EKS add-ons to ensure they
are compatible with the next version of Kubernetes. ",
            "insightStatus": { "status": "PASSING", "reason": "All installed EKS
add-on versions are compatible with next Kubernetes version."},
        },
        {
            "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLEccccc",
            "name": "kube-proxy version skew",
```

```
            "category": "UPGRADE_READINESS",
            "kubernetesVersion": "X.XX",
            "lastRefreshTime": 1734557314.000,
            "lastTransitionTime": 1734557309.000,
            "description": "Checks version of kube-proxy in cluster to see if
 upgrade would cause non compliance with supported Kubernetes kube-proxy version
 skew policy.",
            "insightStatus":
            {
                "status": "PASSING",
                "reason": "kube-proxy versions match the cluster control plane
 version.",
            },
        },
        {
            "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLEddddd",
            "name": "Deprecated APIs removed in Kubernetes vX.XX",
            "category": "UPGRADE_READINESS",
            "kubernetesVersion": "X.XX",
            "lastRefreshTime": 1734557315.000,
            "lastTransitionTime": 1734557309.000,
            "description": "Checks for usage of deprecated APIs that are scheduled
 for removal in Kubernetes vX.XX. Upgrading your cluster before migrating to the
 updated APIs supported by vX.XX could cause application impact.",
            "insightStatus":
            {
                "status": "PASSING",
                "reason": "No deprecated API usage detected within the last 30
 days.",
            },
        },
    ],
"nextToken": null,
}
```


4. Pour obtenir des informations descriptives sur une information, exécutez la commande suivante. Apportez les modifications suivantes à la commande si nécessaire, puis exécutez la commande modifiée.
   - *region-code*Remplacez-le par le code de votre AWS région.
   - *a1b2c3d4-5678-90ab-cdef-EXAMPLE22222*Remplacez-le par un ID d'aperçu extrait de la liste des informations du cluster.
   - Remplacez *my-cluster* par le nom de votre cluster.

```
aws eks describe-insight --region region-code --id a1b2c3d4-5678-90ab-cdef-
EXAMPLE22222 --cluster-name my-cluster
```

L'exemple qui suit illustre un résultat.


```
{
  "insight":
    {
      "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
      "name": "Kubelet version skew",
      "category": "UPGRADE_READINESS",
      "kubernetesVersion": "1.27",
      "lastRefreshTime": 1734557309.000,
      "lastTransitionTime": 1734557309.000,
      "description": "Checks for kubelet versions of worker nodes in the cluster
 to see if upgrade would cause non compliance with supported Kubernetes kubelet
 version skew policy.",
      "insightStatus":
        {
          "status": "UNKNOWN",
          "reason": "Unable to determine status of node kubelet versions.",
        },
      "recommendation": "Upgrade your worker nodes to match the Kubernetes version
 of your cluster control plane.",
      "additionalInfo":
        {
          "Kubelet version skew policy": "https://kubernetes.io/releases/version-
skew-policy/#kubelet",
          "Updating a managed node group": "https://docs.aws.amazon.com/eks/latest/
userguide/update-managed-node-group.html",
        },
      "resources": [],
      "categorySpecificSummary":
        { "deprecationDetails": [], "addonCompatibilityDetails": [] },
    },
}
```

Mettre à jour un cluster existant vers une nouvelle version de Kubernetes

> **Tip**
>
> Inscrivez-vous aux prochains ateliers Amazon EKS.

Dès qu'une nouvelle version Kubernetes est disponible dans Amazon EKS, vous pouvez mettre à jour votre cluster Amazon EKS.

> **Important**
>
> Une fois que vous avez mis à niveau un cluster, vous ne pouvez pas revenir à une version antérieure. Avant de mettre à jour vers une nouvelle version de Kubernetes, nous vous recommandons de consulter les informations dans Comprendre le cycle de vie des versions de Kubernetes sur EKS et les étapes de mise à niveau dans cette rubrique.

Les nouvelles versions de Kubernetes introduisent parfois des modifications importantes. Nous vous recommandons donc de comparer la version actuelle de vos applications à la toute nouvelle version Kubernetes avant de procéder à la mise à jour sur vos clusters de production. Pour ce faire, créez un flux d'intégration continue afin de tester le comportement de votre application avant de passer à une nouvelle version Kubernetes.

Dans le processus de mise à jour, Amazon EKS lance de nouveaux nœuds de serveur d'API avec la version de Kubernetes mise à jour pour remplacer les versions existantes. Amazon EKS effectue la surveillance de l'état standard de l'infrastructure et de la disponibilité pour le trafic réseau sur ces nouveaux nœuds afin de vérifier qu'ils fonctionnent comme prévu. Cependant, une fois que vous avez lancé la mise à niveau du cluster, vous ne pouvez plus la suspendre ni l'arrêter. Si l'une de ces vérifications échoue, Amazon EKS annule le déploiement d'infrastructure, et votre cluster reste dans la version Kubernetes précédente. Les applications en cours d'exécution ne sont pas affectées et votre cluster ne se retrouve jamais dans un état non déterministe ou irrécupérable. Amazon EKS sauvegarde régulièrement tous les clusters gérés, et des mécanismes existent pour récupérer des clusters si nécessaire. Nous évaluons et améliorons constamment nos processus de gestion de l'infrastructure Kubernetes.

Pour mettre à jour le cluster, Amazon EKS requiert jusqu'à cinq adresses IP disponibles correspondant aux sous-réseaux que vous avez spécifiés lors de la création de votre cluster. Amazon EKS crée de nouvelles interfaces réseau élastiques de cluster (interfaces réseau) dans l'un des sous-réseaux que vous avez spécifiés. Les interfaces réseau peuvent être créées dans des sous-réseaux différents de ceux dans lesquels se trouvent vos interfaces réseau existantes. Assurez-vous donc que les règles de votre groupe de sécurité autorisent la communication de cluster requise pour tous les sous-réseaux que vous avez spécifiés lors de la création de votre cluster. Si l'un des sous-réseaux que vous avez spécifiés lors de la création du cluster n'existe pas, ne dispose pas d'adresses IP disponibles en nombre suffisant ou ne dispose pas de règles de groupe de sécurité permettant la communication nécessaire au cluster, la mise à jour peut échouer.

Pour garantir que le point de terminaison du serveur API de votre cluster soit toujours accessible, Amazon EKS fournit un plan de contrôle Kubernetes hautement disponible et effectue des mises à jour propagées des instances du serveur API pendant les opérations de mise à jour. Afin de tenir compte des changements d'adresses IP des instances du serveur API prenant en charge votre point de terminaison du serveur API Kubernetes, vous devez vous assurer que vos clients du serveur API gèrent efficacement les reconnexions. Les versions récentes de `kubectl` et des bibliothèques clientes Kubernetes officiellement prises en charge effectuent ce processus de reconnexion de manière transparente.

> **Note**
>
> Pour en savoir plus sur le contenu d'une mise à jour de cluster, consultez Bonnes pratiques pour les mises à niveau de cluster dans le Guide des bonnes pratiques EKS. Cette ressource vous aide à planifier une mise à niveau et à comprendre la stratégie de mise à niveau d'un cluster.

Considérations relatives au mode automatique Amazon EKS

- La capacité de calcul du mode automatique Amazon EKS contrôle la version Kubernetes des nœuds. Après avoir mis à niveau le plan de contrôle, le mode automatique EKS commencera à mettre à jour progressivement les nœuds gérés. Le mode automatique EKS respecte les budgets de perturbation des pods.
- Vous n'avez pas besoin de mettre à niveau manuellement les capacités du mode automatique Amazon EKS, notamment les capacités d'autoscaling de calcul, de stockage en bloc et d'équilibrage de charge.

Résumé

Voici un résumé général du processus de mise à niveau du cluster Amazon EKS :Le résumé général du processus de mise à niveau du cluster Amazon EKS est le suivant :

1. Assurez-vous que votre cluster est dans un état permettant la mise à niveau. Cela inclut la vérification des Kubernetes APIs utilisés par les ressources déployées dans le cluster, afin de s'assurer que le cluster ne présente aucun problème de santé. Nous vous recommandons d'utiliser les informations de mise à niveau Amazon EKS pour évaluer l'état de préparation de votre cluster à la mise à niveau.
2. Mettez à niveau le plan de contrôle vers la version mineure suivante (par exemple, de 1.32 à 1.33).
3. Mettez à niveau les nœuds du plan de données pour qu'ils correspondent à ceux du plan de contrôle.
4. Mettez à niveau toutes les applications supplémentaires qui s'exécutent sur le cluster (par exemple, `cluster-autoscaler`).
5. Mettez à niveau les modules complémentaires fournis par Amazon EKS, tels que ceux inclus par défaut :
   - Version recommandée du CNI Amazon VPC
   - Version recommandée de CoreDNS
   - Version recommandée de `kube-proxy`
6. Mettez à niveau tous les clients qui communiquent avec le cluster (par exemple, `kubectl`).

Étape 1 : préparer la mise à niveau

Comparez la version Kubernetes de votre plan de contrôle de cluster à celle de vos nœuds.

- Obtenez la version Kubernetes du plan de contrôle de votre cluster.

```
kubectl version
```

- Obtenez la version Kubernetes de vos nœuds. Cette commande renvoie tous les nœuds Amazon EC2, Fargate et hybrides autogérés et gérés. Chaque pod Fargate est répertorié comme son propre nœud.

```
kubectl get nodes
```

Avant de mettre à jour votre plan de contrôle vers une nouvelle version de Kubernetes, assurez-vous que la version mineure de Kubernetes des nœuds gérés et des nœuds Fargate de votre cluster est identique à celle de votre plan de contrôle. Par exemple, si votre plan de contrôle exécute la version `1.29` et que l'un de vos nœuds exécute la version `1.28`, vous devez mettre à jour vos nœuds vers la version `1.29` avant de mettre à jour votre plan de contrôle vers la version 1.30. Nous vous recommandons également de mettre à jour vos nœuds autogérés et hybrides vers la même version que votre plan de contrôle avant de mettre à jour ce dernier. Pour plus d'informations, consultez the section called "Mettre à jour", the section called "Méthodes de mise à jour" et the section called "Mettre à niveau les nœuds hybrides". Si vous disposez de nœuds Fargate dont la version mineure est inférieure à celle du plan de contrôle, veuillez d'abord supprimer le pod représenté par le nœud. Mettez ensuite à jour votre plan de contrôle. Les pods restants seront mis à niveau vers la nouvelle version après leur redéploiement.

Étape 2 : examiner les considérations relatives à la mise à niveau

Amazon EKS Cluster Insights analyse automatiquement les clusters par rapport à une liste de problèmes potentiels liés à la mise à niveau de la version Kubernetes, tels que l'utilisation d'API Kubernetes obsolètes. Amazon EKS met régulièrement à jour la liste des vérifications à effectuer en fonction des évaluations des modifications apportées au projet Kubernetes. Amazon EKS met également à jour la liste des vérifications à mesure que des modifications sont introduites dans le service Amazon EKS avec les nouvelles versions. Pour de plus amples informations, veuillez consulter the section called "Informations sur le cluster".

Veuillez consulter le Guide de migration des API obsolètes dans la documentation Kubernetes.

Vérifiez les informations relatives à la mise à niveau

Utilisez les informations relatives à la mise à niveau d'Amazon EKS pour identifier les problèmes. Pour de plus amples informations, veuillez consulter the section called "Afficher les informations de mise à niveau (console)".

Considérations détaillées

- Dans la mesure où Amazon EKS exécute un plan de contrôle hautement disponible, vous devez mettre à jour une seule version mineure à la fois. Pour plus d'informations sur cette exigence, consultez Kubernetes Version and Version Skew Support Policy (Politique de prise en charge des versions et versions asymétriques de Kubernetes). Supposez que la version actuelle de votre cluster soit la version `1.28` et que vous souhaitiez la mettre à jour vers la version `1.30`. Vous

devez d'abord mettre à jour la version `1.28` de votre cluster vers la version `1.29`, puis mettre à jour la version `1.29` de votre cluster vers la version `1.30`.

- Vérifiez le décalage de version entre le `kube-apiserver` Kubernetes et le `kubelet` sur vos nœuds.
  - À partir de la version `1.28` de Kubernetes, le `kubelet` peut être jusqu'à trois versions mineures plus anciennes que le `kube-apiserver`. Voir la politique d'asymétrie des versions en amont de Kubernetes.
  - Si le `kubelet` de vos nœuds gérés et Fargate est en version Kubernetes `1.25` ou supérieure, vous pouvez mettre à jour votre cluster jusqu'à trois versions sans mettre à jour la version du `kubelet`. Par exemple, si la version du `kubelet` est `1.25`, vous pouvez mettre à jour la version de votre cluster Amazon EKS de `1.25` vers `1.26` ou vers `1.27` et vers `1.28` alors que le `kubelet` reste sur la version `1.25`.
- Avant de commencer une mise à jour, il est recommandé de vous assurer que le `kubelet` de vos nœuds est à la même version Kubernetes que votre plan de contrôle.
- Si votre cluster est configuré avec une version du plug-in CNI Amazon VPC pour Kubernetes antérieure à `1.8.0`, nous vous recommandons de mettre à jour le plug-in vers la dernière version avant de mettre à jour votre cluster. Pour mettre à jour le plugin, consultez the section called "CNI Amazon VPC".
- Vous pouvez effectuer une sauvegarde de votre cluster Amazon EKS afin de restaurer l'état de votre cluster Amazon EKS et son stockage persistant en cas de défaillance lors du processus de mise à niveau. Consultez the section called " AWS Backup"

Étape 3 : mettre à jour le plan de contrôle du cluster

> **Important**
>
> Amazon EKS a temporairement supprimé une fonctionnalité qui vous obligeait à utiliser un indicateur `--force` pour mettre à niveau votre cluster en cas de problèmes liés aux informations sur le cluster. Pour plus d'informations, voir Annulation temporaire de l'application des informations de mise à niveau lors de la mise à jour de la version du cluster sur. GitHub
> Amazon EKS actualise les informations sur le cluster 24 h après la « dernière heure d'actualisation ». Vous pouvez comparer l'heure à laquelle vous avez résolu un problème à la « dernière heure d'actualisation » des informations sur le cluster.

De plus, la mise à jour du statut des informations peut prendre jusqu'à 30 jours après la résolution d'un problème d'utilisation d'une API obsolète. Mettre à niveau les informations recherche toujours les utilisations d'API obsolètes sur une période de 30 jours consécutifs.

Vous pouvez soumettre la demande de mettre à niveau la version de votre plan de contrôle EKS à l'aide de :

- eksctl
- la AWS console
- la AWS CLI

Mettre à jour le cluster – eksctl

Cette procédure nécessite `eksctl` version `0.215.0` ou ultérieure. Vous pouvez vérifier votre version avec la commande suivante :

```
eksctl version
```

Pour les instructions d'installation et de mise à jour de `eksctl`, consultez la rubrique Installation dans la documentation `eksctl`.

Mettez à jour la version Kubernetes de votre plan de contrôle Amazon EKS. Remplacez `<cluster-name>` par le nom de votre cluster. Remplacez `<version-number>` par le numéro de version pris en charge par Amazon EKS vers lequel vous voulez mettre à niveau votre cluster. Pour obtenir la liste des numéros de version pris en charge, consultez Versions prises en charge par Amazon EKS.

```
eksctl upgrade cluster --name <cluster-name> --version <version-number> --approve
```

Cette mise à jour peut prendre plusieurs minutes.

Passez au the section called "Étape 4 : mettre à jour les composants du cluster".

Cluster de mise à jour - AWS console

1. Ouvrez la console Amazon EKS.
2. Sélectionnez Mettre à niveau maintenant pour le cluster que vous souhaitez mettre à niveau.

3. Sélectionnez la version vers laquelle vous souhaitez mettre à jour votre cluster, puis choisissez Mettre à niveau.
4. Cette mise à jour peut prendre plusieurs minutes. Passez au the section called "Étape 4 : mettre à jour les composants du cluster".

Mettre à jour le cluster - AWS CLI

1. Vérifiez que la AWS CLI est installée et que vous êtes connecté. Pour plus d'informations, voir Installation ou mise à jour vers la dernière version de la AWS CLI.
2. Mettez à jour votre cluster Amazon EKS à l'aide de la commande AWS CLI suivante. Remplacez `<cluster-name>` et `<region-code>` par le nom et le code de région du cluster que vous voulez mettre à niveau. Remplacez `<version-number>` par le numéro de version pris en charge par Amazon EKS vers lequel vous voulez mettre à niveau votre cluster. Pour obtenir la liste des numéros de version pris en charge, consultez Versions prises en charge par Amazon EKS.

```
aws eks update-cluster-version --name <cluster-name> \
  --kubernetes-version <verion-number> --region <region-code>
```

L'exemple qui suit illustre un résultat.

```
{
    "update": {
        "id": "<update-id>",
        "status": "InProgress",
        "type": "VersionUpdate",
        "params": [
            {
                "type": "Version",
                "value": "<version-number>"
            },
            {
                "type": "PlatformVersion",
                "value": "eks.1"
            }
        ],
[...]
        "errors": []
    }
```

3. Cette mise à jour peut prendre plusieurs minutes. Contrôlez l'état de mise à jour de votre cluster avec la commande suivante. En plus d'utiliser le même `<cluster-name>` et `<region-code>`, utilisez l'`<update-id>` renvoyé par la commande précédente.

```
aws eks describe-update --name <cluster-name> \
   --region <region-code> --update-id <update-id>
```

   Lorsqu'un état `Successful` s'affiche, la mise à jour est terminée.

4. Passez au the section called "Étape 4 : mettre à jour les composants du cluster".

Étape 4 : mettre à jour les composants du cluster

1. Une fois la mise à jour de votre cluster terminée, mettez à jour vos nœuds vers la même version Kubernetes que celle de votre cluster mis à jour. Pour plus d'informations, consultez the section called "Méthodes de mise à jour", the section called "Mettre à jour" et the section called "Mettre à niveau les nœuds hybrides". Tous les nouveaux pods lancés sur Fargate ont une version `kubelet` qui correspond à la version de votre cluster. Les pods Fargate existants ne sont pas modifiés.
2. (Facultatif) Si vous avez déployé le composant Kubernetes Cluster Autoscaler sur votre cluster avant de mettre à jour le cluster, mettez ce composant à jour vers la version la plus récente correspondant à la version majeure et mineure de Kubernetes vers laquelle vous avez effectué la mise à jour.
   a. Ouvrez la page versions de Cluster Autoscaler dans un navigateur Web et recherchez la dernière version de Cluster Autoscaler qui correspond à la version majeure et mineure de Kubernetes de votre cluster. Par exemple, si la version Kubernetes de votre cluster est `1.30`, recherchez la dernière version publiée de Cluster Autoscaler qui commence par `1.30`. Enregistrez le numéro de version sémantique (`1.30.n`, par exemple) de cette version à utiliser à l'étape suivante.
   b. Identifiez l'image Cluster Autoscaler sur la version que vous avez enregistrée à l'étape précédente avec la commande suivante. Si nécessaire, remplacez `X.XX.X` par votre propre valeur.

```
kubectl -n kube-system set image deployment.apps/cluster-autoscaler cluster-
autoscaler=registry.k8s.io/autoscaling/cluster-autoscaler:vX.XX.X
```

3. (Clusters dotés de nœuds GPU uniquement) Si votre cluster possède des groupes de nœuds compatibles avec le GPU (par exemple,`p3.2xlarge`), vous devez mettre à jour le plug-in d'appareil NVIDIA pour Kubernetes DaemonSet sur votre cluster. `<vX.X.X>`Remplacez-le par la s-device-plugin version Nvidia/K8 de votre choix avant d'exécuter la commande suivante.

   ```
   kubectl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/<vX.X.X>/
   deployments/static/nvidia-device-plugin.yml
   ```

4. Mettez à jour le plug-in CNI Amazon VPC pour Kubernetes, CoreDNS et les modules complémentaires `kube-proxy`. Nous vous recommandons de mettre à jour les modules complémentaires avec les versions minimales répertoriées dans les jetons de compte de service.
   - Si vous utilisez des modules complémentaires Amazon EKS, dans la console Amazon EKS, sélectionnez Clusters, puis le nom du cluster que vous avez mis à jour dans le volet gauche. Les notifications s'affichent dans la console. Elles vous informent qu'une nouvelle version est disponible pour chaque module complémentaire disposant d'une mise à jour disponible. Pour mettre à jour un module complémentaire, sélectionnez l'onglet Add-ons (Modules complémentaires). Dans l'une des zones d'un module complémentaire dont une mise à jour est disponible, sélectionnez Update now (Mettre à jour maintenant), sélectionnez une version disponible, puis cliquez sur Update (Mise à jour).
   - Vous pouvez également utiliser la AWS CLI ou mettre `eksctl` à jour des modules complémentaires. Pour de plus amples informations, veuillez consulter the section called “Mettre à jour un module complémentaire”.
5. Si nécessaire, mettez à jour votre version de `kubectl`. Vous devez utiliser une version `kubectl` qui se situe à une différence de version mineure près du plan de contrôle de votre cluster Amazon EKS.

Réduire la version Kubernetes d'un cluster Amazon EKS

Il n'est pas possible de réduire la version Kubernetes d'un cluster Amazon EKS. À la place, créez un nouveau cluster sur une version Amazon EKS précédente et migrer les charges de travail.

Supprimer un cluster

Lorsque vous avez terminé d'utiliser un cluster Amazon EKS, vous devez supprimer les ressources qui lui sont associées afin de ne pas entraîner de coûts superflus.

Vous pouvez supprimer un cluster à l'aide `eksctl` de la AWS Management Console CLI ou de la AWS CLI.

Considérations

- Si vous recevez une erreur en raison de la suppression du créateur du cluster, reportez-vous à cet article pour la résoudre.
- Les ressources du service géré Amazon pour Prometheus ne font pas partie du cycle de vie du cluster et doivent être gérées indépendamment de celui-ci. Lorsque vous supprimez votre cluster, veillez à supprimer également tous les scrapers concernés afin de mettre fin aux coûts applicables. Pour plus d'informations, consultez Rechercher et supprimer des scrapers dans le Guide de l'utilisateur du Service géré Amazon pour Prometheus.
- Pour supprimer un cluster connecté, reportez-vous à la section the section called "Annuler l'enregistrement d'un cluster"
- Avant de supprimer un cluster, assurez-vous que la protection contre la suppression est désactivée pour votre cluster.

Considérations relatives au mode automatique EKS

- Tous les nœuds du mode automatique EKS seront supprimés, y compris les instances gérées EC2.
- Tous les équilibreurs de charge seront supprimés.

Pour de plus amples informations, veuillez consulter the section called "Désactivation du mode automatique EKS".

Etapes prérequises

Vous devez d'abord effectuer les étapes suivantes avant de pouvoir supprimer un cluster. Ces étapes s'appliquent quelle que soit la méthode que vous utilisez pour supprimer votre cluster.

1. Répertoriez tous les services qui s'exécutent dans votre cluster.

```
kubectl get svc --all-namespaces
```

2. Supprimez tous les services qui ont une valeur `EXTERNAL-IP` associée. Ces services ont un équilibreur de charge Elastic Load Balancing en avant-plan, et vous devez les supprimer dans

Kubernetes pour que l'équilibreur de charge et les ressources associées soient correctement libérés. Remplacez *service-name* par le nom de chaque service répertorié comme décrit.

```
kubectl delete svc service-name
```

3. Supprimez également toutes les ressources d'entrée. Si vous ne supprimez pas les ressources d'entrée, l'équilibreur de charge de l'application est conservé même si vous avez supprimé le cluster. *ingress-name*Remplacez-le par le nom de vos ressources d'entrée.

```
kubectl get ingress --all-namespaces
```

```
kubectl delete ing ingress-name
```

Supprimer le cluster (eksctl)

Cette procédure nécessite `eksctl` version `0.215.0` ou ultérieure. Vous pouvez vérifier votre version avec la commande suivante :

```
eksctl version
```

Pour les instructions d'installation ou de mise à niveau de `eksctl`, consultez la rubrique Installation dans la documentation `eksctl`.

1. Suivez les étapes préalables. Ensuite, supprimez votre cluster et les nœuds associés à l'aide de la commande suivante, en les *prod* remplaçant par le nom de votre cluster.

```
eksctl delete cluster --name prod
```

Sortie :

```
[#]  using region region-code
[#]  deleting EKS cluster "prod"
[#]  will delete stack "eksctl-prod-nodegroup-standard-nodes"
[#]  waiting for stack "eksctl-prod-nodegroup-standard-nodes" to get deleted
[#]  will delete stack "eksctl-prod-cluster"
[#]  the following EKS cluster resource(s) for "prod" will be deleted: cluster. If in
 doubt, check CloudFormation console
```

Supprimer le cluster (AWS console)

1. Suivez les étapes préalables. Ensuite, supprimez tous les groupes de nœuds et les profils Fargate.
   a. Ouvrez la console Amazon EKS.
   b. Dans le panneau de navigation de gauche, sélectionnez Clusters Amazon EKS, puis dans la liste à onglets des clusters, sélectionnez le nom du cluster que vous voulez supprimer.
   c. Sélectionnez l'onglet Calcul et choisissez un groupe de nœuds à supprimer. Sélectionnez Supprimer, saisissez le nom du groupe de nœuds, puis sélectionnez Supprimer. Supprimez tous les groupes de nœuds du cluster.

      > **Note**
      >
      > Tous les groupes de nœuds répertoriés sont des groupes de nœuds gérés.

   d. Sélectionnez un profil Fargate à supprimer, choisissez Supprimer, saisissez le nom du profil, puis sélectionnez Supprimer. Supprimez tous les profils Fargate dans le cluster.
2. Supprimez toutes les piles de nœuds AWS CloudFormation autogérées.
   a. Ouvrez la AWS CloudFormation console.
   b. Sélectionnez la pile du nœud à supprimer, puis choisissez Supprimer.
   c. Dans la boîte de dialogue de confirmation Delete stack (Supprimer la pile), choisissez Delete stack (Supprimer la pile). Supprimez toutes les piles de nœuds autogérées dans le cluster.
3. Supprimez le cluster.
   a. Ouvrez la console Amazon EKS.
   b. Sélectionnez le cluster à supprimer, puis choisissez Supprimer.
   c. Dans l'écran de confirmation de suppression du cluster, choisissez Delete (Supprimer).
4. (Facultatif) Supprimez la pile VPC. AWS CloudFormation
   a. Ouvrez la AWS CloudFormation console.
   b. Sélectionnez la pile VPC à supprimer, puis choisissez Delete (Supprimer).
   c. Dans la boîte de dialogue de confirmation Delete stack (Supprimer la pile), choisissez Delete stack (Supprimer la pile).

Supprimer le cluster (AWS CLI)

1. Suivez les étapes préalables. Ensuite, supprimez tous les groupes de nœuds et les profils Fargate.

a. Répertoriez les groupes de nœuds de votre cluster à l'aide de la commande suivante.

```
aws eks list-nodegroups --cluster-name my-cluster
```

> **Note**
>
> Tous les groupes de nœuds répertoriés sont des groupes de nœuds gérés.

b. Supprimez chaque groupe de nœuds à l'aide de la commande suivante. Supprimez tous les groupes de nœuds du cluster.

```
aws eks delete-nodegroup --nodegroup-name my-nodegroup --cluster-name my-cluster
```

c. Répétez les profils Fargate dans votre cluster à l'aide de la commande suivante.

```
aws eks list-fargate-profiles --cluster-name my-cluster
```

d. Supprimez chaque profil Fargate avec la commande suivante. Supprimez tous les profils Fargate dans le cluster.

```
aws eks delete-fargate-profile --fargate-profile-name my-fargate-profile --
cluster-name my-cluster
```

2. Supprimez toutes les piles de nœuds AWS CloudFormation autogérées.

a. Répertoriez vos AWS CloudFormation piles disponibles à l'aide de la commande suivante. Identifiez le nom du modèle du nœud dans la sortie obtenue.

```
aws cloudformation list-stacks --query "StackSummaries[].StackName"
```

b. Supprimez chaque pile de nœud avec la commande suivante, en remplaçant *node-stack* par le nom de votre pile de nœuds. Supprimez toutes les piles de nœuds autogérées dans le cluster.

```
aws cloudformation delete-stack --stack-name node-stack
```

3. Supprimez le cluster avec la commande suivante, en remplaçant *my-cluster* par le nom de votre cluster.

```
aws eks delete-cluster --name my-cluster
```

4. (Facultatif) Supprimez la pile VPC. AWS CloudFormation

   a. Répertoriez vos AWS CloudFormation piles disponibles à l'aide de la commande suivante. Recherchez le nom du modèle VPC dans la sortie obtenue.

   ```
   aws cloudformation list-stacks --query "StackSummaries[].StackName"
   ```

   b. Supprimez la pile VPC avec la commande suivante, en remplaçant *my-vpc-stack* par le nom de votre pile VPC.

   ```
   aws cloudformation delete-stack --stack-name my-vpc-stack
   ```

Protection des clusters EKS contre toute suppression accidentelle

La suppression accidentelle d'un cluster EKS peut nuire au fonctionnement du cluster Kubernetes.

Vous pouvez désormais protéger les clusters EKS contre toute suppression accidentelle. Si vous activez la protection contre la suppression sur un cluster, vous devez d'abord désactiver cette protection avant de pouvoir supprimer le cluster.

Le but de la protection contre les suppressions est de prévenir les accidents. Vous devez limiter avec soin les personnes autorisées à supprimer des clusters.

Si vous essayez de supprimer un cluster actif dont la protection contre la suppression est activée, vous recevrez un `InvalidRequestException`.

Important

Si vous activez la protection contre la suppression sur un cluster, vous devez disposer des autorisations IAM UpdateClusterConfig et des autorisations DeleteCluster IAM pour d'abord supprimer la protection contre la suppression, puis supprimer le cluster.

Note

Si l'état du cluster est en cours de création, a échoué ou est en cours de suppression, vous pouvez supprimer le cluster même si la protection contre la suppression est activée.

Pour activer la protection contre la suppression d'un cluster existant

Vous ne pouvez exécuter cette opération que sur un cluster en état actif.

```
aws eks update-cluster-config --name <cluster-name> --region <aws-region> --deletion-protection
```

Pour désactiver la protection contre la suppression d'un cluster existant

```
aws eks update-cluster-config --name <cluster-name> --region <aws-region> --no-deletion-protection
```

Point de terminaison du serveur d'API du cluster

Cette rubrique vous aide à activer l'accès privé pour le point de terminaison de serveur d'API Kubernetes de votre cluster Amazon EKS et à désactiver complètement l'accès public afin que celui-ci ne soit pas accessible depuis Internet.

Lorsque vous créez un cluster, Amazon EKS crée un point de terminaison pour le serveur d'API Kubernetes géré que vous utilisez pour communiquer avec votre cluster (à l'aide d'outils de gestion Kubernetes comme `kubectl`). Par défaut, ce point de terminaison du serveur d'API est public sur Internet, et l'accès au serveur d'API est sécurisé à l'aide d'une combinaison d' AWS Identity and Access Management (IAM) et de contrôle d'accès basé sur le rôle (RBAC) Kubernetes natif. Ce point de terminaison est connu sous le nom de point de terminaison public du cluster. Il existe également un point de terminaison privé du cluster. Pour plus d'informations sur le point de terminaison privé du cluster, consultez la section the section called "Point de terminaison privé du cluster" suivante.

Format du point de terminaison du cluster `IPv6`

EKS crée un point de terminaison à double pile unique selon le format ci-dessous pour tout nouveau cluster `IPv6` créé après octobre 2024. Un IPv6 cluster est un cluster que vous sélectionnez `IPv6` dans le paramètre IP family (`ipFamily`) du cluster.

Example

AWS

Point de public/private terminaison du cluster EKS : `eks-cluster.region.api.aws`

AWS GovCloud (US)

Point de public/private terminaison du cluster EKS : `eks-cluster.region.api.aws`

Amazon Web Services in China

Point de public/private terminaison du cluster EKS : `eks-cluster.region.api.amazonwebservices.com.cn`

> **Note**
>
> Le point de terminaison de cluster à double pile a été introduit en octobre 2024. Pour plus d'informations sur les clusters `IPv6`, consultez the section called "IPv6". Les clusters créés avant octobre 2024 utilisent plutôt le format de point de terminaison suivant.

Format du point de terminaison du cluster **`IPv4`**

EKS crée un point de terminaison unique au format suivant pour chaque cluster avec `IPv4` sélectionné dans le paramètre de famille d'adresses IP (ipFamily) du cluster :

Example

AWS

public/private Point final du cluster EKS `eks-cluster.region.eks.amazonaws.com`

AWS GovCloud (US)

public/private Point final du cluster EKS `eks-cluster.region.eks.amazonaws.com`

Amazon Web Services in China

public/private Point final du cluster EKS `eks-cluster.region.amazonwebservices.com.cn`

> **Note**
>
> Avant octobre 2024, les clusters `IPv6` utilisaient également ce format de point de terminaison. Pour ces clusters, les points de terminaison public et privé ne résolvaient que des adresses `IPv4`.

Point de terminaison privé du cluster

Vous pouvez activer l'accès privé au serveur d'API Kubernetes pour que toutes les communications entre vos nœuds et le serveur d'API restent au sein de votre VPC. Vous pouvez limiter les adresses IP qui peuvent accéder à votre serveur API à partir d'Internet, ou désactiver complètement l'accès Internet au serveur d'API.

> **Note**
>
> Comme ce point de terminaison est destiné au serveur d'API Kubernetes et n'est pas un point de AWS PrivateLink terminaison traditionnel pour communiquer avec une AWS API, il n'apparaît pas en tant que point de terminaison dans la console Amazon VPC.

Lorsque vous activez l'accès privé au point de terminaison de votre cluster, Amazon EKS crée automatiquement une zone hébergée privée Route 53 et l'associe au VPC de votre cluster. Cette zone hébergée privée est gérée par Amazon EKS et n'apparaît pas dans vos ressources Route 53. Pour que la zone hébergée privée achemine correctement le trafic vers votre serveur d'API, votre VPC doit avoir `enableDnsHostnames` et `enableDnsSupport` définis sur `true`, et les options DHCP définies pour votre VPC doivent inclure `AmazonProvidedDNS` dans leur liste de serveurs de nom de domaine. Pour plus d'informations, consultez Mise à jour du support DNS pour votre VPC dans le Guide de l'utilisateur d'Amazon VPC.

Vous pouvez définir les exigences d'accès au point de terminaison de votre serveur d'API lorsque vous créez un nouveau cluster, et vous pouvez mettre à jour l'accès au point de terminaison du serveur d'API pour un cluster à tout moment.

Modification de l'accès au point de terminaison de cluster

Utilisez les procédures de cette section afin de modifier l'accès au point de terminaison pour un cluster existant. Le tableau suivant présente les combinaisons d'accès au point de terminaison de serveur d'API prises en charge et leur comportement.

Accès public au point de terminaison	Accès privé au point de terminaison	Attitude
Activé	Désactivé	• Il s'agit du comportement par défaut pour les nouveaux clusters Amazon EKS. • Les requêtes vers l'API Kubernetes qui proviennent de l'intérieur du VPC de votre cluster (par exemple, les communications nœud vers plan de contrôle) sortent du VPC mais restent sur le réseau Amazon. • Le serveur d'API de votre cluster est accessible depuis internet. Vous pouvez éventuellement utiliser la liste des accès publics CIDRs pour contrôler l'accès au point de terminaison public à l'aide d'une liste de blocs CIDR. Si vous limitez l'accès à des blocs CIDR spécifiques, nous vous recommandons d'activer également le point de terminaison privé ou de vous assurer que les blocs CIDR que vous spécifiez incluent les adresses à partir desquelles les nœuds et les pods Fargate (si vous les utilisez) accèdent au point de terminaison public.
Activé	Activé	• Les requêtes vers l'API Kubernetes provenant du VPC de votre cluster (par exemple, les communica tions nœud vers plan de contrôle) utilisent le point de terminaison d'un VPC privé. Vous utilisez les règles du groupe de sécurité du cluster pour contrôler l'accès au point de terminaison privé. • Le serveur d'API de votre cluster est accessible depuis internet. Vous pouvez éventuellement utiliser la liste des accès publics CIDRs pour contrôler l'accès au point de terminaison public à l'aide d'une liste de blocs CIDR. • Si vous utilisez des nœuds hybrides avec votre cluster Amazon EKS, il n'est pas recommandé d'activer à la fois l'accès public et privé aux points de terminais on. Comme les nœuds hybrides fonctionnent en dehors de votre VPC, ils résolvent le point de terminais

Accès public au point de terminaison	Accès privé au point de terminaison	Attitude
		on du cluster vers les adresses IP publiques. Il est recommandé d'utiliser un accès aux points de terminais on public ou privé pour les clusters dotés de nœuds hybrides.
Désactivé	Activé	• Tout le trafic vers le serveur d'API du cluster doit provenir de votre VPC ou d'un réseau connecté. • Il n'y a pas d'accès public vers votre serveur d'API depuis Internet. Toutes les commandes `kubectl` doivent provenir du VPC ou d'un réseau connecté. Pour les options de connectivité, consultez the section called "Accès à un serveur d'API privé uniquement". • Vous utilisez les règles du groupe de sécurité du cluster pour contrôler l'accès au point de terminaison privé. Notez que l'accès public n'affecte CIDRs pas le point de terminaison privé. • Le point de terminaison du serveur d'API du cluster est résolu par les serveurs DNS publics à une adresse IP privée à partir du VPC. Dans le passé, le point de terminaison ne pouvait être résolu qu'à partir du VPC. Si votre point de terminaison ne se résout pas à une adresse IP privée dans le VPC pour un cluster existant, vous pouvez : • Activer l'accès public, puis le désactiver à nouveau. Vous n'avez besoin de le faire qu'une seule fois pour un cluster et le point de terminaison se résoudra en une adresse IP privée à partir de ce point. • Mettez à jour votre cluster.

Contrôles d'accès aux terminaux

Notez que chacune des méthodes suivantes pour contrôler l'accès au point de terminaison n'affecte que le point de terminaison concerné.

Groupe de sécurité du cluster

Le groupe de sécurité du cluster contrôle deux types de connexions : les connexions à l'API kubelet et le point de terminaison privé. Les connexions à l'`kubelet`API sont utilisées dans les `kubectl port-forward` commandes `kubectl attach` `kubectl cp`kubectl `exec`,`kubectl logs`,, et. Le groupe de sécurité du cluster n'affecte pas le point de terminaison public.

Accès public CIDRs

L'accès public CIDRs contrôle l'accès au point de terminaison public par une liste de blocs CIDR. Notez que l'accès public n'affecte CIDRs pas le point de terminaison privé. L'accès public CIDRs se comporte différemment sur les `IPv6` `IPv4` clusters et les clusters en fonction de leur date de création, décrite ci-dessous :

Blocs CIDR dans le point de terminaison public (cluster **`IPv6`**)

Vous pouvez ajouter des blocs CIDR `IPv6` et `IPv4` au point de terminaison public d'un cluster `IPv6`, car le point de terminaison public est double pile. Cette option ne s'applique qu'aux nouveaux clusters dont la `ipFamily` est définie sur `IPv6` et qui sont créés en octobre 2024 ou après. Vous pouvez identifier ces clusters grâce à leur nouveau nom de domaine de point de terminaison `api.aws`.

Blocs CIDR dans le point de terminaison public (cluster **`IPv4`**)

Vous pouvez ajouter des blocs CIDR `IPv4` au point de terminaison public d'un cluster `IPv4`. Vous ne pouvez pas ajouter des blocs CIDR `IPv6` au point de terminaison public d'un cluster `IPv4`. Si vous essayez, EKS renvoie le message d'erreur suivant : `The following CIDRs are invalid in publicAccessCidrs`

Blocs CIDR dans le point de terminaison public (cluster **`IPv6`** créé avant octobre 2024)

Vous pouvez ajouter des blocs CIDR `IPv4` au point de terminaison public des anciens clusters `IPv6` que vous avez créés avant octobre 2024. Vous pouvez identifier ces clusters grâce au point de terminaison `eks.amazonaws.com`. Vous ne pouvez pas ajouter des blocs CIDR `IPv6` au point de terminaison public de ces anciens clusters `IPv6` que vous avez créés avant octobre 2024. Si vous essayez, EKS renvoie le message d'erreur suivant : `The following CIDRs are invalid in publicAccessCidrs`

Accès à un serveur d'API privé uniquement

Si vous avez désactivé l'accès public au point de terminaison du serveur d'API Kubernetes de votre cluster, vous ne pouvez accéder à l'API serveur que depuis votre VPC ou un réseau connecté. Voici quelques méthodes d'accès possibles au point de terminaison du serveur d'API Kubernetes :

Réseau connecté

Connectez votre réseau au VPC à l'aide d'une passerelle de transit AWS ou d'une autre option de connectivité, puis utilisez un ordinateur dans le réseau connecté. Vous devez vous assurer que le groupe de sécurité de votre plan de contrôle Amazon EKS contient des règles pour autoriser le trafic entrant sur le port 443 depuis votre réseau connecté.

Hôte Amazon EC2 Bastion

Vous pouvez lancer une EC2 instance Amazon dans un sous-réseau public du VPC de votre cluster, puis vous connecter via SSH à cette instance pour exécuter des commandes. `kubectl` Pour plus d'informations, consultez la section Hôtes bastions Linux sur AWS. Vous devez vous assurer que le groupe de sécurité de votre plan de contrôle Amazon EKS contient des règles pour autoriser le trafic entrant sur le port 443 depuis l'hôte bastion. Pour de plus amples informations, veuillez consulter the section called “Exigences de groupe de sécurité”.

Lorsque vous configurez `kubectl` votre hôte bastion, veillez à utiliser des AWS informations d'identification déjà mappées à la configuration RBAC de votre cluster, ou ajoutez le principal IAM que votre bastion utilisera à la configuration RBAC avant de supprimer l'accès public du point de terminaison. Pour plus d'informations, consultez the section called “Accès à l'API Kubernetes” et the section called “Accès non autorisé ou refusé (`kubectl`)”.

Environnement de développement intégré AWS Cloud9

AWS Cloud9 est un environnement de développement intégré (IDE) basé sur le cloud qui vous permet d'écrire, d'exécuter et de déboguer votre code avec un simple navigateur. Vous pouvez créer un IDE AWS Cloud9 dans le VPC de votre cluster et utiliser l'IDE pour communiquer avec votre cluster. Pour plus d'informations, consultez Création d'un environnement dans AWS Cloud9. Vous devez vous assurer que votre groupe de sécurité de plan de contrôle Amazon EKS contient des règles permettant d'autoriser le trafic entrant sur le port 443 à partir de votre groupe de sécurité IDE. Pour de plus amples informations, veuillez consulter the section called “Exigences de groupe de sécurité”.

Lorsque vous configurez `kubectl` votre IDE AWS Cloud9, veillez à utiliser des AWS informations d'identification déjà mappées à la configuration RBAC de votre cluster, ou ajoutez le principal IAM

que votre IDE utilisera à la configuration RBAC avant de supprimer l'accès public des terminaux. Pour plus d'informations, consultez the section called "Accès à l'API Kubernetes" et the section called "Accès non autorisé ou refusé (`kubectl`)".

Modifiez cette page sur GitHub

Configuration de l'accès réseau au point de terminaison du serveur d'API du cluster

Vous pouvez modifier l'accès aux points de terminaison de votre serveur API de cluster à l'aide de la AWS CLI AWS Management Console ou dans les sections suivantes.

Configuration de l'accès aux points de terminaison - AWS console

1. Ouvrez la console Amazon EKS.
2. Choisissez le nom du cluster pour afficher les informations le concernant.
3. Sélectionnez l'onglet Réseau, puis Gérer l'accès au point de terminaison.
4. Pour Accès privé, choisissez d'activer ou de désactiver l'accès privé au point de terminaison du serveur d'API Kubernetes de votre cluster. Si vous activez l'accès privé, les requêtes d'API Kubernetes provenant de l'intérieur du VPC de votre cluster utilisent le point de terminaison d'un VPC privé. Vous devez activer l'accès privé pour désactiver l'accès public.
5. Pour Accès public, choisissez d'activer ou de désactiver l'accès public au point de terminaison du serveur d'API Kubernetes de votre cluster. Si vous désactivez l'accès public, le serveur d'API Kubernetes de votre cluster peut uniquement recevoir des requêtes provenant de l'intérieur du VPC du cluster.
6. (Facultatif) Si vous avez activé l'accès public, vous pouvez spécifier les adresses IP Internet autorisées à communiquer avec le point de terminaison public. Sélectionnez Advanced Settings (Paramètres avancés). Entrez un bloc CIDR, tel que *`203.0.113.5/32`*. Le bloc ne peut pas inclure d'adresses réservées. Vous pouvez entrer des blocs supplémentaires en sélectionnant Add Source (Ajouter une source). Vous pouvez spécifier un nombre maximal de blocs CIDR. Pour de plus amples informations, veuillez consulter the section called "Quotas de service". Si vous ne spécifiez aucun bloc CIDR, le point de terminaison public du serveur d'API reçoit des requêtes provenant de toutes les adresses IP pour les `IPv4` (`0.0.0.0/0`) et également pour les `IPv6` (`::/0`) dans un cluster `IPv6` à double pile. Si vous limitez l'accès à votre point de terminaison public à l'aide de blocs CIDR, il est recommandé d'activer également l'accès au point de terminaison privé, afin que les nœuds et les pods Fargate Pods (le cas échéant) puissent

communiquer avec le cluster. Sans le point de terminaison privé activé, vos sources CIDR de point d'accès public doivent inclure les sources de sortie de votre VPC. Par exemple, si vous avez un nœud dans un sous-réseau privé qui communique à Internet via une passerelle NAT, vous devez ajouter l'adresse IP sortante de la passerelle NAT dans le cadre d'un bloc CIDR autorisé sur votre point de terminaison public.

7. Choisissez Update (Mettre à jour) pour terminer.

Configuration de l'accès aux points de terminaison - AWS CLI

Effectuez les étapes suivantes à l'aide de la version AWS CLI `1.27.160` ou ultérieure. Vous pouvez vérifier votre version actuelle avec `aws --version`. Pour installer ou mettre à niveau la AWS CLI, reportez-vous à la section Installation de la AWS CLI.

1. Mettez à jour l'accès aux points de terminaison de votre serveur d'API de cluster à l'aide de la commande AWS CLI suivante. Remplacez les valeurs par le nom de votre cluster et les valeurs d'accès au point de terminaison souhaitées. Si vous définissez `endpointPublicAccess=true`, vous pouvez (éventuellement) entrer un seul bloc CIDR ou une liste de blocs CIDR séparés par des virgules pour `publicAccessCidrs`. Les blocs ne peuvent pas inclure d'adresses réservées. Si vous spécifiez des blocs CIDR, le point de terminaison du serveur d'API public ne recevra que les demandes des blocs répertoriés. Vous pouvez spécifier un nombre maximal de blocs CIDR. Pour de plus amples informations, veuillez consulter the section called "Quotas de service". Si vous limitez l'accès à votre point de terminaison public à l'aide de blocs CIDR, il est recommandé d'activer également l'accès au point de terminaison privé afin que les nœuds et les Pods Fargate (si vous les utilisez) puissent communiquer avec le cluster. Sans le point de terminaison privé activé, vos sources CIDR de point d'accès public doivent inclure les sources de sortie de votre VPC. Par exemple, si vous avez un nœud dans un sous-réseau privé qui communique à Internet via une passerelle NAT, vous devez ajouter l'adresse IP sortante de la passerelle NAT dans le cadre d'un bloc CIDR autorisé sur votre point de terminaison public. Si vous ne spécifiez aucun bloc CIDR, le point de terminaison public du serveur d'API reçoit des requêtes provenant de toutes les adresses IP (0.0.0.0/0), ainsi que des `IPv6` (`::/0`) dans le cas d'un cluster `IPv6` à double pile.

> **Note**
>
> La commande suivante active l'accès privé et l'accès public à partir d'une adresse IP unique pour le point de terminaison du serveur API. Remplacez *`203.0.113.5/32`* par

un seul bloc CIDR ou une liste de blocs CIDR séparés par des virgules auxquels vous souhaitez restreindre l'accès réseau.

```
aws eks update-cluster-config \
    --region region-code \
    --name my-cluster \
    --resources-vpc-config
 endpointPublicAccess=true,publicAccessCidrs="203.0.113.5/32",endpointPrivateAccess=true
```

L'exemple qui suit illustre un résultat.

```
{
    "update": {
        "id": "e6f0905f-a5d4-4a2a-8c49-EXAMPLE00000",
        "status": "InProgress",
        "type": "EndpointAccessUpdate",
        "params": [
            {
                "type": "EndpointPublicAccess",
                "value": "true"
            },
            {
                "type": "EndpointPrivateAccess",
                "value": "true"
            },
            {
                "type": "publicAccessCidrs",
                "value": "[\"203.0.113.5/32\"]"
            }
        ],
        "createdAt": 1576874258.137,
        "errors": []
    }
}
```

2. Surveillez le statut de la mise à jour de l'accès à votre point de terminaison avec la commande suivante, en indiquant le nom de votre cluster et l'ID de mise à jour qui a été renvoyé par la commande précédente. Votre mise à jour est terminée lorsqu'elle affiche l'état `Successful`.

```
aws eks describe-update \
```

```
    --region region-code \
    --name my-cluster \
    --update-id e6f0905f-a5d4-4a2a-8c49-EXAMPLE00000
```

L'exemple qui suit illustre un résultat.

```
{
    "update": {
        "id": "e6f0905f-a5d4-4a2a-8c49-EXAMPLE00000",
        "status": "Successful",
        "type": "EndpointAccessUpdate",
        "params": [
            {
                "type": "EndpointPublicAccess",
                "value": "true"
            },
            {
                "type": "EndpointPrivateAccess",
                "value": "true"
            },
            {
                "type": "publicAccessCidrs",
                "value": "[\"203.0.113.5/32\"]"
            }
        ],
        "createdAt": 1576874258.137,
        "errors": []
    }
}
```

Modifiez cette page sur GitHub

Déployer des nœuds Windows sur des clusters EKS

Découvrez comment activer et gérer la prise en charge de Windows pour votre cluster Amazon EKS afin d'exécuter des conteneurs Windows parallèlement à des conteneurs Linux.

Considérations

Avant de déployer des nœuds Windows, veuillez tenir compte des considérations suivantes.

- Le mode automatique EKS ne prend pas en charge les nœuds Windows
- Vous pouvez utiliser le réseau hôte sur les nœuds Windows à l'aide des pods `HostProcess`. Pour plus d'informations, consultez la section Créer une fenêtre HostProcessPod dans la documentation de Kubernetes.
- Les clusters Amazon EKS doivent contenir un ou plusieurs nœuds Linux ou Fargate pour exécuter les pods système de base qui ne fonctionnent que sous Linux, tels que CoreDNS.
- Les journaux d'événements `kubelet` et `kube-proxy` sont journalisés vers le journal d'événements `EKS Windows` et sont limités à 200 Mo.
- Vous ne pouvez pas utiliser Attribuer des groupes de sécurité à des pods individuels avec des pods s'exécutant sur des nœuds Windows.
- Vous ne pouvez pas utiliser la mise en réseau personnalisée avec des nœuds Windows.
- Il n'est pas possible d'utiliser `IPv6` avec les nœuds Windows.
- Les nœuds Windows prennent en charge une interface réseau Elastic par nœud. Par défaut, le nombre de pods pouvant être exécutés par nœud Windows est égal au nombre d'adresses IP disponibles par interface réseau Elastic pour le type d'instance du nœud, moins un. Pour plus d'informations, consultez la section Adresses IP par interface réseau et par type d'instance dans le guide de EC2 l'utilisateur Amazon.
- Dans un cluster Amazon EKS, un seul service avec un équilibreur de charge peut prendre en charge jusqu'à 1 024 pods dorsaux. Chaque pod a sa propre adresse IP. La limite précédente de 64 pods n'est plus le cas, après une mise à jour de Windows Server à partir de OS Build 17763.2746.
- Les conteneurs Windows ne sont pas pris en charge pour les pods Amazon EKS sur Fargate.
- Vous ne pouvez pas utiliser les nœuds hybrides Amazon EKS avec Windows comme système d'exploitation pour l'hôte.
- Vous ne pouvez pas journaliser les journaux à partir du pod `vpc-resource-controller`. Vous le pouviez auparavant lorsque vous déployiez le contrôleur sur le plan de données.
- Il y a un temps de stabilisation avant qu'une adresse `IPv4` ne soit affectée à un nouveau pod. Cela empêche le trafic de s'écouler vers un pod plus ancien avec la même adresse `IPv4` en raison de règles `kube-proxy` périmées.
- La source du contrôleur est gérée sur GitHub. Pour contribuer ou signaler des problèmes contre le contrôleur, consultez le projet sur GitHub.
- Lorsque vous spécifiez un ID AMI personnalisé pour les groupes de nœuds gérés par Windows, ajoutez-le `eks:kube-proxy-windows` à votre carte de configuration AWS IAM Authenticator.

Pour de plus amples informations, veuillez consulter the section called "Limites et conditions lors de la spécification d'un ID d'AMI".

- S'il est essentiel de préserver IPv4 les adresses disponibles pour votre sous-réseau, reportez-vous au Guide des meilleures pratiques d'EKS - Gestion des adresses IP réseau Windows pour obtenir des conseils.
- Considérations relatives aux entrées d'accès EKS
  - Les entrées d'accès à utiliser avec les nœuds Windows doivent être de type `EC2_WINDOWS`. Pour de plus amples informations, veuillez consulter the section called "Créer des entrées d'accès".

    Pour créer une entrée d'accès pour un nœud Windows :

    ```
    aws eks create-access-entry --cluster-name my-cluster --principal-arn arn:aws:
     iam::111122223333:role/<role-name> --type EC2_Windows
    ```

Conditions préalables

- Un cluster existant.
- Votre cluster doit comporter au moins un nœud Linux ou un pod Fargate (nous recommandons au moins deux) pour exécuter CoreDNS. Si vous activez la prise en charge Windows héritée, vous devez utiliser un nœud Linux (vous ne pouvez pas utiliser un pod Fargate) pour exécuter CoreDNS.
- Un rôle IAM de cluster Amazon EKS existant.

Activer la prise en charge de Windows

1. Si votre cluster ne comporte pas de nœuds Amazon Linux et que vous utilisez des groupes de sécurité pour les pods, passez à l'étape suivante. Sinon, confirmez que la politique gérée `AmazonEKSVPCResourceController` est attachée à votre rôle de cluster. Remplacez *`eksClusterRole`* par le nom de rôle de votre cluster.

   ```
   aws iam list-attached-role-policies --role-name eksClusterRole
   ```

   L'exemple qui suit illustre un résultat.

```
{
    "AttachedPolicies": [
        {
            "PolicyName": "AmazonEKSClusterPolicy",
            "PolicyArn": "arn:aws: iam::aws:policy/AmazonEKSClusterPolicy"
        },
        {
            "PolicyName": "AmazonEKSVPCResourceController",
            "PolicyArn": "arn:aws: iam::aws:policy/AmazonEKSVPCResourceController"
        }
    ]
}
```

Si la politique est attachée, comme c'est le cas dans la sortie précédente, passez à l'étape suivante.

2. Associez la politique gérée par Amazon EKSVPCResource Controller au rôle IAM de votre cluster Amazon EKS. Remplacez *eksClusterRole* par le nom de rôle de votre cluster.

```
aws iam attach-role-policy \
  --role-name eksClusterRole \
  --policy-arn arn:aws: iam::aws:policy/AmazonEKSVPCResourceController
```

3. Mettez à jour le VPC CNI ConfigMap pour activer Windows IPAM. Notez que si le VPC CNI est installé sur votre cluster à l'aide d'un graphique Helm ou en tant que module complémentaire Amazon EKS, vous ne pourrez peut-être pas modifier directement le. ConfigMap Pour plus d'informations sur la configuration des modules complémentaires Amazon EKS, consultez the section called "Champs que vous pouvez personnaliser".

   a. Créez un fichier nommé *vpc-resource-controller-configmap.yaml* avec les contenus suivants.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: amazon-vpc-cni
  namespace: kube-system
data:
  enable-windows-ipam: "true"
```

   b. Appliquer le `ConfigMap` à votre cluster.

```
kubectl apply -f vpc-resource-controller-configmap.yaml
```

4. Si le mode d'authentification de votre cluster est défini pour activer la configmap `aws-auth` :

   - Vérifiez que votre `aws-auth` `ConfigMap` contient un mappage pour le rôle d'instance du nœud Windows afin d'inclure le groupe d'autorisation RBAC `eks:kube-proxy-windows`. Vous pouvez vérifier en exécutant la commande suivante.

     ```
     kubectl get configmap aws-auth -n kube-system -o yaml
     ```

     L'exemple qui suit illustre un résultat.

     ```
     apiVersion: v1
     kind: ConfigMap
     metadata:
       name: aws-auth
       namespace: kube-system
     data:
       mapRoles: |
         - groups:
           - system:bootstrappers
           - system:nodes
           - eks:kube-proxy-windows # This group is required for Windows DNS resolution
      to work
           rolearn: arn:aws: iam::111122223333:role/eksNodeRole
           username: system:node:{{EC2PrivateDNSName}}
     [...]
     ```

     Vous devriez voir `eks:kube-proxy-windows` répertorié sous les groupes. Si le groupe n'est pas spécifié, vous devez mettre à jour votre `ConfigMap` ou le créer pour inclure le groupe requis. Pour en savoir plus sur `ConfigMap` `aws-auth`, consultez the section called "Appliquer la ConfigMap aws-auth à votre cluster".

5. Si le mode d'authentification de votre cluster est défini pour désactiver la configmap `aws-auth`, vous pouvez utiliser les entrées d'accès EKS. Créez un nouveau rôle de nœud à utiliser avec les instances Windows, et EKS créera automatiquement une entrée d'accès de type `EC2_WINDOWS`.

Déployer des pods Windows

Lorsque vous déployez des pods sur votre cluster, vous devez spécifier le système d'exploitation qu'ils utilisent si vous exécutez plusieurs types de nœuds.

Pour les pods Linux, utilisez le texte de sélecteur de nœud suivant dans vos manifestes.

```
nodeSelector:
        kubernetes.io/os: linux
        kubernetes.io/arch: amd64
```

Pour les pods Windows, utilisez le texte de sélecteur de nœud suivant dans vos manifestes.

```
nodeSelector:
        kubernetes.io/os: windows
        kubernetes.io/arch: amd64
```

Vous pouvez déployer un exemple d'application pour voir les sélecteurs de nœuds utilisés.

Prise en charge d'une densité de pods plus élevée sur les nœuds Windows

Dans Amazon EKS, chaque pod se voit attribuer une adresse `IPv4` provenant de votre VPC. De ce fait, le nombre de pods que vous pouvez déployer sur un nœud est limité par les adresses IP disponibles, même si les ressources sont suffisantes pour exécuter davantage de pods sur le nœud. Étant donné qu'une seule interface réseau élastique est prise en charge par un nœud Windows, par défaut, le nombre maximal d'adresses IP disponibles sur un nœud Windows est égal à :

```
Number of private IPv4 addresses for each interface on the node - 1
```

Une adresse IP est utilisée comme adresse IP principale de l'interface réseau, elle ne peut donc pas être attribuée aux pods.

Vous pouvez activer une densité de pods plus élevée sur les nœuds Windows en activant la délégation de préfixe IP. Cette fonctionnalité vous permet d'attribuer un préfixe `/28` `IPv4` à l'interface réseau principale, au lieu d'attribuer des adresses secondaires `IPv4`. L'attribution d'un préfixe IP augmente le nombre maximum d'adresses `IPv4` disponibles sur le nœud à :

```
(Number of private IPv4 addresses assigned to the interface attached to the node - 1) *
 16
```

Grâce à ce nombre considérablement plus important d'adresses IP disponibles, celles-ci ne devraient plus limiter votre capacité à faire évoluer le nombre de pods sur vos nœuds. Pour de plus amples informations, veuillez consulter the section called "Augmentation du nombre d'adresses IP".

Désactiver la prise en charge Windows

1. Si votre cluster contient des nœuds Amazon Linux et que vous utilisez des groupes de sécurité pour les pods avec eux, passez cette étape.

   Supprimez la politique IAM gérée `AmazonVPCResourceController` de votre rôle de cluster. Remplacez *`eksClusterRole`* par le nom de votre rôle de cluster.

   ```
   aws iam detach-role-policy \
       --role-name eksClusterRole \
       --policy-arn arn:aws:iam::aws:policy/AmazonEKSVPCResourceController
   ```

2. Désactivez Windows IPAM dans le ConfigMap `amazon-vpc-cni`.

   ```
   kubectl patch configmap/amazon-vpc-cni \
                       -n kube-system \
                       --type merge \
                       -p '{"data":{"enable-windows-ipam":"false"}}'
   ```

Déployer des clusters privés avec un accès Internet limité

Cette rubrique décrit comment déployer un cluster Amazon EKS déployé sur le AWS cloud, mais ne disposant pas d'un accès Internet sortant. Si vous avez un cluster local sur AWS Outposts, consultezthe section called "Nœuds", au lieu de cette rubrique.

Si vous n'êtes pas familier avec la mise en réseau Amazon EKS, consultez Démystifier la mise en réseau des clusters pour les composants master Amazon EKS. Si votre cluster ne dispose pas d'un accès Internet sortant, il doit répondre aux exigences suivantes :

Exigences en matière d'architecture de cluster

- Votre cluster doit extraire les images d'un registre de conteneurs situé dans votre VPC. Vous pouvez créer un registre de conteneurs Amazon Elastic Container Registry dans votre VPC et y

copier des images de conteneurs pour que vos nœuds puissent les extraire. Pour de plus amples informations, veuillez consulter the section called “Copier une image dans un référentiel”.

- Votre cluster doit avoir un accès privé au point de terminaison activé. Ceci est nécessaire pour que les nœuds s'enregistrent auprès du point de terminaison du cluster. L'accès public au point de terminaison est facultatif. Pour de plus amples informations, veuillez consulter the section called “Accès au point de terminaison de cluster”.

Exigences relatives aux nœuds

- Les nœuds Linux et Windows autogérés doivent inclure les arguments de démarrage suivants avant leur lancement. Ces arguments contournent l'introspection Amazon EKS et ne nécessitent pas d'accès à l'API Amazon EKS depuis le VPC.

  a. Déterminez la valeur du point de terminaison de votre cluster à l'aide de la commande suivante. Remplacez *my-cluster* par le nom de votre cluster.

  ```
  aws eks describe-cluster --name my-cluster --query cluster.endpoint --output text
  ```

  L'exemple qui suit illustre un résultat.

  ```
  https://EXAMPLE108C897D9B2F1B21D5EXAMPLE.sk1.region-code.eks.amazonaws.com
  ```

  b. Déterminez la valeur de l'autorité de certification de votre cluster à l'aide de la commande suivante. Remplacez *my-cluster* par le nom de votre cluster.

  ```
  aws eks describe-cluster --name my-cluster --query cluster.certificateAuthority --
  output text
  ```

  La sortie renvoyée est une longue chaîne.

  c. Remplacez les valeurs de `apiServerEndpoint` et contenues `certificateAuthority` dans l' NodeConfig objet par les valeurs renvoyées dans le résultat des commandes précédentes. Pour plus d'informations sur la spécification des arguments bootstrap lors du lancement de nœuds Amazon Linux 2023 autogérés, consultez the section called “Amazon Linux” et. the section called “Windows”

  - Pour les nœuds Linux :

  ```
  ---
  ```

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="BOUNDARY"

--BOUNDARY
Content-Type: application/node.eks.aws

---
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
    name: my-cluster
    apiServerEndpoint: [.replaceable]https://
EXAMPLE108C897D9B2F1B21D5EXAMPLE.sk1.region-code.eks.amazonaws.com
    certificateAuthority: [.replaceable]Y2VydGlmaWNhdGVBdXRob3JpdHk=
    ...
```

Pour des arguments supplémentaires, consultez le script bootstrap sur GitHub.

- Pour les nœuds Windows :

> **Note**
>
> Si vous utilisez un CIDR de service personnalisé, vous devez le spécifier à l'aide du paramètre `-ServiceCIDR`. Sinon, la résolution DNS pour les pods dans le cluster échouera.

```
-APIServerEndpoint cluster-endpoint -Base64ClusterCA certificate-authority
```

Pour des arguments supplémentaires, consultez the section called "Paramètres de configuration du script d'amorçage".

- Le `aws-auth` `ConfigMap` de votre cluster doit être créé à partir de votre VPC. Pour plus d'informations sur la création et l'ajout d'entrées dans la `ConfigMap` `aws-auth`, entrez `eksctl create iamidentitymapping --help` dans votre terminal. Si le `ConfigMap` n'existe pas sur votre serveur, `eksctl` le créera lorsque vous utiliserez la commande pour ajouter un mappage d'identité.

Exigences relatives aux pods

- Identité du pod : les pods configurés avec l'identité du pod EKS obtiennent leurs informations d'identification à partir de l'API EKS Auth. S'il n'y a pas d'accès Internet sortant, vous devez créer et utiliser un point de terminaison d'un VPC pour l'API EKS Auth : `com.amazonaws.region-code.eks-auth`. Pour plus d'informations sur les points de terminaison d'un VPC EKS et EKS Auth, consultez the section called " AWS PrivateLink".
- IRSA - Les pods configurés avec des rôles IAM pour les comptes de service obtiennent des informations d'identification à partir d'un appel d'API du AWS Security Token Service (AWS STS). S'il n'y a pas d'accès Internet sortant, vous devez créer et utiliser un point de terminaison VPC AWS STS dans votre VPC. La plupart AWS `v1` SDKs utilisent le point de terminaison AWS STS global par défaut (`sts.amazonaws.com`), qui n'utilise pas le point de terminaison VPC AWS STS. Pour utiliser le point de terminaison VPC AWS STS, vous devrez peut-être configurer votre SDK pour utiliser le point de terminaison AWS STS régional (). `sts.`*`region-code`*`.amazonaws.com` Pour de plus amples informations, veuillez consulter the section called "Critères d'évaluation STS".
- Les sous-réseaux VPC de votre cluster doivent disposer d'un point de terminaison d'interface VPC pour tous les AWS services auxquels vos pods doivent accéder. Pour plus d'informations, consultez Accéder à un AWS service à l'aide d'un point de terminaison VPC d'interface. Certains services et points de terminaison couramment utilisés sont répertoriés dans le tableau suivant. Pour une liste complète des points de terminaison, consultez Services AWS qui s'intègrent à AWS PrivateLink dans le Guide AWS PrivateLink .

  Nous vous recommandons d'activer les noms DNS privés pour vos points de terminaison VPC, afin que les charges de travail puissent continuer à utiliser les points de terminaison de AWS service public sans problème.

Service	Endpoint
Amazon EC2	com.amazonaws. *region-code* .ec2
Amazon Elastic Container Registry (pour extraire des images de conteneurs)	com.amazonaws. *region-code* .ecr.api, com.amazonaws. *region-code* .ecr.dkr et com.amazonaws. *region-code* .s3
Application Load Balancers et Network Load Balancers Amazon	com.amazonaws. *region-code* . équilibrage de charge élastique

Service	Endpoint
(Facultatif) AWS X-Ray (obligatoire pour le traçage envoyé à AWS X-Ray)	com.amazonaws. *region-code* .xray
(Facultatif) Amazon SSM (requis pour l'agent SSM pour les tâches de gestion des nœuds. Alternative à SSH)	com.amazonaws. *region-code* .ssm
Amazon CloudWatch Logs (obligatoire pour les journaux de nœuds et de pods envoyés à Amazon CloudWatch Logs)	com.amazonaws. *region-code* .journaux
AWS Service de jeton de sécurité (obligato ire lors de l'utilisation de rôles IAM pour les comptes de service)	com.amazonaws. *region-code* .sts
Amazon EKS Auth (requis lors de l'utilisation d'associations d'identités du pod)	com.amazonaws. *region-code* .eks-auth
Amazon EKS	com.amazonaws. *region-code* .eks

- Tous les nœuds autogérés doivent être déployés sur des sous-réseaux qui disposent des points de terminaison de l'interface VPC dont vous avez besoin. Si vous créez un groupe de nœuds gérés, le groupe de sécurité des points de terminaison de l'interface VPC doit autoriser le CIDR pour les sous-réseaux, ou vous devez ajouter le groupe de sécurité des nœuds créé au groupe de sécurité des points de terminaison de l'interface VPC.
- Stockage EFS : si vos pods utilisent des volumes Amazon EFS, avant de déployer le système de fichiers élastique Store avec Amazon EFS, le fichier kustomization.yaml du pilote doit être modifié pour que les images du conteneur utilisent la AWS même région que le cluster Amazon EKS.
- Route53 ne prend pas en charge. AWS PrivateLink Vous ne pouvez pas gérer les enregistrements DNS Route53 à partir d'un cluster Amazon EKS privés. Cela a un impact sur Kubernetes external-dns.
- Si vous utilisez l'AMI optimisée EKS, vous devez activer le point de terminaison `ec2` dans le tableau ci-dessus. Vous pouvez également définir manuellement le nom DNS du nœud. L'AMI optimisée permet EC2 APIs de définir automatiquement le nom DNS du nœud.
- Vous pouvez utiliser le AWS Load Balancer Controller pour déployer des équilibreurs de charge AWS d'application (ALB) et des équilibreurs de charge réseau sur votre cluster privé. Lors de

son déploiement, vous devez utiliser les indicateurs de ligne de commande pour définir `enable-shield`, `enable-waf` et `enable-wafv2` sur false. La détection de certificats avec des noms d'hôte provenant d'objets Ingress n'est pas prise en charge. Cela est dû au fait que le contrôleur doit atteindre AWS Certificate Manager, qui ne possède pas de point de terminaison d'interface VPC.

Le contrôleur prend en charge les Network Load Balancers avec des cibles IP, qui sont nécessaires pour une utilisation avec Fargate. Pour plus d'informations, consultez the section called "Répartition de la charge des applications" et the section called "Créer un équilibreur de charge de réseau".

- Cluster Autoscaler est pris en charge. Lors du déploiement de pods Cluster Autoscaler, assurez-vous que la ligne de commande inclut `--aws-use-static-instance-list=true`. Pour plus d'informations, consultez la section Utiliser une liste d'instances statiques sur GitHub. Le VPC du nœud de travail doit également inclure le point de terminaison VPC STS et le point de AWS terminaison VPC à mise à l'échelle automatique.
- Certains logiciels de conteneur utilisent des appels d'API qui accèdent au AWS Marketplace Metering Service pour surveiller l'utilisation. Les clusters privés n'autorisent pas ces appels, vous ne pouvez donc pas utiliser ces types de conteneurs dans les clusters privés.

Mise à l'échelle du calcul du cluster avec Karpenter et Cluster Autoscaler

La scalabilité automatique est une fonction qui augmente ou diminue automatiquement vos ressources en fonction de l'évolution de la demande. Il s'agit d'une fonction majeure de Kubernetes qui, autrement, nécessiterait des ressources humaines importantes pour être exécutée manuellement.

Mode automatique EKS

Le mode automatique Amazon EKS met automatiquement à l'échelle les ressources de cluster computing. Si un pod ne peut pas être planifié sur les nœuds existants, le mode automatique EKS en crée un nouveau. Le mode automatique EKS consolide également les charges de travail et supprime les nœuds inutilisés. Le mode automatique EKS s'appuie sur Karpenter.

Pour plus d'informations, consultez :

- Mode automatique EKS

- the section called "Création d'un groupe de nœuds"
- the section called "Déploiement d'une charge de travail de gonflement"

Solutions supplémentaires

Amazon EKS prend en charge deux produits d'autoscaling supplémentaires :

Karpenter

Karpenter est un autoscaler de cluster Kubernetes flexible et performant qui permet d'améliorer la disponibilité des applications et l'efficacité du cluster. Karpenter lance des ressources de calcul dimensionnées adéquatement (par exemple, des instances Amazon EC2) en moins d'une minute, en réponse à la variation de la charge des applications. Grâce à l'intégration de Kubernetes avec AWS, Karpenter peut allouer des ressources de calcul juste-à-temps qui répondent précisément aux exigences de votre charge de travail. Karpenter alloue automatiquement de nouvelles ressources de calcul en fonction des exigences spécifiques des charges de travail du cluster. Celles-ci comprennent les exigences en matière de calcul, de stockage, d'accélération et de planification. Amazon EKS prend en charge les clusters utilisant Karpenter, bien que Karpenter fonctionne avec tout cluster Kubernetes conforme. Pour en savoir plus, veuillez consulter la documentation Karpenter.

> **Important**
>
> Karpenter est un logiciel open source que les clients AWS sont responsables d'installer, de configurer et de gérer dans leurs clusters Kubernetes. AWS fournit un soutien technique lorsque Karpenter est exécuté sans modification, dans une version compatible, sur des clusters Amazon EKS. Il est essentiel que les clients maintiennent la disponibilité et la sécurité du contrôleur Karpenter ainsi que les procédures de test appropriées lors de sa mise à niveau ou du cluster Kubernetes dans lequel il s'exécute, comme tout autre logiciel géré par le client. Karpenter n'a pas de Contrat de niveau de service (SLA) AWS, et il revient aux clients de s'assurer que les instances EC2 lancées par Karpenter répondent à leurs exigences opérationnelles.

Cluster Autoscaler

Le Kubernetes Cluster Autoscaler ajuste automatiquement le nombre de nœuds dans votre cluster lorsque les pods échouent ou sont reprogrammés sur d'autres nœuds. Le Cluster

Autoscaler utilise des groupes Auto Scaling. Pour plus d'informations, consultez Cluster Autoscaler sur AWS.

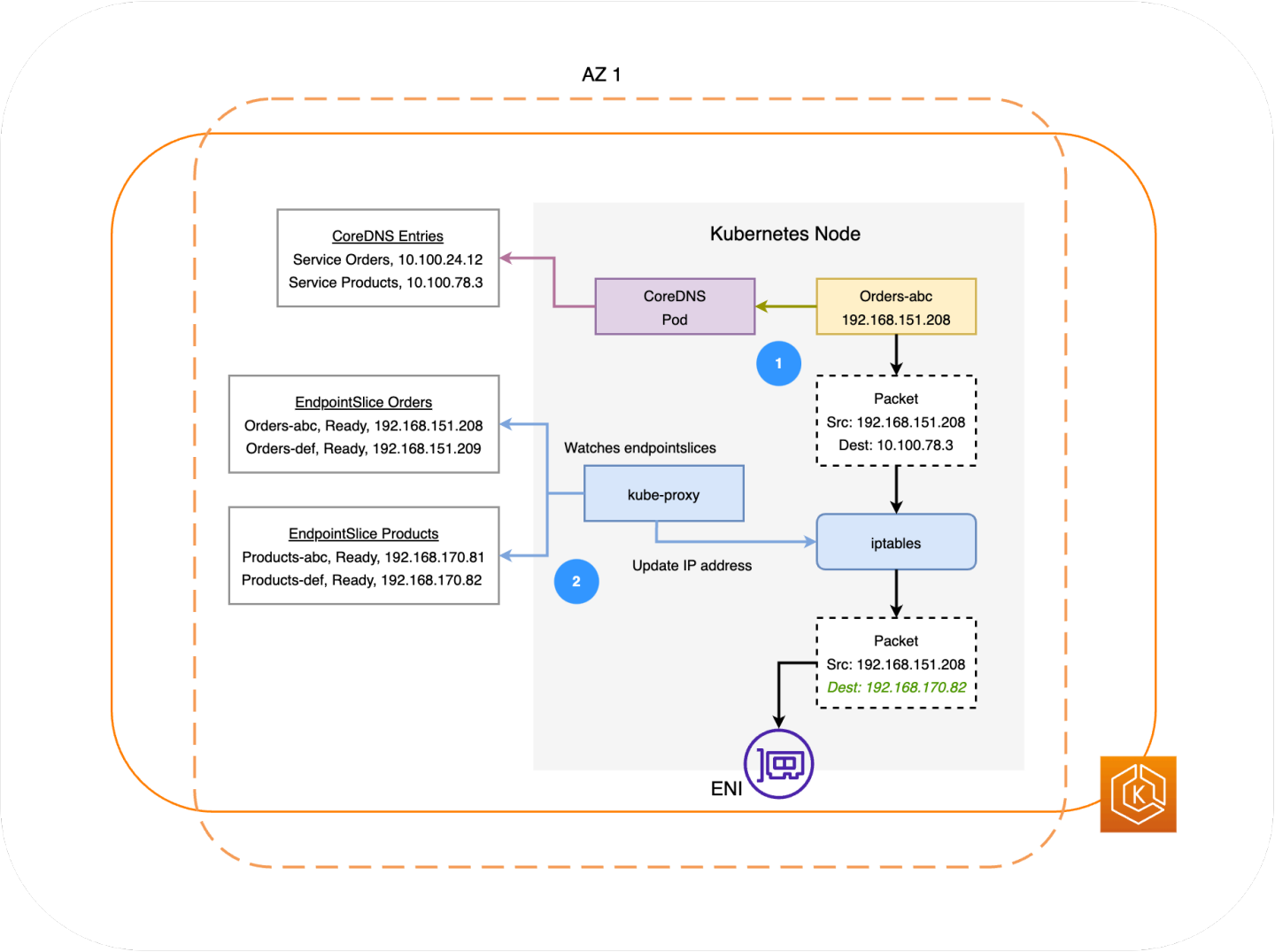
Découvrez le changement de zone du contrôleur de récupération d'application (ARC) Amazon dans Amazon EKS

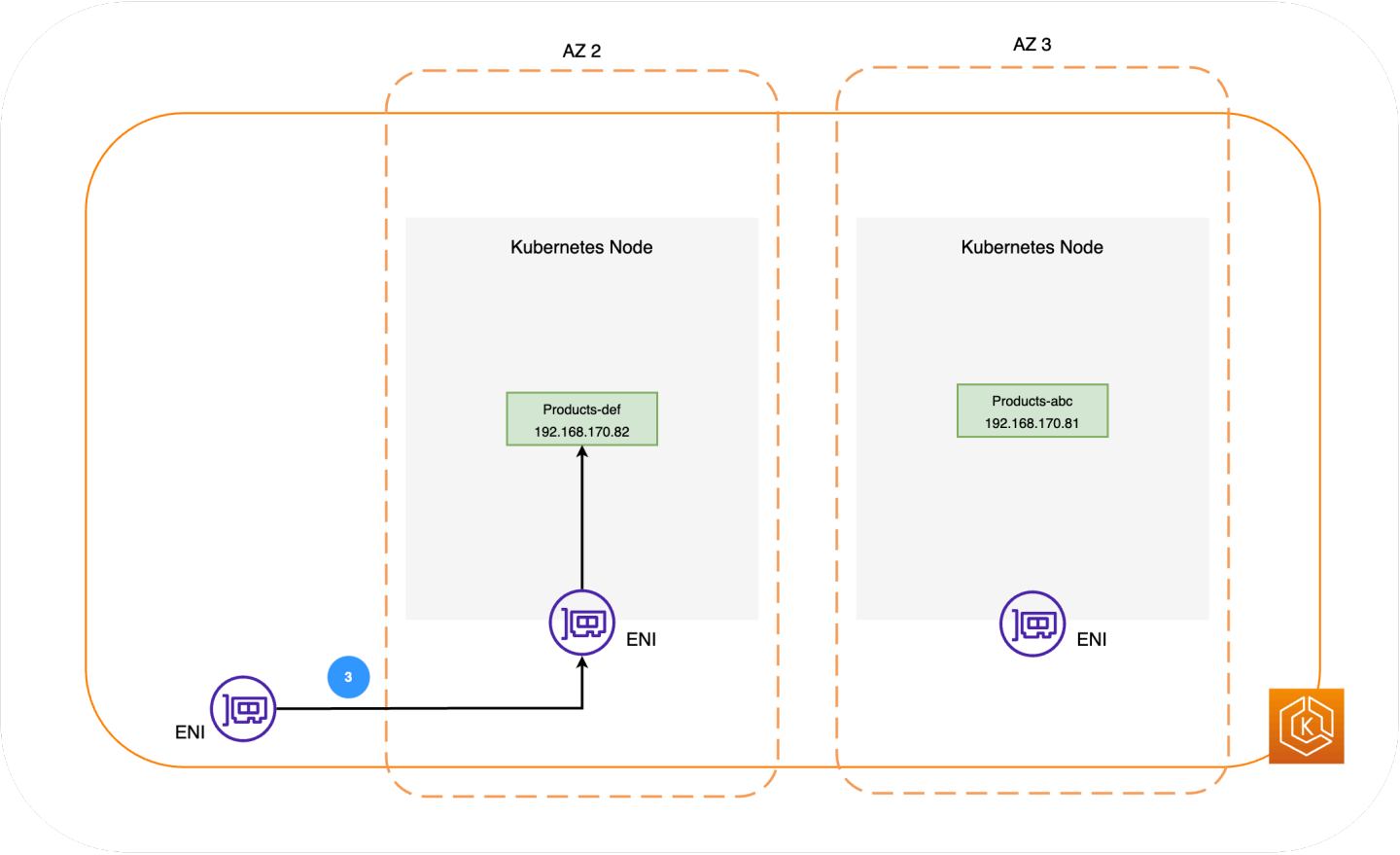
Kubernetes dispose de fonctionnalités natives qui vous permettent de rendre vos applications plus résilientes face à des événements tels que la dégradation de l'état ou la défaillance d'une zone de disponibilité (AZ). Lorsque vous exécutez vos charges de travail dans un cluster Amazon EKS, vous pouvez améliorer davantage la tolérance aux pannes et la récupération de votre environnement d'application en utilisant le changement de zone du contrôleur de récupération d'application (ARC) Amazon ou le changement automatique de zone. Le changement de zone ARC est conçu comme une mesure temporaire qui vous permet de déplacer le trafic d'une ressource hors d'une AZ défaillante jusqu'à l'expiration du changement de zone ou jusqu'à ce que vous l'annuliez. Vous pouvez prolonger le changement de zone si nécessaire.

Vous pouvez démarrer un changement de zone pour un cluster EKS, ou vous pouvez autoriser AWS le transfert de trafic pour vous en activant le changement automatique de zone. Ce changement met à jour le flux de trafic east-to-west réseau dans votre cluster afin de ne considérer que les points de terminaison réseau des pods exécutés sur des nœuds de travail comme sains AZs. En outre, tout ALB ou NLB gérant le trafic entrant pour les applications de votre cluster EKS acheminera automatiquement le trafic vers les cibles saines. AZs Pour les clients qui recherchent une disponibilité maximale, dans le cas où une zone de disponibilité serait défaillante, il peut être important de pouvoir détourner tout le trafic de la zone défaillante jusqu'à ce qu'elle soit rétablie. Pour cela, vous pouvez également activer un ALB ou un NLB avec le changement de zone ARC.

Comprendre le flux de trafic réseau est-ouest entre les pod

Le diagramme suivant illustre deux exemples de charges de travail, Orders et Products. Le but de cet exemple est de montrer comment les charges de travail et les pods AZs communiquent entre eux dans différents environnements.


AZ 1
CoreDNS Entries
Service Orders, 10.100.24.12
Service Products, 10.100.78.3
Kubernetes Node
CoreDNS
Pod
Orders-abc
192.168.151.208
1
Packet
Src: 192.168.151.208
Dest: 10.100.78.3
EndpointSlice Orders
Orders-abc, Ready, 192.168.151.208
Orders-def, Ready, 192.168.151.209
Watches endpointslices
kube-proxy
iptables
EndpointSlice Products
Products-abc, Ready, 192.168.170.81
Products-def, Ready, 192.168.170.82
2
Update IP address
Packet
Src: 192.168.151.208
Dest: 192.168.170.82
ENI



1. Pour que Orders puisse communiquer avec Products, Orders doit d'abord résoudre le nom DNS du service de destination. Orders communique avec CoreDNS pour récupérer l'adresse IP virtuelle (IP du cluster) de ce service. Une fois que Orders a résolu le nom du service Products, il envoie le trafic vers cette adresse IP cible.
2. Le kube-proxy s'exécute sur chaque nœud du cluster et surveille en permanence EndpointSlicesles services. Lorsqu'un service est créé, un EndpointSlice est créé et géré en arrière-plan par le EndpointSlice contrôleur. Chacun EndpointSlice possède une liste ou un tableau de points de terminaison contenant un sous-ensemble d'adresses Pod, ainsi que les nœuds sur lesquels ils s'exécutent. Le kube-proxy configure des règles de routage pour chacun de ces points de terminaison de pods à l'aide de `iptables` sur les nœuds. Le kube-proxy est également responsable d'une forme basique d'équilibrage de charge, redirigeant le trafic destiné à l'adresse IP du cluster d'un service vers l'adresse IP d'un pod directement. Le kube-proxy effectue cette opération en réécrivant l'adresse IP de destination sur la connexion sortante.
3. Les paquets réseau sont ensuite envoyés au Products Pod dans AZ 2 en utilisant ENIs les nœuds respectifs, comme indiqué dans le schéma précédent.

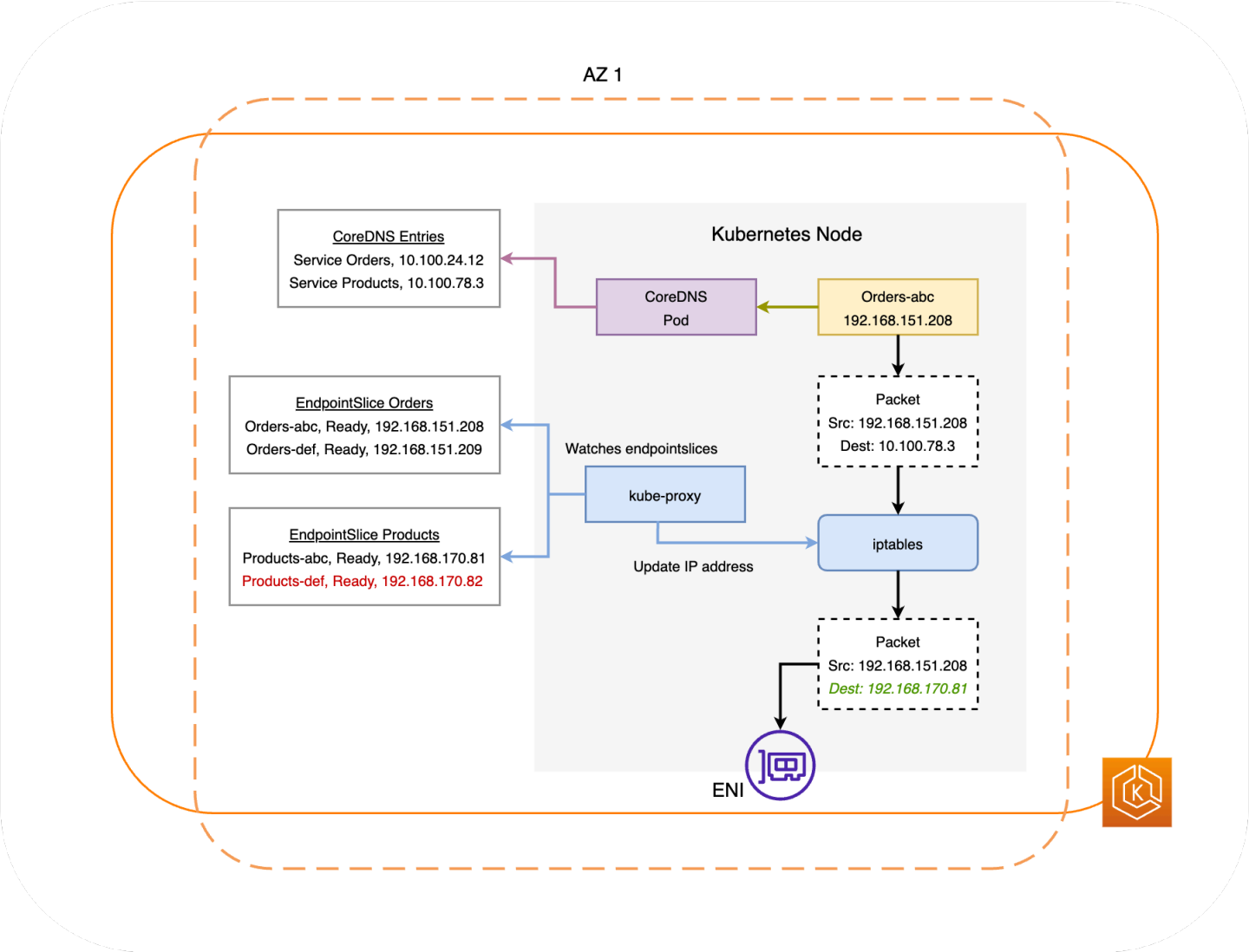
Comprendre le changement de zone ARC dans Amazon EKS

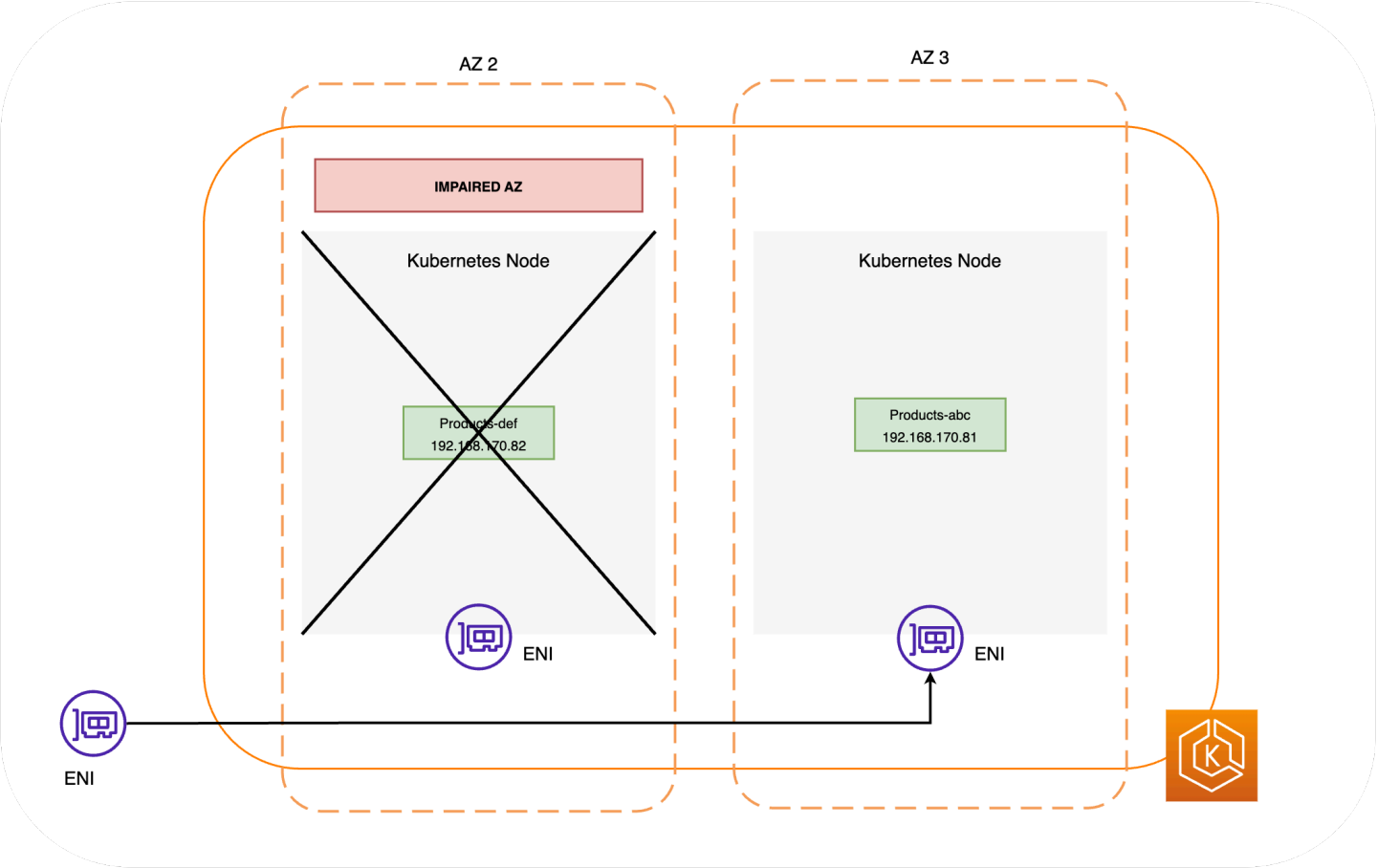
En cas de défaillance d'une zone de disponibilité dans votre environnement, vous pouvez initier un changement de zone pour votre environnement de cluster Amazon EKS. Vous pouvez également autoriser la gestion du trafic changeant AWS à votre place grâce à l'autoshift zonal. Grâce à l'autoshift zonal, il AWS surveille l'état général de la zone Z et répond à une éventuelle détérioration de la zone Z en transférant automatiquement le trafic vers l'autre zone affectée dans votre environnement de cluster.

Une fois que le décalage de zone est activé avec ARC sur votre cluster Amazon EKS, vous pouvez démarrer un changement de zone ou activer le décalage automatique de zone à l'aide de la console ARC, de la AWS CLI ou du décalage zonal et du décalage automatique de zone. APIs Lors d'un changement de zone EKS, les opérations suivantes sont effectuées automatiquement :

- Tous les nœuds de la zone de disponibilité affectée sont isolés. Cela empêche le planificateur Kubernetes de planifier de nouveaux pods sur les nœuds de la zone de disponibilité défaillante.
- Si vous utilisez des groupes de nœuds gérés, le rééquilibrage des zones de disponibilité est suspendu et votre groupe Auto Scaling est mis à jour pour garantir que les nouveaux nœuds du plan de données EKS ne sont lancés qu'en bon AZs état.
- Les nœuds de la zone de disponibilité défaillante ne sont pas résiliés et les pods ne sont pas expulsés des nœuds. Cela garantit que lorsque le changement de zone expire ou est annulé, votre trafic peut être renvoyé en toute sécurité vers la zone de disponibilité pour une capacité maximale.
- Le EndpointSlice contrôleur trouve tous les points de terminaison du Pod dans la zone AZ altérée et les retire de la zone correspondante EndpointSlices. Cela garantit que seuls les terminaux Pod sains AZs sont ciblés pour recevoir du trafic réseau. Lorsqu'un décalage de zone est annulé ou expire, le EndpointSlice contrôleur le met à jour EndpointSlices pour inclure les points de terminaison dans l'AZ restaurée.

Les diagrammes suivants fournissent une vue d'ensemble de la manière dont le changement de zone EKS garantit que seuls les points de terminaison de pods sains sont ciblés dans votre environnement de cluster.


AZ 1
Kubernetes Node
CoreDNS Entries
Service Orders, 10.100.24.12
Service Products, 10.100.78.3
CoreDNS
Pod
Orders-abc
192.168.151.208
Packet
Src: 192.168.151.208
Dest: 10.100.78.3
EndpointSlice Orders
Orders-abc, Ready, 192.168.151.208
Orders-def, Ready, 192.168.151.209
Watches endpointslices
kube-proxy
EndpointSlice Products
Products-abc, Ready, 192.168.170.81
Products-def, Ready, 192.168.170.82
Update IP address
iptables
Packet
Src: 192.168.151.208
Dest: 192.168.170.81
ENI



Exigences relatives au changement de zone EKS

Pour que le changement de zone fonctionne correctement avec EKS, vous devez configurer votre environnement de cluster à l'avance afin qu'il soit résilient en cas de défaillance d'une zone de disponibilité. Voici une liste d'options de configuration qui contribuent à garantir la résilience.

- Provisionnez les nœuds de travail de votre cluster sur plusieurs AZs
- Provisionnez une capacité de calcul suffisante pour faire face à la suppression d'une seule AZ
- Pré-dimensionnez vos pods, y compris CoreDNS, dans chaque AZ
- Répartissez plusieurs répliques de Pod sur l'ensemble du site AZs, afin de garantir que lorsque vous abandonnerez un seul AZ, vous disposerez toujours d'une capacité suffisante
- Effectuez la colocalisation des pods interdépendants ou liés dans la même zone de disponibilité
- Vérifiez que votre environnement de cluster fonctionne comme prévu sans une zone de disponibilité en démarrant manuellement un changement de zone à partir d'une zone de disponibilité. Vous pouvez également activer le transfert automatique de zone et vous fier aux essais de transfert automatique. Il n'est pas nécessaire de tester les déplacements manuels ou les

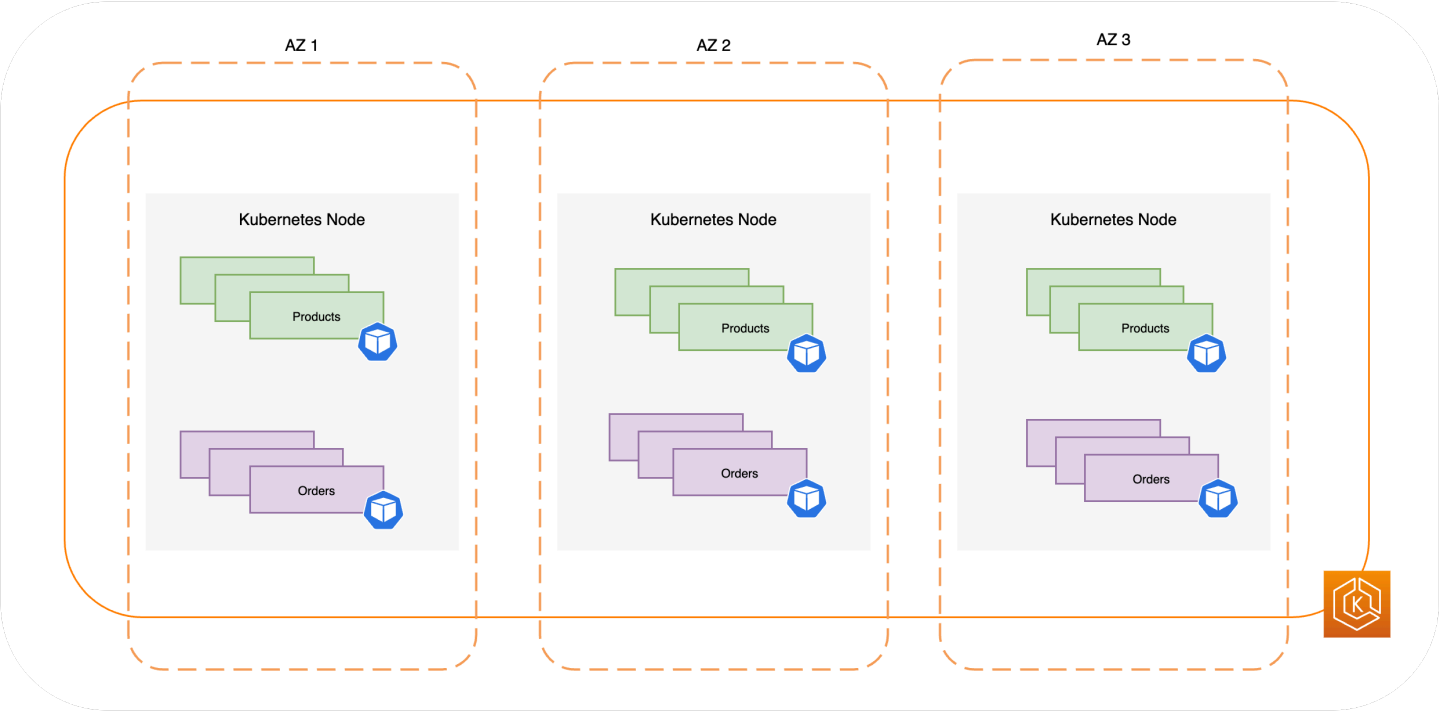
tests de déplacement de zone pour que le changement de zone fonctionne dans EKS, mais cela est fortement recommandé.

Provisionnez vos composants master EKS dans plusieurs zones de disponibilité

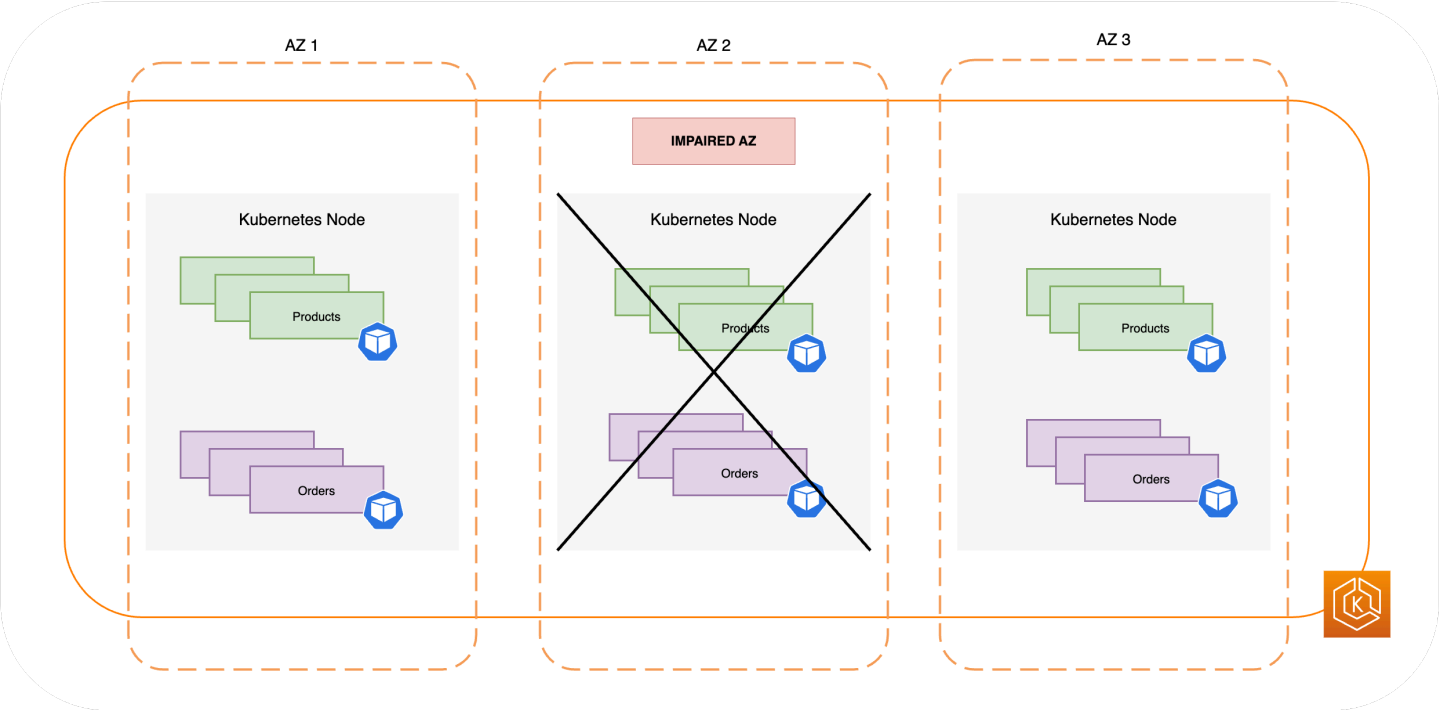
AWS Les régions disposent de plusieurs sites distincts dotés de centres de données physiques, appelés zones de disponibilité (AZs). AZs sont conçus pour être physiquement isolés les uns des autres afin d'éviter un impact simultané susceptible d'affecter une région entière. Lorsque vous provisionnez un cluster EKS, nous vous recommandons de déployer vos nœuds de travail sur plusieurs nœuds au AZs sein d'une même région. Cela permet de rendre votre environnement de cluster plus résilient face à la détérioration d'une seule AZ et vous permet de maintenir une haute disponibilité pour vos applications qui s'exécutent dans l'autre AZs. Lorsque vous amorcez un changement de zone par rapport à l'AZ concernée, le réseau intégré au cluster de votre environnement EKS se met automatiquement à jour pour n'utiliser que les zones saines AZs, afin de maintenir la haute disponibilité de votre cluster.

Le fait de disposer d'une configuration multi-AZ pour votre environnement EKS améliore la fiabilité globale de votre système. Cependant, les environnements multi-AZ influencent la manière dont les données des applications sont transférées et traitées, ce qui a un impact sur les frais de réseau de votre environnement. Plus précisément, le trafic sortant fréquent entre les zones (trafic réparti entre les zones AZs) peut avoir un impact majeur sur les coûts liés à votre réseau. Vous pouvez appliquer différentes stratégies pour contrôler le volume de trafic inter-zones entre les pods de votre cluster EKS et réduire les coûts associés. Pour plus d'informations sur la manière d'optimiser les coûts réseau lors de l'exécution d'environnements EKS hautement disponibles, consultez ces bonnes pratiques.

Le schéma suivant illustre un environnement EKS à haute disponibilité avec trois environnements sains. AZs



Le schéma suivant montre comment un environnement EKS composé de trois AZs est résilient à une déficience de l'AZ et reste hautement disponible car il en reste deux en bonne santé AZs.



Provisionner une capacité de calcul suffisante pour faire face à la suppression d’une seule zone de disponibilité

Afin d’optimiser l’utilisation des ressources et les coûts de votre infrastructure de calcul dans le plan de données EKS, il est une bonne pratique d’aligner la capacité de calcul sur les exigences de votre charge de travail. Cependant, si tous vos composants master sont à pleine capacité, vous devez ajouter de nouveaux composants master au plan de données EKS avant de pouvoir planifier de nouveaux pods. Lorsque vous exécutez des charges de travail critiques, il est généralement recommandé de disposer d’une capacité redondante en ligne pour faire face à des scénarios tels que des augmentations soudaines de la charge et des problèmes d’état des nœuds. Si vous prévoyez d’utiliser le changement de zone, vous envisagez de supprimer toute la capacité d’une zone de disponibilité en cas de défaillance. Cela signifie que vous devez ajuster votre capacité de calcul redondante de manière à ce qu'elle soit suffisante pour gérer la charge, même en AZs mode hors ligne.

Lorsque vous mettez à l’échelle vos ressources de calcul, le processus d’ajout de nouveaux nœuds au plan de données EKS prend un certain temps. Cela peut avoir des implications sur les performances en temps réel et la disponibilité de vos applications, en particulier en cas de défaillance de zone. Votre environnement EKS doit être capable d’absorber la charge liée à la perte d’une zone de disponibilité sans entraîner de dégradation de l’expérience pour vos utilisateurs finaux ou vos clients. Cela implique de réduire au minimum ou d’éliminer le décalage entre le moment où un nouveau pod est nécessaire et celui où il est effectivement planifié sur un composants master.

En outre, en cas d'altération de la zone, vous devez vous efforcer d'atténuer le risque de vous heurter à une contrainte de capacité de calcul qui empêcherait l'ajout de nouveaux nœuds à votre plan de données EKS en mode sain. AZs

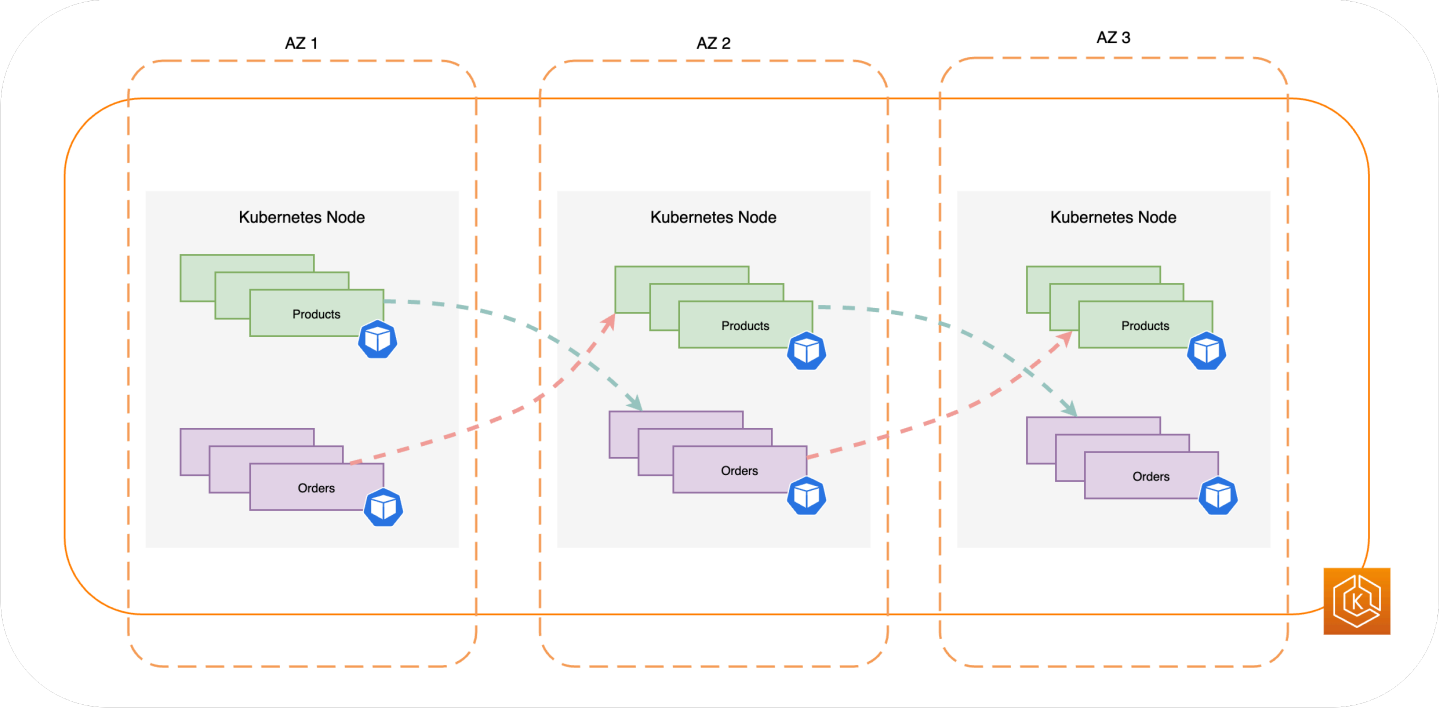
Pour réduire le risque de ces impacts négatifs potentiels, nous vous recommandons de surallouer la capacité de calcul de certains nœuds de travail de chacun des AZs. Ce faisant, le planificateur Kubernetes dispose d'une capacité préexistante pour le placement de nouveaux pods, ce qui est particulièrement important lorsque vous perdez l'un d'entre eux dans votre environnement. AZs

Exécuter et répartir plusieurs réplicas de pod entre les zones de disponibilité

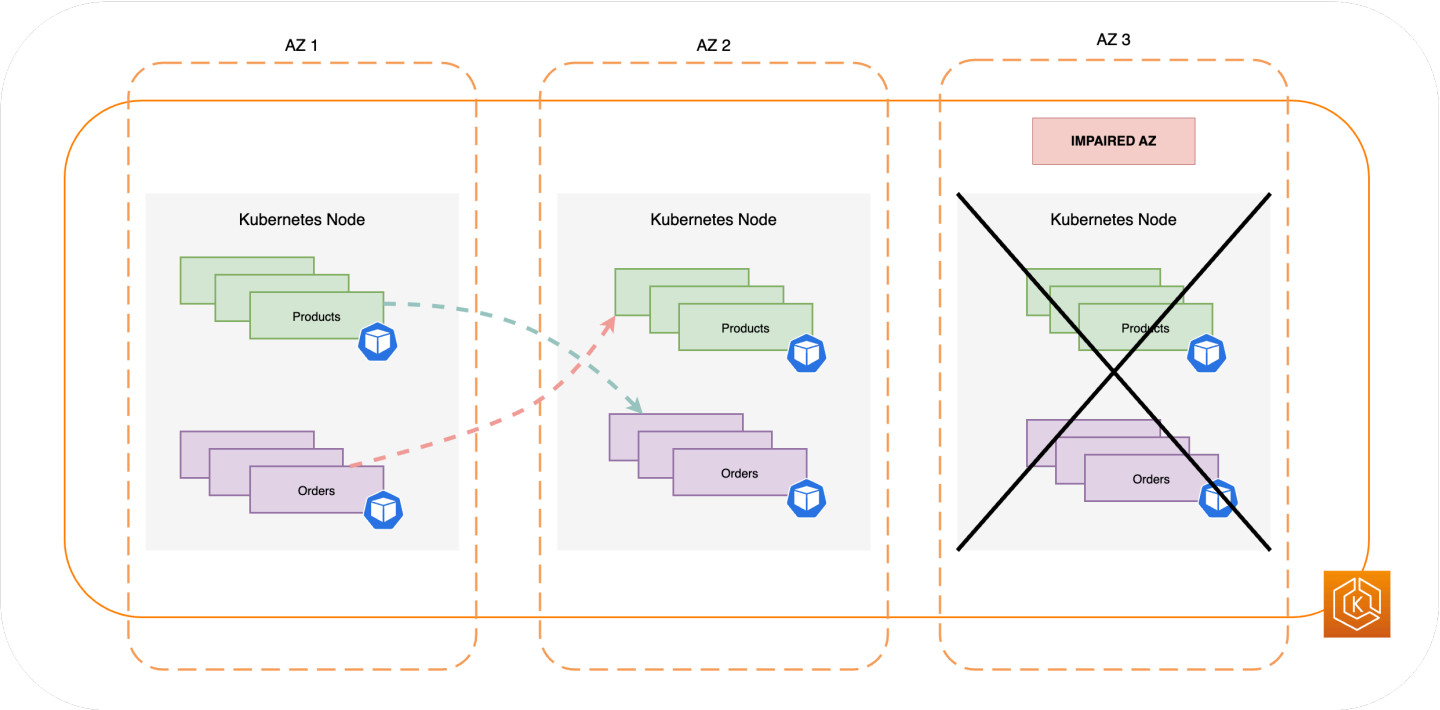
Kubernetes vous permet de mettre à l’échelle vos charges de travail à l’avance en exécutant plusieurs instances (réplicas de pod) d’une même application. L’exécution de plusieurs réplicas de pods pour une application élimine les points de défaillance uniques et augmente les performances globales en réduisant la pression sur les ressources d’un seul réplica. Toutefois, pour bénéficier à

la fois d'une haute disponibilité et d'une meilleure tolérance aux pannes pour vos applications, nous vous recommandons d'exécuter plusieurs réplicas de votre application et de répartir les réplicas sur différents domaines de défaillance, également appelés domaines topologiques. Dans ce scénario, les domaines de défaillance sont les zones de disponibilité. En utilisant des contraintes de répartition topologique, vous pouvez configurer vos applications pour qu'elles bénéficient d'une stabilité statique préexistante. Ensuite, en cas de défaillance de l'AZ, votre environnement disposera de suffisamment de répliques en bon état AZs pour gérer immédiatement les pics ou les pics de trafic.

Le schéma suivant illustre un environnement EKS où le east-to-west trafic est fluide lorsque tout le monde est AZs en bonne santé.



Le schéma suivant illustre un environnement EKS caractérisé par un flux de east-to-west trafic dans lequel un seul AZ est défaillant et où vous avez entamé un changement de zone.



L'extrait de code suivant est un exemple de configuration de votre charge de travail avec plusieurs réplicas dans Kubernetes.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: orders
spec:
  replicas: 9
  selector:
    matchLabels:
      app:orders
  template:
    metadata:
      labels:
        app: orders
        tier: backend
    spec:
      topologySpreadConstraints:
      - maxSkew: 1
        topologyKey: "topology.kubernetes.io/zone"
        whenUnsatisfiable: ScheduleAnyway
        labelSelector:
          matchLabels:
```

```
            app: orders
```

Il est essentiel d'exécuter plusieurs réplicas de votre logiciel de serveur DNS (CoreDNS/kube-dns) et d'appliquer des contraintes de répartition topologique similaires, si elles ne sont pas configurées par défaut. Cela permet de garantir qu'en cas de défaillance d'AZ unique, vous disposez de suffisamment de pods DNS sains pour continuer AZs à traiter les demandes de découverte de service pour les autres pods communicants du cluster. Le module complémentaire CoreDNS EKS possède des paramètres par défaut pour les modules CoreDNS qui garantissent que, si AZs plusieurs nœuds sont disponibles, ils sont répartis dans les zones de disponibilité de votre cluster. Si vous le souhaitez, vous pouvez remplacer ces paramètres par défaut par vos propres configurations personnalisées.

Lorsque vous installez CoreDNS avec Helm, vous pouvez mettre à jour le `replicaCount` dans le fichier values.yaml afin de vous assurer que vous disposez d'un nombre suffisant de réplicas dans chaque zone de disponibilité. En outre, pour vous assurer que ces répliques sont réparties entre les différents AZs environnements de votre cluster, veillez à mettre à jour la `topologySpreadConstraints` propriété dans le même `values.yaml` fichier. L'extrait de code suivant illustre comment vous pouvez configurer CoreDNS pour cela.

CoreDNS Helm values.yaml

```
replicaCount: 6
topologySpreadConstraints:
  - maxSkew: 1
    topologyKey: topology.kubernetes.io/zone
    whenUnsatisfiable: ScheduleAnyway
    labelSelector:
      matchLabels:
        k8s-app: kube-dns
```

En cas de défaillance d'une zone de disponibilité, vous pouvez absorber la charge accrue sur les pods CoreDNS en utilisant un système autoscaling pour CoreDNS. Le nombre d'instances DNS dont vous aurez besoin dépend du nombre de charges de travail exécutées dans votre cluster. CoreDNS est lié au CPU, ce qui lui permet de s'adapter en fonction du CPU à l'aide du Horizontal Pod Autoscaler (HPA). Voici un exemple que vous pouvez modifier en fonction de vos besoins.

```
apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
metadata:
  name: coredns
  namespace: default
```

```
spec:
  maxReplicas: 20
  minReplicas: 2
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: coredns
  targetCPUUtilizationPercentage: 50
```

EKS peut également gérer l'autoscaling du déploiement CoreDNS dans la version du module complémentaire EKS de CoreDNS. Cet autoscaler CoreDNS surveille en continu l'état du cluster, y compris le nombre de nœuds et le nombre de cœurs de processeur. Sur la base de ces informations, le contrôleur ajuste dynamiquement le nombre de réplicas du déploiement CoreDNS dans un cluster EKS.

Pour activer la configuration de l'autoscaling dans le module complémentaire CoreDNS EKS, utilisez le paramètre de configuration suivant :

```
{
  "autoScaling": {
    "enabled": true
  }
}
```

Vous pouvez également utiliser le NodeLocal DNS ou l'autoscaler proportionnel au cluster pour dimensionner CoreDNS. Pour plus d'informations, consultez Mise à l'échelle horizontale de CoreDNS.

Colocaliser les pod interdépendants dans la même zone de disponibilité

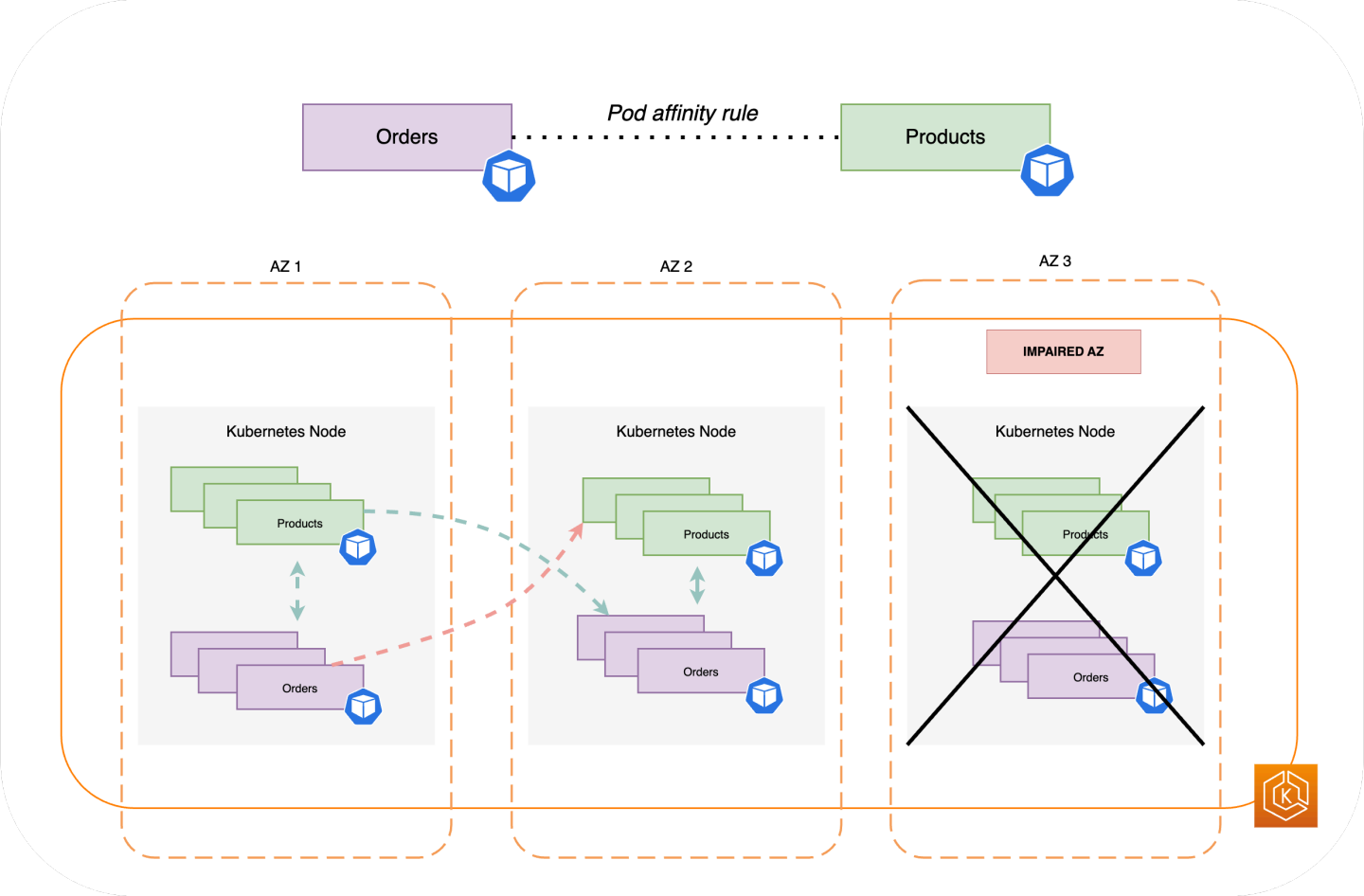
Généralement, les applications ont des charges de travail distinctes qui doivent communiquer entre elles pour mener à bien un end-to-end processus. Si ces applications distinctes sont réparties entre différentes zones AZs et ne sont pas hébergées dans la même zone, une seule altération de la zone Z peut avoir un impact sur le end-to-end processus. Par exemple, si l'application A possède plusieurs répliques en AZ 1 et AZ 2, mais que l'application B possède toutes ses répliques en AZ 3, la perte de l'AZ 3 affectera les end-to-end processus entre les deux charges de travail, l'application A et l'application B. Si vous combinez les contraintes de dispersion topologique avec l'affinité des pods, vous pouvez améliorer la résilience de votre application en répartissant les pods sur l'ensemble du site. AZs De plus, cela permet de configurer une relation entre certains pods afin de garantir leur colocalisation.

Grâce aux règles d'affinité des pods, vous pouvez définir des relations entre les charges de travail afin d'influencer le comportement du planificateur Kubernetes, de manière à ce qu'il effectue la colocalisation des pods sur le même nœud de travail ou dans la même AZ. Vous pouvez également configurer le niveau de rigueur des contraintes de planification.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: products
  namespace: ecommerce
  labels:
    app.kubernetes.io/version: "0.1.6"

    spec:
      serviceAccountName: graphql-service-account
      affinity:
        podAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
          - labelSelector:
              matchExpressions:
              - key: app
                operator: In
                values:
                - orders
            topologyKey: "kubernetes.io/hostname"
```

Le diagramme suivant montre plusieurs pods que le planificateur a colocalisés sur le même nœud à l'aide des règles d'affinité des pods.



Vérifier que votre environnement de cluster peut gérer la perte d'une zone de disponibilité

Une fois que vous avez rempli les conditions décrites dans les sections précédentes, l'étape suivante consiste à vérifier que vous disposez d'une capacité de calcul et de charge de travail suffisante pour gérer la perte d'une zone de disponibilité. Pour ce faire, vous pouvez lancer manuellement un changement de zone dans EKS. Vous pouvez également activer le transfert automatique de zone et configurer des essais, qui permettent également de vérifier que vos applications fonctionnent comme prévu avec une zone de disponibilité en moins dans votre environnement de cluster.

Questions fréquentes (FAQ)

Pourquoi utiliser cette fonctionnalité ?

En utilisant le changement de zone ARC ou le transfert automatique de zone dans votre cluster EKS, vous pouvez mieux maintenir la disponibilité des applications Kubernetes en automatisant le processus de récupération rapide consistant à transférer le trafic réseau du cluster hors d'une zone

de disponibilité défaillante. Avec ARC, vous pouvez éviter les étapes longues et compliquées qui peuvent entraîner une période de récupération prolongée lors d’événements de défaillance d’une zone de disponibilité.

Comment fonctionne cette fonctionnalité avec les autres AWS services ?

EKS s'intègre à ARC, qui fournit l'interface principale dans laquelle vous pouvez effectuer des opérations de restauration AWS. Afin de garantir que le routage du trafic au sein du cluster soit correctement acheminé hors d’une zone de disponibilité défaillante, EKS modifie la liste des points de terminaison réseau pour les pods s’exécutant dans le plan de données Kubernetes. Si vous utilisez Elastic Load Balancing pour le routage du trafic externe vers le cluster, vous pouvez enregistrer vos équilibreurs de charge auprès d’ARC et lancer un changement de zone sur ceux-ci afin d’empêcher le trafic d’affluer vers la zone de disponibilité dégradée. Zonal Shift fonctionne également avec les groupes Amazon EC2 Auto Scaling créés par des groupes de nœuds gérés par EKS. Afin d’empêcher qu’une zone de disponibilité défaillante ne soit utilisée pour de nouveaux pods Kubernetes ou le lancement de nouveaux nœuds, EKS supprime la zone de disponibilité défaillante des groupes Auto Scaling.

En quoi cette fonctionnalité diffère-t-elle des protections Kubernetes par défaut ?

Cette fonctionnalité fonctionne en tandem avec plusieurs protections intégrées à Kubernetes qui contribuent à la résilience des applications des clients. Vous pouvez configurer des sondes de disponibilité et de vitalité des pods qui déterminent quand un pod doit prendre en charge le trafic. Lorsque ces sondes échouent, Kubernetes supprime ces pods en tant que cibles pour un service, et le trafic n’est plus envoyé au pod. Bien que cela soit utile, il n’est pas simple pour les clients de configurer ces surveillances de l’état de manière à garantir leur échec lorsqu’une AZ est dégradée. La fonctionnalité de changement de zone ARC fournit un filet de sécurité supplémentaire qui vous aide à isoler entièrement une AZ dégradée lorsque les protections natives de Kubernetes ne sont pas suffisantes. Le changement de zone vous offre également un moyen simple de tester la disponibilité opérationnelle et la résilience de votre architecture.

Puis-je AWS commencer un changement de zone en mon nom ?

Oui, si vous voulez utiliser le changement de zone ARC de manière entièrement automatisée, vous pouvez activer le changement automatique de zone ARC. Avec le changement automatique de zone, vous pouvez compter sur vous AWS pour surveiller l'état de santé de votre cluster EKS et AZs pour démarrer automatiquement un changement de zone lorsqu'une altération de l'AZ est détectée.

Que se passe-t-il si j’utilise cette fonctionnalité et que mes composants master et mes charges de travail ne sont pas mis à l’échelle au préalable ?

Si vous n'avez pas effectué de mise à l'échelle préalable et que vous comptez sur le provisionnement de nœuds ou de pod supplémentaires pendant un changement de zone, vous risquez un retard dans la récupération. Le processus d'ajout de nouveaux nœuds au plan de données Kubernetes prend un certain temps, ce qui peut avoir un impact sur les performances en temps réel et la disponibilité de vos applications, en particulier en cas de défaillance de zone. En outre, en cas d'altération de la zone, vous pouvez rencontrer une contrainte de capacité de calcul potentielle qui pourrait empêcher l'ajout de nouveaux nœuds au nœud sain AZs.

Si vos charges de travail ne sont pas prédimensionnées et réparties sur l'ensemble AZs de votre cluster, une altération zonale peut avoir un impact sur la disponibilité d'une application exécutée uniquement sur les nœuds de travail dans une zone de zone affectée. Afin d'atténuer le risque d'une interruption complète de la disponibilité de votre application, EKS dispose d'un dispositif de sécurité qui permet d'envoyer le trafic vers les points de terminaison des pods dans une zone défaillante si cette charge de travail a tous ses points de terminaison dans la zone de disponibilité défaillante. Cependant, nous vous recommandons vivement de pré-dimensionner et de répartir vos applications sur l'ensemble du AZs site afin de maintenir la disponibilité en cas de problème de zone.

Comment cela fonctionne-t-il si j'exécute une application avec état ?

Si vous exécutez une application avec état, vous devez évaluer sa tolérance aux pannes en fonction de votre cas d'utilisation et de votre architecture. Si vous avez une active/standby architecture ou un modèle, il se peut que l'actif se trouve dans un AZ altéré. Au niveau de l'application, si l'instance de secours n'est pas activée, vous risquez de rencontrer des problèmes avec votre application. Vous pouvez également rencontrer des problèmes lorsque de nouveaux pods Kubernetes sont lancés en bon état AZs, car ils ne pourront pas se connecter aux volumes persistants liés à l'AZ altérée.

Cette fonctionnalité est-elle compatible avec Karpenter ?

La prise en charge de Karpenter n'est actuellement pas disponible avec le changement de zone ARC et le changement automatique de zone dans EKS. Si un AZ est altéré, vous pouvez ajuster la NodePool configuration Karpenter correspondante en supprimant l'AZ défectueux afin que les nouveaux nœuds de travail ne soient lancés que dans l'autre AZs.

Cette fonctionnalité est-elle compatible avec EKS Fargate ?

Cette fonctionnalité n'est pas compatible avec EKS Fargate. Par défaut, lorsqu'EKS Fargate reconnaît un événement sanitaire dans une zone, Pods préfère fonctionner dans l'autre zone. AZs

Le plan de contrôle Kubernetes géré par Amazon EKS sera-t-il affecté ?

Non, par défaut, Amazon EKS exécute et adapte le plan de contrôle Kubernetes sur plusieurs plans afin de garantir une AZs haute disponibilité. Le changement de zone ARC et le déplacement automatique de zone n'agissent que sur le plan de données Kubernetes.

Cette nouvelle fonctionnalité entraîne-t-elle des coûts supplémentaires ?

Vous pouvez utiliser le changement de zone ARC et le changement automatique de zone dans votre cluster EKS sans frais supplémentaires. Cependant, vous continuerez à payer les instances provisionnées et nous vous recommandons vivement de mettre à l'échelle au préalable votre plan de données Kubernetes avant d'utiliser cette fonctionnalité. Il est important de trouver un équilibre entre le coût et la disponibilité des applications.

Ressources supplémentaires

- Comment fonctionne un changement de zone
- Bonnes pratiques pour le changement de zone dans ARC
- Ressources et scénarios pris en charge pour le changement de zone et le changement automatique de zone
- Exécution de charges de travail résilientes sur Amazon EKS
- Élimination du décalage de mise à l'échelle des nœuds Kubernetes grâce à la priorité des pods et au surprovisionnement
- Mise à l'échelle des pods CoreDNS pour un trafic DNS élevé

Activer le changement de zone EKS pour éviter les zones de disponibilité défaillantes

Amazon Application Recovery Controller (ARC) vous aide à gérer et à coordonner la restauration de vos applications dans toutes les zones de disponibilité (AZs) et fonctionne avec de nombreux services, dont Amazon EKS. Grâce à la prise en charge par EKS du changement de zone ARC, vous pouvez déplacer le trafic réseau au sein du cluster hors d'une AZ défaillante. Vous pouvez également autoriser AWS à surveiller l'état de votre AZ en mauvais état AZs et à transférer temporairement le trafic réseau hors de portée en votre nom.

Comment utiliser le changement de zone EKS :

1. Activez votre cluster EKS avec le contrôleur de récupération d'application (ARC) Amazon. Cela se fait au niveau du cluster à l'aide de la console Amazon EKS, de la AWS CLI ou de eksctl. CloudFormation
2. Une fois activé, vous pouvez gérer les changements de zone ou les changements automatiques de zone à l'aide de la console ARC, de la AWS CLI ou du décalage zonal et du décalage automatique de zone. APIs

Veuillez noter qu'après avoir enregistré un cluster EKS avec ARC, vous devez encore configurer ARC. Par exemple, vous pouvez utiliser la console ARC pour configurer le changement de zone.

Pour plus d'informations sur le fonctionnement du changement de zone EKS et sur la conception de vos charges de travail pour gérer les zones de disponibilité défaillantes, consultez the section called "En savoir plus sur le changement de zone".

Considérations

- Le mode automatique EKS ne prend pas en charge le contrôleur de récupération d'application Amazon, le changement de zone et le déplacement automatique de zone.
- Nous vous recommandons d'attendre au moins 60 secondes entre chaque opération de changement de zone afin de garantir le traitement correct de chaque demande.

  Si vous essayez d'effectuer des changements de zone successifs (à moins de 60 secondes d'intervalle), le service Amazon EKS risque de ne pas traiter correctement toutes les demandes de changement. Cela est dû au mécanisme de sondage actuel qui met à jour l'état de zone du cluster. Si vous devez effectuer plusieurs changements de zone, veillez à laisser suffisamment de temps entre les opérations pour que le système puisse traiter chaque changement.

Qu'est-ce qu'Amazon Application Recovery Controller ?

Le contrôleur de récupération d'application (ARC) Amazon vous aide à vous préparer et à accélérer la récupération des applications exécutées sur AWS. Le changement de zone vous permet de vous remettre rapidement en cas de défaillance d'une zone de disponibilité (AZ), en déplaçant temporairement le trafic d'une ressource prise en charge d'une zone Z vers un trafic normal AZs dans la AWS région.

En savoir plus sur le contrôleur de récupération d'application (ARC) Amazon

Qu'est-ce que le changement de zone ?

Le changement de zone est une fonctionnalité d'ARC qui vous permet de déplacer le trafic vers une ressource telle qu'un cluster EKS ou un Elastic Load Balancer hors d'une zone de disponibilité d' AWS une région afin d'atténuer rapidement un problème et de récupérer rapidement votre application. Vous pouvez choisir de déplacer le trafic, par exemple, parce qu'un mauvais déploiement entraîne des problèmes de latence ou parce que la zone de disponibilité est défaillante. Un changement de zone ne nécessite aucune configuration préalable.

En savoir plus sur le changement de zone ARC

Qu'est-ce que le changement de zone autoshift ?

L'autoshift zonal est une fonctionnalité d'ARC que vous pouvez activer pour autoriser AWS le transfert du trafic d'un AZ pour les ressources prises en charge, en votre nom, vers un trafic sain AZs dans la AWS région. AWS lance un changement automatique lorsque la télémétrie interne indique la présence d'une anomalie dans un AZ d'une région susceptible d'avoir un impact sur les clients. La télémétrie interne intègre des métriques provenant de plusieurs sources, notamment le AWS réseau et les services Amazon EC2 et Elastic Load Balancing.

AWS met fin aux changements automatiques lorsque les indicateurs indiquent qu'il n'y a plus de problème ou de problème potentiel.

En savoir plus sur le changement automatique de zone ARC

Que fait EKS pendant un changement automatique ?

EKS met à jour les configurations réseau pour éviter de rediriger le trafic vers des zones altérées AZs. De plus, si vous utilisez des groupes de nœuds gérés, EKS ne lancera de nouveaux nœuds dans les nœuds sains que AZs lors d'un changement de zone. Lorsque le changement de zone expire ou est annulé, les configurations réseau seront restaurées pour inclure l'AZ qui avait été précédemment détectée comme non fonctionnelle.

En savoir plus sur le changement de zone EKS.

Enregistrer le cluster EKS auprès d'Amazon Application Recovery Controller (ARC) (AWS console)

1. Recherchez le nom et la région du cluster EKS que vous voulez enregistrer avec ARC.

2. Accédez à la console EKS dans cette région et sélectionnez votre cluster.
3. Sur la page Informations sur le cluster, sélectionnez l'onglet Vue d'ensemble.
4. Sous l'en-tête Changement de zone, sélectionnez le bouton Gérer.
5. Sélectionnez activer ou désactiver pour Changement de zone EKS.

Votre cluster EKS est désormais enregistré auprès d'ARC.

Si vous souhaitez AWS détecter et éviter les zones de disponibilité altérées, vous devez configurer l'autoshift par zone ARC. Vous pouvez par exemple le faire dans la console ARC.

Étapes suivantes

- Découvrez comment activer le changement automatique de zone.
- Découvrez comment lancer manuellement un changement de zone.

Présentation du fonctionnement du contrôle d'accès dans Amazon EKS

Découvrez comment gérer l'accès à votre cluster Amazon EKS. L'utilisation d'Amazon EKS nécessite de connaître la manière dont Kubernetes et AWS Identity and Access Management (AWS IAM) gèrent le contrôle d'accès.

Contenu de cette section :

the section called "Accès à l'API Kubernetes" : découvrez comment permettre à des applications ou des utilisateurs de s'authentifier auprès de l'API Kubernetes. Vous pouvez utiliser des entrées d'accès, l'aws-auth ou un ConfigMap fournisseur OIDC externe.

the section called "Accéder aux ressources du cluster"— Découvrez comment configurer le AWS Management Console pour communiquer avec votre cluster Amazon EKS. Utilisez la console pour consulter les ressources Kubernetes du cluster, telles que les espaces de noms, les nœuds et les pods.

the section called "Modifier les ressources du cluster"— Découvrez les autorisations requises pour modifier les ressources Kubernetes.

the section called "Accès au cluster avec kubectl" : découvrez comment configurer kubectl pour communiquer avec votre cluster Amazon EKS. Utilisez la AWS CLI pour créer un fichier kubeconfig.

the section called "Accès des charges de travail à AWS "— Découvrez comment associer un compte de service Kubernetes à IAM Roles AWS . Vous pouvez utiliser les rôles d'identité du pod ou IAM pour les comptes de service (IRSA).

Tâches courantes

- Accordez aux développeurs l'accès à l'API Kubernetes. Consultez les ressources Kubernetes dans le. AWS Management Console
  - Solution : utilisez des entrées d'accès pour associer les autorisations Kubernetes RBAC aux utilisateurs ou aux rôles AWS IAM.
- Configurez kubectl pour communiquer avec un cluster Amazon EKS à l'aide AWS des informations d'identification.

  - Solution : utilisez la AWS CLI pour créer un fichier kubeconfig.
- Utilisez un fournisseur d'identité externe, comme Ping Identity, pour authentifier les utilisateurs auprès de l'API Kubernetes.
  - Solution : associez un fournisseur OIDC externe.
- Donnez aux charges de travail de votre cluster Kubernetes la possibilité d'appeler. AWS APIs
  - Solution : utilisez Pod Identity pour associer un rôle AWS IAM à un compte de service Kubernetes.

Contexte

- Découvrez comment fonctionnent les comptes de service Kubernetes.
- Examen du modèle de contrôle d'accès basé sur les rôles (RBAC) de Kubernetes
- Pour plus d'informations sur la gestion de l'accès aux AWS ressources, consultez le guide de l'utilisateur AWS IAM. Vous pouvez également suivre une formation d'introduction gratuite sur l'utilisation d' AWS IAM.

Considérations relatives au mode automatique EKS

Le mode automatique EKS s'intègre à l'identité du pod EKS ainsi qu'aux entrées d'accès EKS.

- Le mode automatique EKS utilise les entrées d'accès pour accorder au plan de contrôle EKS des autorisations Kubernetes. Par exemple, les stratégies d'accès permettent au mode automatique EKS de lire les informations concernant les points de terminaison réseau et les services.
  - Vous ne pouvez pas désactiver les entrées d'accès sur un cluster du mode automatique EKS.
  - Vous pouvez éventuellement activer le `ConfigMap` `aws-auth`.
  - Les entrées d'accès pour le mode automatique EKS sont configurées automatiquement. Vous pouvez consulter ces entrées d'accès, mais vous ne pouvez pas les modifier.
  - Si vous utilisez un NodeClass pour créer un rôle Node IAM personnalisé, vous devez créer une entrée d'accès pour le rôle conformément à la politique EKSAuto NodePolicy d'accès Amazon.
- Si vous souhaitez accorder des autorisations aux charges de travail pour les AWS services, utilisez EKS Pod Identity.
  - Il n'est pas nécessaire d'installer l'agent d'identité du pod sur les clusters du mode automatique EKS.

Accorder aux utilisateurs et aux rôles IAM l'accès à Kubernetes APIs

Votre cluster dispose d'un point de terminaison API Kubernetes. Kubectl utilise cette API. Vous pouvez vous authentifier auprès de cette API à l'aide de deux types d'identités :

- Un responsable de la gestion des AWS identités et des accès (IAM) (rôle ou utilisateur) : ce type nécessite une authentification auprès d'IAM. Les utilisateurs peuvent se connecter en AWS tant qu'utilisateur IAM ou avec une identité fédérée en utilisant les informations d'identification fournies par le biais d'une source d'identité. Les utilisateurs ne peuvent se connecter qu'avec une identité fédérée si votre administrateur a précédemment configuré une fédération d'identité avec des rôles IAM. Lorsque les utilisateurs accèdent AWS par le biais de la fédération, ils assument indirectement un rôle. Lorsque les utilisateurs utilisent ce type d'identité :
  - Vous pouvez leur attribuer des autorisations Kubernetes afin qu'ils puissent travailler avec les objets Kubernetes sur votre cluster. Pour plus d'informations sur la manière d'attribuer des autorisations à vos principaux IAM afin qu'ils puissent accéder aux objets Kubernetes sur votre cluster, consultez the section called "Entrées d'accès".
  - Vous pouvez leur attribuer des autorisations IAM afin qu'ils puissent travailler avec votre cluster Amazon EKS et ses ressources à l'aide de l'API, de la AWS CLI AWS CloudFormation AWS Management Console, ou `eksctl` d'Amazon EKS. Pour plus d'informations, consultez la rubrique Actions définies par Amazon Elastic Kubernetes Service dans la Référence des autorisations de service.
  - Les nœuds rejoignent votre cluster en endossant un rôle IAM. La possibilité d'accéder à votre cluster à l'aide des principaux IAM est fournie par l'authentificateur AWS IAM pour Kubernetes, qui s'exécute sur le plan de contrôle Amazon EKS.
- Un utilisateur de votre propre fournisseur OpenID Connect (OIDC) : ce type nécessite une authentification auprès de votre fournisseur OIDC. Pour plus d'informations sur la configuration de votre propre fournisseur OIDC avec votre cluster Amazon EKS, consultez the section called "Association d'un fournisseur OIDC externe". Lorsque les utilisateurs utilisent ce type d'identité :
  - Vous pouvez leur attribuer des autorisations Kubernetes afin qu'ils puissent travailler avec les objets Kubernetes sur votre cluster.
  - Impossible de leur attribuer des autorisations IAM afin qu'ils puissent travailler avec votre cluster Amazon EKS et ses ressources à l'aide de l'API, de la AWS CLI ou `eksctl` de Amazon EKS. AWS CloudFormation AWS Management Console

Vous pouvez utiliser les deux types d'identités avec votre cluster. La méthode d'authentification IAM ne peut pas être désactivée. La méthode d'authentification OIDC est facultative.

Associer les identités IAM aux autorisations Kubernetes

L'authentificateur AWS IAM pour Kubernetes est installé sur le plan de contrôle de votre cluster. Il active les principaux de gestion des identités et des accès (IAM)AWS (rôles et utilisateurs) que vous autorisez à accéder aux ressources Kubernetes sur votre cluster. Vous pouvez autoriser les principaux IAM à accéder aux objets Kubernetes sur votre cluster à l'aide de l'une des méthodes suivantes :

- Création d'entrées d'accès : si votre cluster est à la version de plate-forme indiquée dans la section Conditions préalables pour la version Kubernetes de votre cluster ou à une version ultérieure, nous vous recommandons d'utiliser cette option.

  Utilisez les entrées d'accès pour gérer les autorisations Kubernetes des principaux IAM depuis l'extérieur du cluster. Vous pouvez ajouter et gérer l'accès au cluster à l'aide de l'API EKS, de l'interface de ligne de AWS commande AWS SDKs AWS CloudFormation,, et AWS Management Console. Cela signifie que vous pouvez gérer les utilisateurs avec les mêmes outils que ceux avec lesquels vous avez créé le cluster.

  Pour commencer, suivez Changer le mode d'authentification pour utiliser les entrées d'accès, puis Migration des entrées aws-auth existantes pour accéder aux ConfigMap entrées.
- Ajouter des entrées au **`aws-auth` `ConfigMap`** : si la version de plateforme de votre cluster est antérieure à celle indiquée dans la section Prérequis, vous devez utiliser cette option. Si la version de plateforme de votre cluster est égale ou supérieure à la version de plateforme indiquée dans la section Conditions préalables pour la version Kubernetes de votre cluster, et que vous avez ajouté des entrées au fichier `ConfigMap`, nous vous recommandons de migrer ces entrées vers les entrées d'accès. Vous ne pouvez toutefois pas migrer les entrées qu'Amazon EKS a ajoutées à la `ConfigMap`, telles que les entrées pour les rôles IAM utilisés avec des groupes de nœuds gérés ou des profils Fargate. Pour de plus amples informations, veuillez consulter the section called "Accès à l'API Kubernetes".
  - Si vous devez utiliser l'option `ConfigMap` `aws-auth`, vous pouvez ajouter des entrées à la `ConfigMap` à l'aide de la commande `eksctl create iamidentitymapping`. Pour plus d'informations, consultez la section Gérer les utilisateurs et les rôles IAM dans la documentation `eksctl`.

Définition du mode d'authentification du cluster

Chaque cluster possède un mode d'authentification. Le mode d'authentification détermine les méthodes que vous pouvez utiliser pour autoriser les principaux IAM à accéder aux objets Kubernetes de votre cluster. Il existe trois modes d'authentification.

> **Important**
>
> Une fois la méthode d'accès activée, elle ne peut plus être désactivée.
> Si la méthode `ConfigMap` n'est pas activée lors de la création du cluster, elle ne peut pas être activée ultérieurement. Tous les clusters créés avant l'introduction des entrées d'accès ont cette méthode `ConfigMap` activée.
> Si vous utilisez des nœuds hybrides avec votre cluster, vous devez utiliser les modes d'authentification ou cluster `API` ou `API_AND_CONFIG_MAP`.

Le **`aws-auth` `ConfigMap`** à l'intérieur du cluster

Il s'agit du mode d'authentification d'origine pour les clusters Amazon EKS. Le principal IAM qui a créé le cluster est l'utilisateur initial à avoir accès au cluster au moyen de `kubectl`. L'utilisateur initial doit ajouter d'autres utilisateurs à la liste dans la `ConfigMap` `aws-auth` et attribuer des autorisations qui affectent les autres utilisateurs au sein du cluster. Ces autres utilisateurs ne peuvent pas gérer ou supprimer l'utilisateur initial, car il n'y a aucune entrée à gérer dans la `ConfigMap`.

À la fois le **`ConfigMap`** et les entrées d'accès

Avec ce mode d'authentification, vous pouvez utiliser les deux méthodes pour ajouter des principaux IAM au cluster. Notez que chaque méthode stocke des entrées distinctes ; par exemple, si vous ajoutez une entrée d'accès depuis la AWS CLI, elle n'`aws-authConfigMap`est pas mise à jour.

Accès aux entrées uniquement

Avec ce mode d'authentification, vous pouvez utiliser l'API EKS, l'interface de ligne de AWS commande AWS SDKs, AWS CloudFormation, et AWS Management Console pour gérer l'accès au cluster pour les principaux IAM.

Chaque entrée d'accès possède un type et vous pouvez utiliser la combinaison d'une portée d'accès pour limiter le principal à un espace de noms spécifique et d'une stratégie d'accès pour définir des politiques d'autorisations réutilisables préconfigurées. Vous pouvez également

utiliser le type STANDARD et les groupes RBAC Kubernetes pour attribuer des autorisations personnalisées.

Mode d'authentification	Méthodes
`ConfigMap` uniquement (`CONFIG_MAP`)	`aws-auth ConfigMap`
API EKS et `ConfigMap` (`API_AND_CONFIG_MAP`)	accéder aux entrées dans l'API EKS, l'interface de ligne de AWS commande AWS SDKs AWS CloudFormation,, AWS Management Console et `aws-auth ConfigMap`
API EKS uniquement (`API`)	accéder aux entrées de l'API EKS, de l'interface de ligne de AWS commande AWS SDKs AWS CloudFormation, et AWS Management Console

> **Note**
>
> Le mode automatique Amazon EKS nécessite des entrées d'accès.

Attribution de l'accès Kubernetes aux utilisateurs IAM avec les entrées d'accès EKS

Cette section explique comment gérer l'accès des principaux IAM aux clusters Kubernetes dans Amazon Elastic Kubernetes Service (EKS) en utilisant les entrées d'accès et des politiques d'accès. Vous y trouverez des informations sur la modification des modes d'authentification, la migration depuis les entrées `aws-auth` ConfigMap héritées, la création, la mise à jour et la suppression d'entrées d'accès, l'association de politiques aux entrées, la consultation des autorisations des politiques prédéfinies, et les prérequis et les considérations clés pour une gestion sécurisée des accès.

Présentation

Les entrées d'accès EKS constituent la meilleure méthode pour accorder aux utilisateurs l'accès à l'API Kubernetes. Par exemple, vous pouvez utiliser les entrées d'accès pour permettre aux

développeurs d'utiliser kubectl. Concrètement, une entrée d'accès EKS associe un ensemble d'autorisations Kubernetes à une identité IAM, comme un rôle IAM. Par exemple, un développeur peut assumer un rôle IAM et l'utiliser pour s'authentifier auprès d'un cluster EKS.

Fonctionnalités

- Authentification et autorisation centralisées : contrôle l'accès aux clusters Kubernetes directement via les API Amazon EKS, supprimant la nécessité d'alterner entre les API AWS et les API Kubernetes pour gérer les autorisations des utilisateurs.
- Gestion granulaire des autorisations : utilise des entrées d'accès et des politiques pour définir des permissions fines pour les principaux AWS IAM, notamment la possibilité de modifier ou de retirer l'accès cluster-admin au créateur du cluster.
- Intégration aux outils IaC : prend en charge les outils d'infrastructure en tant que code, tels que AWS CloudFormation, Terraform et AWS CDK, pour définir la configuration des accès lors de la création du cluster.
- Restauration après une mauvaise configuration : permet de restaurer l'accès au cluster via l'API Amazon EKS sans nécessiter d'accès direct à l'API Kubernetes.
- Réduction des frais généraux et sécurité améliorée : centralise les opérations afin de réduire les frais généraux tout en tirant parti des fonctionnalités offertes AWS IAM, telles que la journalisation via CloudTrail et l'authentification multifacteur.

Attachement des autorisations

Vous pouvez attacher des autorisations Kubernetes aux entrées d'accès de deux façons :

- Utiliser une stratégie d'accès. Les stratégies d'accès sont des modèles prédéfinis d'autorisations Kubernetes, gérés par AWS. Pour de plus amples informations, consultez the section called "Vérification des stratégies d'accès".
- Référencer un groupe Kubernetes. Si vous associez une identité IAM à un groupe Kubernetes, vous pouvez créer des ressources Kubernetes qui accordent des autorisations à ce groupe. Pour plus d'informations, consultez Utilisation de l'autorisation RBAC dans la documentation Kubernetes.

Considérations

Lorsque vous activez des entrées d'accès EKS pour des clusters existants, gardez à l'esprit les points suivants :

- Comportement des clusters existants : pour les clusters créés avant l'introduction des entrées d'accès (ceux dont la version de plateforme initiale est antérieure à celle indiquée dans Exigences de version de plateforme), EKS crée automatiquement une entrée d'accès reflétant les autorisations préexistantes. Cette entrée inclut l'identité IAM qui a créé le cluster, ainsi que les autorisations administratives accordées lors de la création du cluster.
- Gestion du ConfigMap **`aws-auth`** hérité : si votre cluster utilise encore le ConfigMap `aws-auth` hérité pour la gestion des accès, seule l'entrée d'accès correspondant au créateur initial du cluster est automatiquement créée lors de l'activation des entrées d'accès. Les rôles ou autorisations supplémentaires ajoutés au ConfigMap (par exemple, rôles IAM personnalisés pour des développeurs ou des services) ne sont pas migrés automatiquement. Pour résoudre ce problème, créez manuellement les entrées d'accès correspondantes.

Mise en route

1. Déterminez l'identité IAM et la stratégie d'accès que vous souhaitez utiliser.
   - the section called "Vérification des stratégies d'accès"
2. Activez les entrées d'accès EKS sur votre cluster. Vérifiez que vous disposez d'une version de plateforme prise en charge.
   - the section called "Mode d'authentification"
3. Créez une entrée d'accès qui associe une identité IAM à une autorisation Kubernetes.
   - the section called "Créer des entrées d'accès"
4. Authentifiez-vous auprès du cluster à l'aide de l'identité IAM.
   - the section called "Configuration de l'interface de ligne de commande AWS"
   - the section called "Configurer kubectl et eksctl"

Association des stratégies d'accès aux entrées d'accès

Vous pouvez attribuer une ou plusieurs stratégies d'accès à des entrées d'accès de type `STANDARD`. Amazon EKS accorde automatiquement aux autres types d'entrées d'accès les autorisations nécessaires pour fonctionner correctement dans votre cluster. Les stratégies d'accès Amazon EKS contiennent des autorisations Kubernetes et non des autorisations IAM. Avant d'associer une stratégie d'accès à une entrée d'accès, assurez-vous de bien comprendre les autorisations Kubernetes incluses dans chaque stratégie d'accès. Pour de plus amples informations, consultez the section called "Vérification des stratégies d'accès". Si aucune stratégie d'accès ne correspond à vos

besoins, n'associez pas de stratégie d'accès à l'entrée d'accès. À la place, indiquez un ou plusieurs noms de groupes pour l'entrée d'accès, puis créez et gérez les objets Kubernetes de contrôle d'accès basé sur les rôles. Pour de plus amples informations, consultez the section called "Créer des entrées d'accès".

- Une entrée d'accès existante. Pour en créer un, consultez the section called "Créer des entrées d'accès".
- Un rôle ou un utilisateur de gestion des identités et des accès AWS disposant des autorisations suivantes : `ListAccessEntries`, `DescribeAccessEntry`, `UpdateAccessEntry`, `ListAccessPolicies`, `AssociateAccessPolicy` et `DisassociateAccessPolicy`. Pour plus d'informations, consultez Actions définies par Amazon Elastic Kubernetes Service dans la Référence des autorisations de service.

Avant d'associer des stratégies d'accès à des entrées d'accès, tenez compte des exigences suivantes :

- Vous pouvez associer plusieurs stratégies d'accès à chaque entrée d'accès, mais vous ne pouvez associer chaque politique à une entrée d'accès qu'une seule fois. Si vous associez plusieurs stratégies d'accès, le principal IAM lié à l'entrée d'accès dispose de toutes les autorisations incluses dans ces stratégies d'accès.
- Vous pouvez étendre une stratégie d'accès à toutes les ressources d'un cluster ou en spécifiant le nom d'un ou de plusieurs espaces de noms Kubernetes. Vous pouvez utiliser des caractères génériques pour le nom d'un espace de noms. Par exemple, si vous souhaitez étendre une stratégie d'accès à tous les espaces de noms commençant par `dev-`, vous pouvez spécifier `dev-*` sous forme de nom d'espace de noms. Assurez-vous que les espaces de noms existent sur votre cluster et que votre orthographe correspond au nom réel de l'espace de noms du cluster. Amazon EKS ne vérifie ni l'orthographe, ni l'existence des espaces de noms indiqués dans votre cluster.
- Vous pouvez modifier la portée d'accès d'une stratégie d'accès après l'avoir associée à une entrée d'accès. Si vous avez limité la stratégie d'accès aux espaces de noms Kubernetes, vous pouvez ajouter ou supprimer des espaces de noms de l'association selon les besoins.
- Si vous associez une stratégie d'accès à une entrée d'accès dont les noms de groupe sont également spécifiés, le principal IAM dispose alors de toutes les autorisations dans toutes les stratégies d'accès associées. Il dispose également de toutes les autorisations dans n'importe quel objet Kubernetes `Role` ou `ClusterRole` spécifié dans n'importe quels objets Kubernetes `Role` et `RoleBinding` qui spécifient les noms de groupes.

- Si vous exécutez la commande `kubectl auth can-i --list`, les autorisations accordées par les stratégies d'accès associées à une entrée d'accès n'apparaîtront pas pour le principal IAM utilisé lors de l'exécution de la commande. La commande n'affiche les autorisations Kubernetes que si vous les avez accordées dans des objets Kubernetes `Role` ou `ClusterRole` que vous avez liés aux noms de groupes ou au nom d'utilisateur spécifiés pour une entrée d'accès.
- Si vous usurpez l'identité d'un utilisateur ou groupe Kubernetes lorsque vous interagissez avec les objets Kubernetes de votre cluster, par exemple en utilisant la commande `kubectl` avec `--as` *`username`* ou `--as-group` *`group-name`* , vous forcez l'utilisation du mécanisme d'autorisation RBAC Kubernetes. Par conséquent, le principal IAM ne dispose d'aucune autorisation attribuée par les stratégies d'accès associées à l'entrée d'accès. Les seules autorisations Kubernetes dont dispose l'utilisateur ou le groupe dont l'identité est usurpée par le principal IAM sont celles que vous lui avez accordées dans les objets Kubernetes `Role` ou `ClusterRole` liés aux noms de groupes ou au nom d'utilisateur. Pour que votre principal IAM bénéficie des autorisations définies dans les stratégies d'accès associées à l'entrée d'accès, n'usurpez pas l'identité d'un utilisateur ou d'un groupe Kubernetes. Le principal IAM conserve également toute autorisation que vous lui avez accordée via des objets Kubernetes `Role` ou `ClusterRole` liés aux noms de groupes ou au nom d'utilisateur que vous avez définis pour l'entrée d'accès. Pour plus d'informations, consultez Usurpation d'identité d'utilisateur dans la documentation Kubernetes.

Vous pouvez associer une stratégie d'accès à une entrée d'accès à l'aide de la AWS Management Console ou de la CLI AWS.

AWS Management Console

1. Ouvrez la console Amazon EKS.
2. Choisissez le nom du cluster qui possède une entrée d'accès à laquelle associer une stratégie d'accès.
3. Choisissez l'onglet Access.
4. Si le type d'entrée d'accès est Standard, vous pouvez associer ou dissocier les stratégies d'accès Amazon EKS. Si le type de votre entrée d'accès est différent de Standard, cette option n'est pas disponible.
5. Choisissez Associer la stratégie d'accès.
6. Dans Nom de la stratégie, sélectionnez la stratégie avec les autorisations que vous souhaitez attribuer au principal IAM. Pour consulter les autorisations incluses dans chaque stratégie, consultez the section called "Vérification des stratégies d'accès".

7. Pour Portée d'accès, choisissez une portée d'accès. Si vous choisissez Cluster, les autorisations de la stratégie d'accès sont accordées au principal IAM pour tous les espaces de noms Kubernetes. Si vous choisissez Espace de noms Kubernetes, vous pouvez ensuite sélectionner Ajouter un nouvel espace de noms. Dans le champ Espace de noms qui s'affiche, saisissez le nom d'un espace de noms Kubernetes de votre cluster. Si vous souhaitez que le principal IAM dispose d'autorisations sur plusieurs espaces de noms, vous pouvez saisir plusieurs espaces de noms.
8. Choisissez Ajouter une stratégie d'accès.

AWS CLI

1. Version `2.12.3` ou ultérieure ou version `1.27.160` ou ultérieure de l'interface de la ligne de commande AWS (AWS CLI) installée et configurée sur votre appareil ou dans AWS CloudShell. Pour vérifier votre version actuelle, utilisez `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Les gestionnaires de package tels que `yum`, `apt-get` ou Homebrew pour macOS ont souvent plusieurs versions de retard par rapport à la dernière version de l'AWS CLI. Pour installer la dernière version, consultez les sections Installation et Configuration rapide avec aws configure du Guide de l'utilisateur de l'interface de la ligne de commande AWS. La version de l'AWS CLI installée dans AWS CloudShell peut également être plusieurs versions derrière la dernière version. Pour la mettre à jour, consultez la section Installation de l'AWS CLI dans votre répertoire personnel dans le Guide de l'utilisateur AWS CloudShell.
2. Affichez les stratégies d'accès disponibles.

```
aws eks list-access-policies --output table
```

L'exemple qui suit illustre un résultat.

```
-------------------------------------------------------------------------------------------
|                                        ListAccessPolicies                                |
                  |
+-----------------------------------------------------------------------------------------
+
||                                          accessPolicies                                 |
                 ||
|+-------------------------------------------------------------------
+------------------------------+|
||                                arn                                 |
 name             ||
```

```
|+-----------------------------------------------------------------------
+------------------------------+|
||  {arn-aws}eks::aws:cluster-access-policy/AmazonEKSAdminPolicy        |
 AmazonEKSAdminPolicy         ||
||  {arn-aws}eks::aws:cluster-access-policy/AmazonEKSClusterAdminPolicy |
 AmazonEKSClusterAdminPolicy  ||
||  {arn-aws}eks::aws:cluster-access-policy/AmazonEKSEditPolicy         |
 AmazonEKSEditPolicy          ||
||  {arn-aws}eks::aws:cluster-access-policy/AmazonEKSViewPolicy         |
 AmazonEKSViewPolicy          ||
|+-----------------------------------------------------------------------
+------------------------------+|
```

Pour consulter les autorisations incluses dans chaque stratégie, consultez the section called "Vérification des stratégies d'accès".

3. Consultez vos entrées d'accès existantes. Remplacez *my-cluster* par le nom de votre cluster.

```
aws eks list-access-entries --cluster-name my-cluster
```

L'exemple qui suit illustre un résultat.

```
{
    "accessEntries": [
        "arn:aws:iam::111122223333:role/my-role",
        "arn:aws:iam::111122223333:user/my-user"
    ]
}
```

4. Associez une stratégie d'accès à une entrée d'accès. L'exemple suivant associe la stratégie d'accès `AmazonEKSViewPolicy` à une entrée d'accès. Chaque fois que le rôle IAM *my-role* tente d'accéder aux objets Kubernetes du cluster, Amazon EKS autorise ce rôle à utiliser les autorisations définies dans la stratégie d'accès uniquement pour les objets Kubernetes situés dans les espaces de noms *my-namespace1* et *my-namespace2*. Remplacez *my-cluster* par le nom de votre cluster, *111122223333* par l'ID de votre compte AWS et *my-role* par le nom du rôle IAM pour lequel vous souhaitez qu'Amazon EKS autorise l'accès aux objets du cluster Kubernetes.

```
aws eks associate-access-policy --cluster-name my-cluster --principal-arn
 arn:aws:iam::111122223333:role/my-role \
    --access-scope type=namespace,namespaces=my-namespace1,my-namespace2 --policy-arn
 arn:aws:eks::aws:cluster-access-policy/AmazonEKSViewPolicy
```

Si vous souhaitez que le principal IAM dispose des autorisations à l'échelle du cluster, remplacez `type=namespace,namespaces=my-namespace1,my-namespace2` par `type=cluster`. Si vous souhaitez associer plusieurs stratégies d'accès à l'entrée d'accès, exécutez la commande plusieurs fois, chaque fois avec une stratégie d'accès unique. Chaque stratégie d'accès associée possède sa propre portée.

> **Note**
>
> Si vous souhaitez ultérieurement modifier la portée d'une stratégie d'accès associée, réexécutez la commande précédente avec la nouvelle portée. Par exemple, si vous souhaitez supprimer *my-namespace2*, vous devez réexécuter la commande en utilisant uniquement `type=namespace,namespaces=my-namespace1` . Si vous souhaitez modifier la portée de `namespace` à `cluster`, vous devez exécuter à nouveau la commande en utilisant `type=cluster`, ce qui supprime `type=namespace,namespaces=my-namespace1,my-namespace2` .

5. Déterminez les stratégies d'accès associées à une entrée d'accès.

```
aws eks list-associated-access-policies --cluster-name my-cluster --principal-arn
 arn:aws:iam::111122223333:role/my-role
```

L'exemple qui suit illustre un résultat.

```
{
    "clusterName": "my-cluster",
    "principalArn": "arn:aws:iam::111122223333",
    "associatedAccessPolicies": [
        {
            "policyArn": "arn:aws:eks::aws:cluster-access-policy/
AmazonEKSViewPolicy",
            "accessScope": {
                "type": "cluster",
                "namespaces": []
            },
            "associatedAt": "2023-04-17T15:25:21.675000-04:00",
            "modifiedAt": "2023-04-17T15:25:21.675000-04:00"
        },
        {
```

```
            "policyArn": "arn:aws:eks::aws:cluster-access-policy/
AmazonEKSAdminPolicy",
            "accessScope": {
                "type": "namespace",
                "namespaces": [
                    "my-namespace1",
                    "my-namespace2"
                ]
            },
            "associatedAt": "2023-04-17T15:02:06.511000-04:00",
            "modifiedAt": "2023-04-17T15:02:06.511000-04:00"
        }
    ]
}
```

Dans l'exemple précédent, le principal IAM associé à cette entrée d'accès dispose d'autorisations de lecture sur tous les espaces de noms du cluster et d'autorisations administrateur sur deux espaces de noms Kubernetes.

6. Dissociez une stratégie d'accès d'une entrée d'accès. Dans cet exemple, la stratégie `AmazonEKSAdminPolicy` est dissociée d'une entrée d'accès. Le principal IAM conserve toutefois les autorisations définies dans la stratégie d'accès `AmazonEKSViewPolicy` pour les objets des espaces de noms *`my-namespace1`* et *`my-namespace2`*, car cette stratégie d'accès n'est pas dissociée de l'entrée d'accès.

```
aws eks disassociate-access-policy --cluster-name my-cluster --principal-arn
 arn:aws:iam::111122223333:role/my-role \
   --policy-arn arn:aws:eks::aws:cluster-access-policy/AmazonEKSAdminPolicy
```

Pour répertorier toutes les stratégies d'accès disponibles, consultez the section called "Vérification des stratégies d'accès".

Migration des entrées **`aws-auth ConfigMap`** existantes pour accéder aux entrées

Si vous avez ajouté des entrées de `ConfigMap` `aws-auth` à votre cluster, nous vous recommandons de créer des entrées d'accès pour les entrées existantes dans votre cluster `ConfigMap` `aws-auth`. Après avoir créé les entrées d'accès, vous pouvez les supprimer de votre `ConfigMap`. Vous ne pouvez pas associer de stratégies d'accès aux entrées dans la `ConfigMap` `aws-auth`. Si vous souhaitez associer des stratégies d'accès à vos principaux IAM, créez des entrées d'accès.

> **Important**
>
> - Lorsqu'un cluster est en mode d'authentification `API_AND_CONFIGMAP` et qu'il existe un mappage pour le même rôle IAM dans les `aws-auth ConfigMap` et dans les entrées d'accès, le rôle utilisera le mappage de l'entrée d'accès pour l'authentification. Les entrées d'accès ont priorité sur les entrées `ConfigMap` pour le même principal IAM.
> - Avant de supprimer les entrées `aws-auth ConfigMap` existantes créées par Amazon EKS pour le groupe de nœuds gérés ou un profil Fargate vers votre cluster, vérifiez que les entrées d'accès correctes pour ces ressources spécifiques existent dans votre cluster Amazon EKS. Si vous supprimez les entrées créées par Amazon EKS dans le `ConfigMap` sans disposer des entrées d'accès équivalentes, votre cluster ne fonctionnera pas correctement.

Conditions préalables

- Avoir des connaissances sur les entrées d'accès et les stratégies d'accès. Pour plus d'informations, consultez the section called "Entrées d'accès" et the section called "Association des stratégies d'accès".
- Un cluster existant avec une version de plateforme égale ou supérieure aux versions répertoriées dans les conditions préalables de la rubrique the section called "Entrées d'accès".
- Version `0.215.0` ou ultérieure de l'outil de ligne de commande `eksctl` installée sur votre appareil ou AWS CloudShell. Pour installer ou mettre à jour `eksctl`, veuillez consulter Installation dans la documentation de `eksctl`.
- Autorisations Kubernetes permettant de modifier le `aws-auth ConfigMap` dans l'espace de noms `kube-system`.
- Un rôle ou un utilisateur AWS Identity and Access Management disposant des autorisations suivantes : `CreateAccessEntry` et`ListAccessEntries`. Pour plus d'informations, consultez la rubrique Actions définies par Amazon Elastic Kubernetes Service dans la Référence des autorisations de service.

`eksctl`

1. Consultez les entrées existantes dans votre `aws-auth ConfigMap`. Remplacez *`my-cluster`* par le nom de votre cluster.

```
eksctl get iamidentitymapping --cluster my-cluster
```

L'exemple qui suit illustre un résultat.

```
ARN
           USERNAME                                      GROUPS
                       ACCOUNT
arn:aws: iam::111122223333:role/EKS-my-cluster-Admins
            Admins                                        system:masters
arn:aws: iam::111122223333:role/EKS-my-cluster-my-namespace-Viewers
            my-namespace-Viewers                          Viewers
arn:aws: iam::111122223333:role/EKS-my-cluster-self-managed-ng-1
            system:node:{{EC2PrivateDNSName}}       system:bootstrappers,system:nodes
arn:aws: iam::111122223333:user/my-user
            my-user
arn:aws: iam::111122223333:role/EKS-my-cluster-fargateprofile1
                              system:node:{{SessionName}}
 system:bootstrappers,system:nodes,system:node-proxier
arn:aws: iam::111122223333:role/EKS-my-cluster-managed-ng
            system:node:{{EC2PrivateDNSName}}       system:bootstrappers,system:nodes
```

2. the section called "Créer des entrées d'accès" pour toutes les entrées `ConfigMap` que vous avez créées et qui apparaissent dans la sortie précédente. Lorsque vous créez les entrées d'accès, assurez-vous de spécifier les mêmes valeurs pour `ARN`, `USERNAME`, `GROUPS` et `ACCOUNT` et de les renvoyer dans votre résultat. Dans l'exemple de résultat, vous devez créer des entrées d'accès pour toutes les entrées sauf les deux dernières, étant donné que ces entrées ont été créées par Amazon EKS pour un profil Fargate et un groupe de nœuds géré.
3. Supprimez les entrées de la `ConfigMap` pour toutes les entrées d'accès que vous avez créées. Si vous ne supprimez pas l'entrée de la `ConfigMap`, les paramètres de l'entrée d'accès pour l'ARN principal IAM remplaceront l'entrée `ConfigMap`. *`111122223333`*Remplacez-le par votre numéro de AWS compte et *`EKS-my-cluster-my-namespace-Viewers`* par le nom du rôle dans l'entrée de votre`ConfigMap`. Si l'entrée que vous supprimez concerne un utilisateur IAM plutôt qu'un rôle IAM, remplacez-la par `user` et `role` *`EKS-my-cluster-my-namespace-Viewers`* par le nom d'utilisateur.

```
eksctl delete iamidentitymapping --arn arn:aws: iam::111122223333:role/EKS-my-
cluster-my-namespace-Viewers --cluster my-cluster
```

Vérification des autorisations des stratégies d'accès

Les stratégies d'accès incluent des `rules` qui contiennent des `verbs` (autorisations) et des `resources` Kubernetes. Les stratégies d'accès n'incluent pas d'autorisations IAM ni de ressources IAM. Similaires aux objets Kubernetes `Role` et `ClusterRole`, les stratégies d'accès ne contiennent que des `rules` `allow`. Vous ne pouvez pas modifier le contenu d'une stratégie d'accès. Vous ne pouvez pas créer vos propres stratégies d'accès. Si les autorisations fournies par les stratégies d'accès ne correspondent pas à vos besoins, créez des objets RBAC Kubernetes et indiquez des noms de groupes dans vos entrées d'accès. Pour de plus amples informations, veuillez consulter the section called "Créer des entrées d'accès". Les autorisations définies dans les stratégies d'accès sont similaires à celles fournies par les rôles de cluster Kubernetes destinés aux utilisateurs. Pour plus d'informations, consultez la section Rôles destinés aux utilisateurs dans la documentation Kubernetes.

Liste de toutes les stratégies

Utilisez l'une des politiques d'accès répertoriées sur cette page ou récupérez la liste de toutes les politiques d'accès disponibles à l'aide de la AWS CLI :

```
aws eks list-access-policies
```

Le résultat attendu doit ressembler à ceci (abrégé par souci de concision) :

```
{
    "accessPolicies": [
        {
            "name": "AmazonAIOpsAssistantPolicy",
            "arn": "arn:aws:eks::aws:cluster-access-policy/AmazonAIOpsAssistantPolicy"
        },
        {
            "name": "AmazonARCRegionSwitchScalingPolicy",
            "arn": "arn:aws:eks::aws:cluster-access-policy/
AmazonARCRegionSwitchScalingPolicy"
        },
        {
            "name": "AmazonEKSAdminPolicy",
            "arn": "arn:aws:eks::aws:cluster-access-policy/AmazonEKSAdminPolicy"
        },
        {
            "name": "AmazonEKSAdminViewPolicy",
            "arn": "arn:aws:eks::aws:cluster-access-policy/AmazonEKSAdminViewPolicy"
```

```
        },
        {
            "name": "AmazonEKSAutoNodePolicy",
            "arn": "arn:aws:eks::aws:cluster-access-policy/AmazonEKSAutoNodePolicy"
        }
        // Additional policies omitted
    ]
}
```

EKSAdminPolitique d'Amazon

Cette stratégie d'accès inclut des autorisations qui accordent à un principal IAM le plus grand nombre d'autorisations sur les ressources. Lorsqu'elle est associée à une entrée d'accès, sa portée d'accès correspond généralement à un ou plusieurs espaces de noms Kubernetes. Si vous souhaitez qu'un principal IAM dispose d'un accès administrateur à toutes les ressources de votre cluster, associez plutôt la stratégie d'accès the section called "Amazon EKSCluster AdminPolicy" à votre entrée d'accès.

ARN – `arn:aws: eks::aws:cluster-access-policy/AmazonEKSAdminPolicy`

Groupes d'API Kubernetes	Ressources Kubernetes	Verbes (autorisations) Kubernetes
apps	daemonsets , deployments , deployments/rollback , deployments/scale , replicasets , replicasets/scale, statefulsets , statefulsets/scale	create, delete, deletecollection , patch, update
apps	controllerrevisions , daemonsets , daemonsets/status , deployments , deployments/scale , deployments/status , replicasets , replicasets/scale , replicasets/status , statefuls	get, list, watch

Groupes d'API Kubernetes	Ressources Kubernetes	Verbes (autorisations) Kubernetes
	ets , statefulsets/scale , statefulsets/status	
authorization.k8s.io	localsubjectaccessreviews	create
autoscaling	horizontalpodautoscalers	create, delete, deletecollection , patch, update
autoscaling	horizontalpodautoscalers , horizontalpodautoscalers/status	get, list, watch
batch	cronjobs, jobs	create, delete, deletecollection , patch, update
batch	cronjobs, cronjobs/status , jobs, jobs/status	get, list, watch
discovery.k8s.io	endpointslices	get, list, watch
extensions	daemonsets , deployments , deployments/rollback , deployments/scale , ingresses , networkpolicies , replicasets , replicasets/scale , replicationcontrollers/scale	create, delete, deletecollection , patch, update

Groupes d'API Kubernetes	Ressources Kubernetes	Verbes (autorisations) Kubernetes
extensions	daemonsets , daemonsets/status , deployments , deployments/scale , deployments/status , ingresses , ingresses/status , networkpolicies , replicasets , replicasets/scale , replicasets/status , replicationcontrollers/scale	get, list, watch
networking.k8s.io	ingresses , ingresses/status , networkpolicies	get, list, watch
networking.k8s.io	ingresses , networkpolicies	create, delete, deletecollection , patch, update
policy	poddisruptionbudgets	create, delete, deletecollection , patch, update
policy	poddisruptionbudgets , poddisruptionbudgets/status	get, list, watch
rbac.authorization.k8s.io	rolebindings , roles	create, delete, deletecollection , get, list, patch, update, watch

Groupes d'API Kubernetes	Ressources Kubernetes	Verbes (autorisations) Kubernetes
	configmaps , endpoints , persistentvolumeclaims , persistentvolumeclaims/status , pods, replicationcontrollers , replicationcontrollers/scale , serviceaccounts , services, services/status	get,list, watch
	pods/attach , pods/exec , pods/portforward , pods/proxy , secrets, services/proxy	get, list, watch
	configmaps , events, persistentvolumeclaims , replicationcontrollers , replicationcontrollers/scale , secrets, serviceaccounts , services, services/proxy	create, delete, deletecollection , patch, update
	pods, pods/attach , pods/exec , pods/portforward , pods/proxy	create, delete, deletecollection , patch, update
	serviceaccounts	impersonate

Groupes d'API Kubernetes	Ressources Kubernetes	Verbes (autorisations) Kubernetes
	`bindings`, `events`, `limitranges` , `namespaces/status` , `pods/log`, `pods/status` , `replicationcontrollers/status` , `resourcequotas` , `resourcequotas/status`	`get`, `list`, `watch`
	`namespaces`	`get`,`list`, `watch`

Amazon EKSCluster AdminPolicy

Cette stratégie d'accès inclut des autorisations qui accordent à un administrateur principal IAM l'accès à un cluster. Lorsqu'elle est associée à une entrée d'accès avec portée Cluster, sa portée concerne l'ensemble du cluster, plutôt qu'un espace de noms unique. Si vous souhaitez qu'un principal IAM ait une portée administrative plus limitée, pensez plutôt à associer la stratégie d'accès the section called "EKSAdminPolitique d'Amazon" à votre entrée d'accès.

ARN – `arn:aws: eks::aws:cluster-access-policy/AmazonEKSClusterAdminPolicy`

Groupes d'API Kubernetes	Kubernetes Non-Resource URLs	Ressources Kubernetes	Verbes (autorisations) Kubernetes
*		*	*
	*		*

Amazon EKSAdmin ViewPolicy

Cette politique d'accès inclut des autorisations qui accordent à un IAM principal l'accès à list/view toutes les ressources d'un cluster. Notez que cela inclut les secrets Kubernetes.

ARN – `arn:aws: eks::aws:cluster-access-policy/AmazonEKSAdminViewPolicy`

Groupes d'API Kubernetes	Ressources Kubernetes	Verbes (autorisations) Kubernetes
*	*	get, list, watch

EKSEditPolitique d'Amazon

Cette stratégie d'accès inclut des autorisations permettant à un principal IAM de modifier la plupart des ressources Kubernetes.

ARN – arn:aws: eks::aws:cluster-access-policy/AmazonEKSEditPolicy

Groupes d'API Kubernetes	Ressources Kubernetes	Verbes (autorisations) Kubernetes
apps	daemonsets , deployments , deployments/rollback , deployments/scale , replicasets , replicasets/scale, statefulsets , statefulsets/scale	create, delete, deletecollection , patch, update
apps	controllerrevisions , daemonsets , daemonsets/status , deployments , deployments/scale , deployments/status , replicasets , replicasets/scale , replicasets/status , statefulsets , statefulsets/scale , statefulsets/status	get, list, watch
autoscaling	horizontalpodautoscalers , horizonta	get, list, watch

Groupes d'API Kubernetes	Ressources Kubernetes	Verbes (autorisations) Kubernetes
	lpodautoscalers/status	
autoscaling	horizontalpodautoscalers	create, delete, deletecollection , patch, update
batch	cronjobs, jobs	create, delete, deletecollection , patch, update
batch	cronjobs, cronjobs/status , jobs, jobs/status	get, list, watch
discovery.k8s.io	endpointslices	get, list, watch
extensions	daemonsets , deployments , deployments/rollback , deployments/scale , ingresses , networkpolicies , replicasets , replicasets/scale , replicationcontrollers/scale	create, delete, deletecollection , patch, update
extensions	daemonsets , daemonsets/status , deployments , deployments/scale , deployments/status , ingresses , ingresses/status , networkpolicies , replicasets , replicasets/scale , replicasets/status , replicationcontrollers/scale	get, list, watch

Groupes d'API Kubernetes	Ressources Kubernetes	Verbes (autorisations) Kubernetes
networking.k8s.io	ingresses , networkpolicies	create, delete, deletecollection , patch, update
networking.k8s.io	ingresses , ingresses/status , networkpolicies	get, list, watch
policy	poddisruptionbudgets	create, delete, deletecollection , patch, update
policy	poddisruptionbudgets , poddisruptionbudgets/status	get, list, watch
	namespaces	get, list, watch
	pods/attach , pods/exec , pods/portforward , pods/proxy , secrets, services/proxy	get, list, watch
	serviceaccounts	impersonate
	pods, pods/attach , pods/exec , pods/portforward , pods/proxy	create, delete, deletecollection , patch, update

Groupes d'API Kubernetes	Ressources Kubernetes	Verbes (autorisations) Kubernetes
	configmaps , events, persistentvolumeclaims , replicationcontrollers , replicationcontrollers/scale , secrets, serviceaccounts , services, services/proxy	create, delete, deletecollection , patch, update
	configmaps , endpoints , persistentvolumeclaims , persistentvolumeclaims/status , pods, replicationcontrollers , replicationcontrollers/scale , serviceaccounts , services, services/status	get, list, watch
	bindings, events, limitranges , namespaces/status , pods/log, pods/status , replicationcontrollers/status , resourcequotas , resourcequotas/status	get, list, watch

EKSViewPolitique d'Amazon

Cette stratégie d'accès inclut des autorisations permettant à un principal IAM de consulter la plupart des ressources Kubernetes.

ARN – arn:aws: eks::aws:cluster-access-policy/AmazonEKSViewPolicy

Groupes d'API Kubernetes	Ressources Kubernetes	Verbes (autorisations) Kubernetes
apps	controllerrevisions , daemonsets , daemonsets/status , deployments , deployments/scale , deployments/status , replicasets , replicasets/scale , replicasets/status , statefulsets , statefulsets/scale , statefulsets/status	get, list, watch
autoscaling	horizontalpodautoscalers , horizontalpodautoscalers/status	get, list, watch
batch	cronjobs, cronjobs/status , jobs, jobs/status	get, list, watch
discovery.k8s.io	endpointslices	get, list, watch
extensions	daemonsets , daemonsets/status , deployments , deployments/scale , deployments/status , ingresses , ingresses/status , networkpolicies , replicasets , replicasets/scale , replicasets/status	get, list, watch

Groupes d'API Kubernetes	Ressources Kubernetes	Verbes (autorisations) Kubernetes
	, replicationcontrollers/scale	
networking.k8s.io	ingresses , ingresses/status , networkpolicies	get, list, watch
policy	poddisruptionbudgets , poddisruptionbudgets/status	get, list, watch
	configmaps , endpoints , persistentvolumeclaims , persistentvolumeclaims/status , pods, replicationcontrollers , replicationcontrollers/scale , serviceaccounts , services, services/status	get, list, watch
	bindings, events, limitranges , namespaces/status , pods/log, pods/status , replicationcontrollers/status , resourcequotas , resourcequotas/status	get, list, watch
	namespaces	get, list, watch

Amazon EKSSecret ReaderPolicy

Cette politique d'accès inclut des autorisations qui permettent à un principal IAM de lire les secrets Kubernetes.

ARN – `arn:aws: eks::aws:cluster-access-policy/AmazonEKSSecretReaderPolicy`

Groupes d'API Kubernetes	Ressources Kubernetes	Verbes (autorisations) Kubernetes
	`secrets`	`get`, `list`, `watch`

Amazon EKSAuto NodePolicy

ARN – `arn:aws: eks::aws:cluster-access-policy/AmazonEKSAutoNodePolicy`

Cette stratégie inclut les autorisations suivantes, permettant aux composants Amazon EKS d'accomplir les tâches suivantes :

- `kube-proxy` : surveiller les points de terminaison réseau et les services, et gérer les événements associés. Cela permet d'activer la fonctionnalité de proxy réseau à l'échelle du cluster.
- `ipamd`— Gérez les ressources réseau AWS VPC et les interfaces réseau de conteneurs (CNI). Cela permet au démon de gestion des adresses IP de gérer le réseau des pods.
- `coredns` : accéder aux ressources de découverte de services telles que les points de terminaison et les services. Cela permet la résolution DNS au sein du cluster.
- `ebs-csi-driver` : gérer les ressources liées au stockage pour les volumes Amazon EBS. Cela permet le provisionnement dynamique et l'attachement des volumes persistants.
- `neuron`— Surveillez les nœuds et les pods des appareils AWS Neuron. Cela permet de gérer les accélérateurs AWS Inferentia et Trainium.
- `node-monitoring-agent` : accéder aux diagnostics et événements des nœuds. Cela permet la surveillance de l'état du cluster et la collecte des diagnostics.

Chaque composant utilise un compte de service dédié et ne dispose que des autorisations nécessaires à sa fonction spécifique.

Si vous spécifiez manuellement un rôle Node IAM dans un NodeClass, vous devez créer une entrée d'accès qui associe le nouveau rôle Node IAM à cette politique d'accès.

Amazon EKSBlock StoragePolicy

> **Note**
>
> Cette politique s'applique uniquement aux rôles AWS liés aux services et ne peut pas être utilisée avec des rôles gérés par le client.

ARN – `arn:aws: eks::aws:cluster-access-policy/AmazonEKSBlockStoragePolicy`

Cette stratégie inclut des autorisations permettant à Amazon EKS de gérer l'élection du leader et les ressources de coordination pour les opérations de stockage :

- `coordination.k8s.io` : créer et gérer des objets Lease pour l'élection du leader. Cela permet aux composants de stockage EKS de coordonner leurs opérations dans tout le cluster grâce à un mécanisme d'élection du leader.

La stratégie est limitée à des ressources de location spécifiques utilisées par les composants de stockage EKS, afin d'éviter tout conflit d'accès avec d'autres ressources de coordination présentes dans le cluster.

Amazon EKS crée automatiquement une entrée d'accès avec cette stratégie d'accès pour le rôle IAM du cluster lorsque le mode automatique est activé, garantissant que les autorisations nécessaires sont en place pour que la fonctionnalité de stockage par blocs fonctionne correctement.

Amazon EKSLoad BalancingPolicy

ARN – `arn:aws: eks::aws:cluster-access-policy/AmazonEKSLoadBalancingPolicy`

Cette stratégie inclut des autorisations qui permettent à Amazon EKS de gérer les ressources d'élection du leader pour l'équilibrage de charge :

- `coordination.k8s.io` : créer et gérer des objets Lease pour l'élection du leader. Cela permet aux composants d'équilibrage de charge EKS de coordonner leurs activités entre plusieurs réplicas grâce à l'élection d'un leader.

La stratégie est limitée aux ressources de location dédiées à l'équilibrage de charge afin d'assurer une bonne coordination, tout en empêchant l'accès à d'autres ressources de location du cluster.

Amazon EKS crée automatiquement une entrée d'accès avec cette stratégie d'accès pour le rôle IAM du cluster lorsque le mode automatique est activé, garantissant que les autorisations nécessaires sont en place pour que la fonctionnalité réseau fonctionne correctement.

EKSNetworkingPolitique d'Amazon

ARN – `arn:aws: eks::aws:cluster-access-policy/AmazonEKSNetworkingPolicy`

Cette stratégie inclut des autorisations qui permettent à Amazon EKS de gérer les ressources d'élection du leader pour les opérations réseau :

- `coordination.k8s.io` : créer et gérer des objets Lease pour l'élection du leader. Cela permet aux composants réseau EKS de coordonner les activités d'allocation d'adresses IP grâce à l'élection d'un leader.

La stratégie est limitée aux ressources de location dédiées au réseau afin d'assurer une bonne coordination, tout en empêchant l'accès à d'autres ressources de location du cluster.

Amazon EKS crée automatiquement une entrée d'accès avec cette stratégie d'accès pour le rôle IAM du cluster lorsque le mode automatique est activé, garantissant que les autorisations nécessaires sont en place pour que la fonctionnalité réseau fonctionne correctement.

EKSComputePolitique d'Amazon

> **Note**
>
> Cette politique s'applique uniquement aux rôles AWS liés aux services et ne peut pas être utilisée avec des rôles gérés par le client.

ARN – `arn:aws: eks::aws:cluster-access-policy/AmazonEKSComputePolicy`

Cette stratégie inclut des autorisations permettant à Amazon EKS de gérer les ressources d'élection du leader pour les opérations de calcul :

- `coordination.k8s.io` : créer et gérer des objets Lease pour l'élection du leader. Cela permet aux composants de calcul EKS de coordonner les activités de mise à l'échelle des nœuds grâce à l'élection d'un leader.

La stratégie est limitée aux ressources de location dédiées à la gestion du calcul, tout en autorisant un accès en lecture de base (`get`, `watch`) à toutes les ressources de location du cluster.

Amazon EKS crée automatiquement une entrée d'accès avec cette stratégie d'accès pour le rôle IAM du cluster lorsque le mode automatique est activé, garantissant que les autorisations nécessaires sont en place pour que la fonctionnalité réseau fonctionne correctement.

Amazon EKSBlock StorageClusterPolicy

> **Note**
>
> Cette politique s'applique uniquement aux rôles AWS liés aux services et ne peut pas être utilisée avec des rôles gérés par le client.

ARN – `arn:aws: eks::aws:cluster-access-policy/AmazonEKSBlockStorageClusterPolicy`

Cette politique accorde les autorisations nécessaires pour la fonctionnalité de stockage en blocs du mode automatique Amazon EKS. Elle permet une gestion efficace des ressources de stockage par blocs au sein des clusters Amazon EKS. La stratégie inclut les autorisations suivantes :

Gestion des pilotes CSI :

- Créer, lire, mettre à jour et supprimer les pilotes CSI, spécifiquement pour le stockage par blocs.

Gestion des volumes :

- Répertorier, surveiller, créer, mettre à jour, patcher et supprimer les volumes persistants.
- Répertorier, surveiller et mettre à jour les demandes de volumes persistants.
- Corriger l'état des demandes de volumes persistants.

Interaction entre les nœuds et les pods :

- Lire les informations sur les nœuds et les pods.
- Gérer les événements liés aux opérations de stockage.

Classes de stockage et attributs :

- Lire les classes de stockage et les nœuds CSI.
- Lire les classes d'attributs de volume.

Attachements de volumes :

- Répertorier, surveiller et modifier les attachements de volumes et leurs statuts.

Opérations d'instantanés :

- Gérer les instantanés de volumes, le contenu des instantanés et les classes d'instantanés.
- Gérer les opérations pour les instantanés de groupes de volumes et les ressources associées.

Cette politique est conçue pour prendre en charge la gestion complète du stockage en blocs au sein des clusters Amazon EKS s'exécutant en mode automatique. Elle combine les autorisations nécessaires pour les opérations de provisionnement, attachement, redimensionnement et création d'instantanés de volumes de stockage par blocs.

Amazon EKS crée automatiquement une entrée d'accès avec cette stratégie d'accès pour le rôle IAM du cluster lorsque le mode automatique est activé, garantissant que les autorisations nécessaires sont en place pour que la fonctionnalité de stockage par blocs fonctionne correctement.

Amazon EKSCompute ClusterPolicy

> **Note**
>
> Cette politique s'applique uniquement aux rôles AWS liés aux services et ne peut pas être utilisée avec des rôles gérés par le client.

ARN – `arn:aws: eks::aws:cluster-access-policy/AmazonEKSComputeClusterPolicy`

Cette politique accorde les autorisations nécessaires à la fonctionnalité de gestion du calcul du mode automatique Amazon EKS. Elle permet une orchestration efficace et une mise à l'échelle des ressources de calcul au sein des clusters Amazon EKS. La stratégie inclut les autorisations suivantes :

Gestion des nœuds :

- Créez, lisez, mettez à jour, supprimez et gérez le statut de NodePools et NodeClaims.
- Gérez NodeClasses, y compris la création, la modification et la suppression.

Gestion de la planification et des ressources :

- Accès en lecture aux pods, aux nœuds, aux volumes persistants, aux demandes de volumes persistants, aux contrôleurs de réplication et aux espaces de noms.
- Accès en lecture aux classes de stockage, aux nœuds CSI et aux attachements de volumes.
- Répertorier et surveiller les déploiements, les ensembles de démons, les ensembles de réplicas et les ensembles avec état.
- Accès en lecture aux budgets d'interruption des pods.

Gestion des événements :

- Créer, lire et gérer les événements du cluster.

Déprovisionnement des nœuds et éviction des pods :

- Mettre à jour, corriger et supprimer des nœuds.
- Créer des évictions de pods et supprimer des pods lorsque nécessaire.

Gestion des définitions de ressources personnalisées (CRD) :

- Créez un nouveau CRDs.
- Gérez les CRDs éléments spécifiques liés à la gestion des nœuds (NodeClasses NodePools NodeClaims,, et NodeDiagnostics).

Cette stratégie est conçue pour assurer une gestion complète des ressources de calcul au sein des clusters Amazon EKS exécutés en mode automatique. Elle regroupe les autorisations nécessaires pour des opérations telles que le provisionnement des nœuds, la planification, la mise à l'échelle et l'optimisation des ressources.

Amazon EKS crée automatiquement une entrée d'accès avec cette stratégie d'accès pour le rôle IAM du cluster lorsque le mode automatique est activé, garantissant ainsi que les autorisations nécessaires à la gestion du calcul sont correctement appliquées.

Amazon EKSLoad BalancingClusterPolicy

ARN – `arn:aws: eks::aws:cluster-access-policy/AmazonEKSLoadBalancingClusterPolicy`

Cette stratégie accorde les autorisations nécessaires pour la fonctionnalité d'équilibrage de charge du mode automatique Amazon EKS. Elle permet une gestion et une configuration efficaces des ressources d'équilibrage de charge au sein des clusters Amazon EKS. La stratégie inclut les autorisations suivantes :

Gestion des événements et des ressources :

- Créer et corriger des événements.
- Accès en lecture aux pods, aux nœuds, aux points de terminaison et aux espaces de noms.
- Mettre à jour l'état des pods.

Gestion des ressources Service et Ingress :

- Gestion complète des services et de leurs états.
- Contrôle complet des ressources Ingress et de leurs états.
- Accès en lecture aux tranches de points de terminaison et aux classes de ressources Ingress.

Liaisons de groupes cibles :

- Créer et modifier les liaisons de groupes cibles et leurs états.
- Accès en lecture aux paramètres de la classe de ressources Ingress.

Gestion des définitions de ressources personnalisées (CRD) :

- Créez et lisez tout CRDs.
- Gestion spécifique de targetgroupbindings.eks.amazonaws.com et ingressclassparams.eks.amazonaws.com. CRDs

Configuration des webhooks :

- Créer et lire des configurations de webhooks de type mutant et validant.
- Gérez la eks-load-balancing-webhook configuration.

Cette stratégie est conçue pour assurer une gestion complète de l'équilibrage de charge au sein des clusters Amazon EKS exécutés en mode automatique. Il combine des autorisations pour diverses opérations, notamment l'exposition aux services, le routage d'entrée et l'intégration aux services d'équilibrage de AWS charge.

Amazon EKS crée automatiquement une entrée d'accès associée à cette stratégie d'accès pour le rôle IAM du cluster lorsque le mode automatique est activé, garantissant ainsi que les autorisations nécessaires à la fonctionnalité d'équilibrage de charge sont correctement appliquées.

Amazon EKSNetworking ClusterPolicy

ARN – `arn:aws: eks::aws:cluster-access-policy/AmazonEKSNetworkingClusterPolicy`

Amazon EKSNetworking ClusterPolicy

Cette stratégie accorde les autorisations nécessaires à la fonctionnalité réseau du mode automatique Amazon EKS. Elle permet une gestion et une configuration efficaces des ressources réseau au sein des clusters Amazon EKS. La stratégie inclut les autorisations suivantes :

Gestion des nœuds et des pods :

- Accès en lecture à NodeClasses et à leurs statuts.
- Accès en lecture à NodeClaims et à leurs statuts.
- Accès en lecture aux pods.

Gestion des nœuds CNI :

- Les autorisations CNINodes et leur statut, y compris la création, la lecture, la mise à jour, la suppression et le correctif.

Gestion des définitions de ressources personnalisées (CRD) :

- Créez et lisez tout CRDs.
- Gestion spécifique (mise à jour, correction, suppression) de la CRD cninodes.eks.amazonaws.com.

Gestion des événements :

- Créer et corriger des événements.

Cette stratégie est conçue pour assurer une gestion complète du réseau au sein des clusters Amazon EKS exécutés en mode automatique. Elle regroupe les autorisations nécessaires pour diverses opérations, incluant la configuration réseau des nœuds, la gestion CNI (Container Network Interface) et la gestion des ressources personnalisées liées au réseau.

La stratégie permet aux composants réseau d'interagir avec les ressources liées aux nœuds, de gérer les configurations spécifiques CNI des nœuds, et de traiter les ressources personnalisées essentielles aux opérations réseau dans le cluster.

Amazon EKS crée automatiquement une entrée d'accès avec cette stratégie d'accès pour le rôle IAM du cluster lorsque le mode automatique est activé, garantissant que les autorisations nécessaires sont en place pour que la fonctionnalité réseau fonctionne correctement.

EKSHybridPolitique d'Amazon

> **Note**
>
> Cette politique s'applique uniquement aux rôles AWS liés aux services et ne peut pas être utilisée avec des rôles gérés par le client.

Cette stratégie d'accès inclut des autorisations qui permettent à Amazon EKS d'accéder aux nœuds d'un cluster. Lorsqu'elle est associée à une entrée d'accès avec portée Cluster, sa portée concerne l'ensemble du cluster, plutôt qu'un espace de noms unique. Cette stratégie est utilisée par les nœuds hybrides Amazon EKS.

ARN – `arn:aws: eks::aws:cluster-access-policy/AmazonEKSHybridPolicy`

Groupes d'API Kubernetes	Kubernetes Non-Resource URLs	Ressources Kubernetes	Verbes (autorisations) Kubernetes
*		`nodes`	`list`

Amazon EKSCluster InsightsPolicy

> **Note**
>
> Cette politique s'applique uniquement aux rôles AWS liés aux services et ne peut pas être utilisée avec des rôles gérés par le client.

ARN – `arn:aws: eks::aws:cluster-access-policy/AmazonEKSClusterInsightsPolicy`

Cette stratégie accorde des autorisations en lecture seule pour la fonctionnalité EKS Cluster Insights. La stratégie inclut les autorisations suivantes :

Accès aux nœuds : - Répertorier et afficher les nœuds du cluster - Lire les informations sur l'état des nœuds

DaemonSet Accès : - Accès en lecture à la configuration de kube-proxy

Cette stratégie est automatiquement gérée par le service EKS pour la fonctionnalité Cluster Insights. Pour de plus amples informations, veuillez consulter the section called "Informations sur le cluster".

AWSBackupFullAccessPolicyForBackup

ARN – `arn:aws: eks::aws:cluster-access-policy/AWSBackupFullAccessPolicyForBackup`

AWSBackupFullAccessPolicyForBackup

Cette politique accorde les autorisations nécessaires à AWS Backup pour gérer et créer des sauvegardes du cluster EKS. Cette politique inclut les autorisations suivantes :

Groupes d'API Kubernetes	Ressources Kubernetes	Verbes (autorisations) Kubernetes
*	*	`list`, `get`

AWSBackupFullAccessPolicyForRestore

ARN – `arn:aws: eks::aws:cluster-access-policy/AWSBackupFullAccessPolicyForRestore`

AWSBackupFullAccessPolicyForRestore

Cette politique accorde les autorisations nécessaires à AWS Backup pour gérer et restaurer les sauvegardes du cluster EKS. Cette politique inclut les autorisations suivantes :

Groupes d'API Kubernetes	Ressources Kubernetes	Verbes (autorisations) Kubernetes
*	*	`list`, `get`, `create`

Amazon EKSACKPolicy

ARN – `arn:aws: eks::aws:cluster-access-policy/AmazonEKSACKPolicy`

Cette politique accorde les autorisations nécessaires pour que les AWS Controllers for Kubernetes (ACK) puissent gérer les AWS ressources de Kubernetes. La stratégie inclut les autorisations suivantes :

Gestion personnalisée des ressources ACK :

- Accès complet à toutes les ressources personnalisées du service ACK sur plus de 50 AWS services, notamment S3, RDS, DynamoDB, Lambda, EC2, etc.
- Créez, lisez, mettez à jour et supprimez les définitions de ressources personnalisées ACK.

Accès à l'espace de noms :

- Accès en lecture aux espaces de noms pour l'organisation des ressources.

Élection du chef :

- Créez et lisez des baux de coordination pour l'élection des dirigeants.
- Mettez à jour et supprimez les baux spécifiques des contrôleurs de service ACK.

Gestion des événements :

- Créez et corrigez des événements pour les opérations ACK.

Cette politique est conçue pour prendre en charge la gestion complète AWS des ressources via Kubernetes APIs. Amazon EKS crée automatiquement une entrée d'accès avec cette politique d'accès pour le rôle IAM de fonctionnalité que vous fournissez lors de la création de la fonctionnalité ACK.

Groupes d'API Kubernetes	Ressources Kubernetes	Verbes (autorisations) Kubernetes
	`namespaces`	`get`, `watch`, `list`
`services.k8s.aws` , `acm.services.k8s.aws` , `acmpca.services.k8s.aws` , `apigateway.services.k8s.aws` , `apigatewayv2.services.k8s.aws` , `applicationautoscaling.services.k8s.aws` , `athena.services.k8s.aws` , `bedrock.services.k8s.aws` , `bedrockagent.services.k8s.aws` , `bedrockagentcorecontrol.services.k8s.aws` , `cloudfront.services.k8s.aws` , `cloudtrail.services.k8s.aws` , `cloudwatch.services.k8s.aws` , `cloudwatchlogs.services.k8s.aws` , `codeartifact.services.k8s.aws` , `cognitoidentityprovider.ser`	*	*

Groupes d'API Kubernetes	Ressources Kubernetes	Verbes (autorisations) Kubernetes
vices.k8s.aws , documentdb.service s.k8s.aws , dynamodb. services.k8s.aws , ec2.services.k8s.aws , ecr.services.k8s.aws , ecrpublic.services .k8s.aws , ecs.servi ces.k8s.aws , efs.services.k8s.aws , eks.services.k8s.aws , elasticache.servic es.k8s.aws , elbv2.ser vices.k8s.aws , emrcontainers.serv ices.k8s.aws , eventbridge.servic es.k8s.aws , iam.servi ces.k8s.aws , kafka.services.k8s .aws , keyspaces .services.k8s.aws , kinesis.services.k 8s.aws , kms.servi ces.k8s.aws , lambda.services.k8 s.aws , memorydb. services.k8s.aws , mq.services.k8s.aw s , networkfirewall.se rvices.k8s.aws , opensearchservice. services.k8s.aws ,		

Groupes d'API Kubernetes	Ressources Kubernetes	Verbes (autorisations) Kubernetes
organizations.services.k8s.aws , pipes.services.k8s.aws , prometheusservice.services.k8s.aws , ram.services.k8s.aws , rds.services.k8s.aws , recyclebin.services.k8s.aws , route53.services.k8s.aws , route53resolver.services.k8s.aws , s3.services.k8s.aws , s3control.services.k8s.aws , sagemaker.services.k8s.aws , secretsmanager.services.k8s.aws , ses.services.k8s.aws , sfn.services.k8s.aws , sns.services.k8s.aws , sqs.services.k8s.aws , ssm.services.k8s.aws , wafv2.services.k8s.aws		
coordination.k8s.io	leases	create, get, list, watch
coordination.k8s.io	leases(location de contrôleurs de service ACK spécifiques uniquement)	delete, update, patch
	events	create, patch

Groupes d’API Kubernetes	Ressources Kubernetes	Verbes (autorisations) Kubernetes
`apiextensions.k8s.io`	`customresourcedefinitions`	`*`

EKSArgoCDClusterPolitique d'Amazon

ARN – `arn:aws: eks::aws:cluster-access-policy/AmazonEKSArgoCDClusterPolicy`

Cette politique accorde les autorisations nécessaires au niveau du cluster pour permettre à Argo CD de découvrir des ressources et de gérer les objets délimités au cluster. La stratégie inclut les autorisations suivantes :

Gestion de l'espace de noms :

- Créez, lisez, mettez à jour et supprimez des espaces de noms pour la gestion des espaces de noms d'applications.

Gestion des définitions de ressources personnalisées :

- Gérez les applications spécifiques à Argo CD CRDs (Applications, AppProjects, ApplicationSets).

Découverte d'API :

- Accès en lecture aux points de terminaison de l'API Kubernetes pour la découverte des ressources.

Cette politique est conçue pour prendre en charge les opérations Argo CD au niveau du cluster, notamment la gestion des espaces de noms et l'installation du CRD. Amazon EKS crée automatiquement une entrée d'accès avec cette politique d'accès pour le rôle IAM de fonctionnalité que vous fournissez lors de la création de la fonctionnalité Argo CD.

Groupes d'API Kubernetes	Kubernetes Non-Resource URLs	Ressources Kubernetes	Verbes (autorisations) Kubernetes
		namespaces	create, get, update, patch, delete
apiextensions.k8s.io		customresourcedefinitions	create
apiextensions.k8s.io		customresourcedefinitions (Argo CD CRDs uniquement)	get, update, patch, delete
	/api, /api/*, /apis, /apis/*		get

Amazon EKSArgo CDPolicy

ARN – arn:aws: eks::aws:cluster-access-policy/AmazonEKSArgoCDPolicy

Cette politique accorde les autorisations au niveau de l'espace de noms nécessaires pour que l'Argo CD puisse déployer et gérer des applications. La stratégie inclut les autorisations suivantes :

Gestion des secrets :

- Accès complet aux secrets relatifs aux informations d'identification Git et aux secrets du cluster.

ConfigMap Accès :

- Accès en lecture ConfigMaps à pour envoyer des avertissements si les clients essaient d'utiliser un Argo CD non compatible. ConfigMaps

Gestion des événements :

- Lisez et créez des événements pour le suivi du cycle de vie des applications.

Gestion des ressources Argo CD :

- Accès complet aux applications ApplicationSets, et AppProjects.
- Gérez les finaliseurs et le statut des ressources Argo CD.

Cette politique est conçue pour prendre en charge les opérations Argo CD au niveau de l'espace de noms, y compris le déploiement et la gestion des applications. Amazon EKS crée automatiquement une entrée d'accès avec cette politique d'accès pour le rôle IAM de fonctionnalité que vous fournissez lors de la création de la fonctionnalité Argo CD, dans le cadre de l'espace de noms Argo CD.

Groupes d'API Kubernetes	Ressources Kubernetes	Verbes (autorisations) Kubernetes
	`secrets`	`*`
	`configmaps`	`get`, `list`, `watch`
	`events`	`get`, `list`, `watch`, `patch`, `create`
`argoproj.io`	`applications` , `applications/finalizers` , `applications/status` , `applicationsets` , `applicationsets/finalizers` , `applicationsets/status` , `appprojects` , `appprojects/finalizers` , `appprojects/status`	`*`

Amazon EKSKROPolicy

ARN – `arn:aws: eks::aws:cluster-access-policy/AmazonEKSKROPolicy`

Cette politique accorde les autorisations nécessaires à la capacité kro (Kube Resource Orchestrator) de créer et de gérer des Kubernetes personnalisés. APIs La stratégie inclut les autorisations suivantes :

Gestion des ressources kro :

- Accès complet à toutes les ressources Kro, y compris ResourceGraphDefinitions les instances de ressources personnalisées.

Gestion des définitions de ressources personnalisées :

- Créez, lisez, mettez à jour et supprimez CRDs pour les options personnalisées APIs définies par ResourceGraphDefinitions.

Élection du chef :

- Créez et lisez des baux de coordination pour l'élection des dirigeants.
- Mettez à jour et supprimez le bail du contrôleur Kro.

Gestion des événements :

- Créez et corrigez des événements pour les opérations Kro.

Cette politique est conçue pour prendre en charge la composition complète des ressources et la gestion personnalisée des API via kro. Amazon EKS crée automatiquement une entrée d'accès avec cette politique d'accès pour le rôle IAM de fonctionnalité que vous fournissez lors de la création de la fonctionnalité kro.

Groupes d'API Kubernetes	Ressources Kubernetes	Verbes (autorisations) Kubernetes
`kro.run`	`*`	`*`
`apiextensions.k8s.io`	`customresourcedefinitions`	`*`
`coordination.k8s.io`	`leases`	`create`, `get`, `list`, `watch`

Groupes d'API Kubernetes	Ressources Kubernetes	Verbes (autorisations) Kubernetes
`coordination.k8s.io`	`leases`(location de manette Kro uniquement)	`delete`, `update`, `patch`
	`events`	`create`, `patch`

Mises à jour des stratégies d'accès

Affichez les détails sur les mises à jour des stratégies d'accès depuis leur introduction. Pour recevoir des alertes automatiques concernant les modifications apportées à cette page, abonnez-vous au flux RSS de la page Historique du document.

Modifier	Description	Date
Ajouter des politiques pour les fonctionnalités EKS	Publier`AmazonEKS ACKPolicy` , `AmazonEKS ArgoCDClusterPolic y` `AmazonEKSArgoCDPol icy` , et `AmazonEKS KROPolicy` pour gérer les fonctionnalités d'EKS	22 novembre 2025
Addition `AmazonEKS SecretReaderPolicy`	Ajouter une nouvelle politique pour l'accès en lecture seule aux secrets	6 novembre 2025
Ajout de la stratégie pour EKS Cluster Insights	Publication `AmazonEKS ClusterInsightsPol icy`	2 décembre 2024
Ajout des stratégies pour Amazon EKS Hybrid	Publication `AmazonEKS HybridPolicy`	2 décembre 2024
Ajout des stratégies pour le mode automatique Amazon EKS	Ces politiques d'accès autorisent le rôle IAM du cluster et le rôle IAM du	2 décembre 2024

Modifier	Description	Date
	nœud à appeler Kubernetes. APIs AWS les utilise pour automatiser les tâches de routine relatives aux ressources de stockage, de calcul et de réseau.	
Addition `AmazonEKSAdminViewPolicy`	Ajout d'une nouvelle stratégie pour étendre les autorisations de visualisation, incluant l'accès à des ressources comme Secrets.	23 avril 2024
Stratégies d'accès introduites.	Amazon EKS a introduit des stratégies d'accès.	29 mai 2023

Modifier le mode d'authentification pour utiliser les entrées d'accès

Pour commencer à utiliser les entrées d'accès, vous devez remplacer le mode d'authentification du cluster par les modes `API_AND_CONFIG_MAP` ou `API`. Cela ajoute l'API pour les entrées d'accès.

Console AWS

1. Ouvrez la console Amazon EKS.
2. Choisissez le nom du cluster pour lequel vous souhaitez créer une entrée d'accès.
3. Choisissez l'onglet Access.
4. Le Mode d'authentification indique le mode d'authentification actuel du cluster. Si le mode indique API EKS, vous pouvez déjà ajouter des entrées d'accès et ignorer les étapes restantes.
5. Choisissez Gérer l'accès.
6. Pour le Mode d'authentification du cluster, sélectionnez un mode avec l'API EKS. Veuillez noter que vous ne pouvez pas revenir à un mode qui supprime l'API EKS et les entrées d'accès.
7. Sélectionnez Enregistrer les modifications. Amazon EKS commence à mettre à jour le cluster, le statut du cluster passe à Mise à jour en cours et la modification est enregistrée dans l'onglet Historique des mises à jour.

8. Veuillez patienter jusqu'à ce que le statut du cluster revienne à « Actif ». Lorsque le cluster est actif, vous pouvez suivre les étapes décrites dans the section called "Créer des entrées d'accès" pour ajouter un accès au cluster pour les principaux IAM.

AWS CLI

1. Installez l'AWS CLI, comme indiqué dans la section Installation du Guide de l'utilisateur de l'interface de la ligne de commande AWS.
2. Exécutez la commande suivante. Remplacez *my-cluster* par le nom de votre cluster. Si vous souhaitez désactiver définitivement la méthode `ConfigMap`, remplacez `API_AND_CONFIG_MAP` par `API`.

   Amazon EKS commence à mettre à jour le cluster, le statut du cluster passe à MISE À JOUR EN COURS et la modification est enregistrée dans la liste aws eks list-updates.

   ```
   aws eks update-cluster-config --name my-cluster --access-config
    authenticationMode=API_AND_CONFIG_MAP
   ```
3. Veuillez patienter jusqu'à ce que le statut du cluster revienne à « Actif ». Lorsque le cluster est actif, vous pouvez suivre les étapes décrites dans the section called "Créer des entrées d'accès" pour ajouter un accès au cluster pour les principaux IAM.

Version de plateforme requise

Pour utiliser les entrées d'accès, le cluster doit disposer d'une version de plateforme identique ou supérieure à celle indiquée dans le tableau suivant, ou d'une version de Kubernetes supérieure à celles indiquées dans le tableau. Si votre version de Kubernetes ne figure pas dans le tableau, toutes les versions de la plateforme prennent en charge les entrées d'accès.

Version de Kubernetes	Version de la plateforme
Non répertoriée	Tous pris en charge
`1.30`	`eks.2`
`1.29`	`eks.1`
`1.28`	`eks.6`

Pour plus d'informations, consultez platform-versions.

Créer des entrées d'accès

Avant de créer des entrées d'accès, tenez compte des éléments suivants :

- Un mode d'authentification correctement configuré. Consultez the section called "Mode d'authentification".
- Une entrée d'accès inclut l'Amazon Resource Name (ARN) d'un et d'un seul principal IAM existant. Un principal IAM ne peut pas être inclus dans plus d'une entrée d'accès. Considérations supplémentaires concernant l'ARN que vous spécifiez :
  - Les bonnes pratiques IAM recommandent d'accéder à votre cluster en utilisant des rôles IAM dotés d'informations d'identification à court terme, plutôt que des utilisateurs IAM dotés d'informations d'identification à long terme. Pour plus d'informations, consultez Exiger que les utilisateurs humains utilisent la fédération avec un fournisseur d'identité pour accéder à AWS à l'aide d'informations d'identification temporaires dans le Guide d'utilisateur IAM.
  - Si l'ARN est destiné à un rôle IAM, il peut inclure un chemin. Les ARN dans les entrées `aws-auth ConfigMap` ne peuvent pas inclure de chemin. Par exemple, votre ARN peut être `arn:aws:iam::<111122223333>:role/<development/apps/my-role>` ou `arn:aws:iam::<111122223333>:role/<my-role>`.
  - Si le type de l'entrée d'accès est différent de `STANDARD` (voir la considération suivante concernant les types), l'ARN doit appartenir au même compte AWS que celui du cluster. Si le type est `STANDARD`, l'ARN peut appartenir soit au même compte AWS, soit à un autre compte que celui du cluster.
  - Vous ne pouvez pas modifier le principal IAM après la création de l'entrée d'accès.
  - Si vous supprimez le principal IAM correspondant à cet ARN, l'entrée d'accès n'est pas supprimée automatiquement. Nous vous recommandons de supprimer l'entrée d'accès avec un ARN pour un principal IAM que vous supprimez. Si vous ne supprimez pas l'entrée d'accès et que vous recréez plus tard le principal IAM, même avec le même ARN, l'entrée d'accès ne fonctionnera pas. Cela s'explique par le fait que, bien que l'ARN soit identique pour le principal IAM recréé, le `roleID` ou le `userID` (consultable avec la commande AWS CLI `aws sts get-caller-identity`) du principal IAM recréé est différent de celui du principal IAM d'origine. Même si le `roleID` ou le `userID` du principal IAM ne s'affiche pas dans l'entrée d'accès, Amazon EKS le conserve en interne avec l'entrée d'accès.

- Chaque entrée d'accès possède un type. Le type de l'entrée d'accès dépend du type de ressource associée ; il ne définit pas les autorisations. Si vous ne spécifiez pas de type, Amazon EKS définit automatiquement le type sur `STANDARD`
  - `EC2_LINUX` : pour un rôle IAM utilisé avec des nœuds autogérés Linux ou Bottlerocket
  - `EC2_WINDOWS` : pour un rôle IAM utilisé avec des nœuds autogérés Windows
  - `FARGATE_LINUX` : pour un rôle IAM utilisé avec AWS Fargate (Fargate)
  - `HYBRID_LINUX` : pour un rôle IAM utilisé avec des nœuds hybrides
  - `STANDARD` : type par défaut si aucun n'est spécifié
  - `EC2` : pour les classes de nœuds personnalisées du mode automatique EKS. Pour de plus amples informations, consultez the section called "Création d'une entrée d'accès pour la classe de nœuds".
  - Vous ne pouvez pas modifier le type après la création de l'entrée d'accès.
- Il n'est pas nécessaire de créer une entrée d'accès pour un rôle IAM utilisé par un groupe de nœuds géré ou par un profil Fargate. EKS créera des entrées d'accès (si cette option est activée) ou mettra à jour la ConfigMap d'authentification (si les entrées d'accès ne sont pas disponibles)
- Si le type d'entrée d'accès est `STANDARD`, vous pouvez spécifier un nom d'utilisateur pour l'entrée d'accès. Si vous ne spécifiez aucune valeur pour le nom d'utilisateur, Amazon EKS en attribue automatiquement une, selon le type d'entrée d'accès et selon que le principal IAM indiqué est un rôle IAM ou un utilisateur IAM. Sauf si vous avez une raison précise de définir votre propre nom d'utilisateur, il est recommandé de ne pas en spécifier et de laisser Amazon EKS le générer automatiquement. Si vous spécifiez votre propre nom d'utilisateur :
  - Il ne peut pas commencer par `system:`, `eks:`, `aws:`, `amazon:` ou `iam:`.
  - Si le nom d'utilisateur s'applique à un rôle IAM, il est recommandé d'ajouter `{{SessionName}}` ou `{{SessionNameRaw}}` à la fin de votre nom d'utilisateur. Si vous ajoutez `{{SessionName}}` ou `{{SessionNameRaw}}` à votre nom d'utilisateur, alors le nom d'utilisateur doit comporter un deux-points avant {{SessionName}}. Lorsque ce rôle est assumé, le nom de la session AWS STS utilisé au moment de l'assumer est automatiquement transmis au cluster et apparaîtra dans les journaux CloudTrail. Par exemple, vous ne pouvez pas avoir le nom d'utilisateur `john{{SessionName}}`. Le nom d'utilisateur doit être `:john{{SessionName}}` ou `jo:hn{{SessionName}}`. Il suffit que les deux-points soient devant `{{SessionName}}`. Le nom d'utilisateur généré par Amazon EKS dans le tableau suivant inclut un ARN. Comme un ARN inclut les deux-points, il répond à cette exigence. Le deux-points n'est pas requis si vous n'incluez pas `{{SessionName}}` dans votre nom d'utilisateur. Notez que dans `{{SessionName}}`, le caractère spécial « @ » est remplacé par

« - » dans le nom de session. `{{SessionNameRaw}}` conserve les caractères spéciaux dans le nom de session.

Type du principal IAM	Type	Valeur du nom d'utilisateur définie automatiquement par Amazon EKS
Utilisateur	`STANDARD`	ARN de l'utilisateur IAM. Exemple : `arn:aws:iam::<111122223333>:user/<my-user>`
Rôle	`STANDARD`	L'ARN STS du rôle lorsqu'il est assumé. Amazon EKS ajoute `{{SessionName}}` au rôle. Exemple : `arn:aws:sts::<111122223333>:assumed-role/<my-role>/{{SessionName}}` Si l'ARN du rôle que vous avez spécifié contenait un chemin, Amazon EKS le supprime dans le nom d'utilisateur généré.
Rôle	`EC2_LINUX` ou `EC2_Windows`	`system:node:{{EC2PrivateDNSName}}`
Rôle	`FARGATE_LINUX`	`system:node:{{SessionName}}`
Rôle	`HYBRID_LINUX`	`system:node:{{SessionName}}`

Vous ne pouvez pas modifier le nom d'utilisateur une fois l'entrée d'accès créée.

- Si le type de l'entrée d'accès est `STANDARD` et que vous souhaitez utiliser l'autorisation Kubernetes RBAC, vous pouvez ajouter un ou plusieurs noms de groupes à l'entrée d'accès. Après avoir créé une entrée d'accès, vous pouvez ajouter et supprimer des noms de groupes. Pour permettre au principal IAM d'accéder aux objets Kubernetes de votre cluster, vous devez créer et gérer les ressources d'autorisation Kubernetes basées sur les rôles (RBAC). Créez des objets Kubernetes `RoleBinding` ou `ClusterRoleBinding` sur votre cluster qui spécifient le nom du groupe sous forme de `subject` pour `kind: Group`. Kubernetes autorise le principal IAM à accéder aux objets du cluster uniquement si ceux-ci sont définis dans un objet `Role` ou `ClusterRole`, et si ce rôle est spécifié dans l'objet `roleRef` de votre liaison. Si vous spécifiez des noms de groupe, il est recommandé de bien maîtriser objets d'autorisation basée sur les rôles (RBAC) . Pour plus d'informations, consultez Utilisation de l'autorisation RBAC dans la documentation Kubernetes.

  > **Important**
  >
  > Amazon EKS ne vérifie pas si des objets RBAC Kubernetes existants dans votre cluster incluent les noms de groupes que vous spécifiez. Par exemple, si vous créez une entrée d'accès pour un groupe qui n'existe pas encore, EKS créera ce groupe au lieu de renvoyer une erreur.

  Au lieu ou en complément de l'autorisation Kubernetes pour permettre au principal IAM d'accéder aux objets Kubernetes de votre cluster, vous pouvez associer des stratégies d'accès Amazon EKS à une entrée d'accès. Amazon EKS autorise le principal IAM à accéder aux objets Kubernetes du cluster en fonction des autorisations définies dans la stratégie d'accès. Vous pouvez limiter la portée d'une stratégie d'accès à certains espaces de noms Kubernetes que vous spécifiez. L'utilisation de stratégies d'accès ne nécessite pas de gérer des objets RBAC Kubernetes. Pour de plus amples informations, consultez the section called "Association des stratégies d'accès".
- Si vous créez une entrée d'accès avec le type `EC2_LINUX` ou `EC2_Windows`, le principal IAM qui crée l'entrée d'accès doit avoir l'autorisation `iam:PassRole`. Pour plus d'informations, consultez la section Octroi d'autorisations à un utilisateur pour transférer un rôle à un service AWS dans le IAM Guide de l'utilisateur.
- À l'instar du comportement IAM standard, la création et les mises à jour des entrées d'accès sont finalement cohérentes et prennent effet au bout de plusieurs secondes après que l'appel API initial a été renvoyé avec succès. Vous devez concevoir vos applications de sorte qu'elles tiennent

compte de ces retards potentiels. Nous recommandons de ne pas inclure la création ou la mise à jour d'entrées d'accès dans les parcours applicatifs critiques ou nécessitant une haute disponibilité. Au lieu de cela, procédez aux modifications dans une routine d'initialisation ou d'installation distincte que vous exécutez moins souvent. Veillez également à vérifier que les modifications ont été propagées avant que les processus de production en dépendent.

- Les entrées d'accès ne prennent pas en charge les rôles liés aux services. Vous ne pouvez pas créer d'entrée d'accès lorsque l'ARN du principal correspond à un rôle lié à un service. Vous pouvez identifier les rôles liés aux services par leur ARN, qui est au format `arn:aws:iam::*:role/aws-service-role/*`.

Vous pouvez créer une entrée d'accès à l'aide de la AWS Management Console ou de la CLI AWS.

AWS Management Console

1. Ouvrez la console Amazon EKS.
2. Choisissez le nom du cluster pour lequel vous souhaitez créer une entrée d'accès.
3. Choisissez l'onglet Access.
4. Choisissez Créer une entrée d'accès.
5. Pour le principal IAM, sélectionnez un utilisateur ou un rôle IAM existant. Les bonnes pratiques IAM recommandent d'accéder à votre cluster en utilisant des rôles IAM dotés d'informations d'identification à court terme, plutôt que des utilisateurs IAM dotés d'informations d'identification à long terme. Pour plus d'informations, consultez Exiger que les utilisateurs humains utilisent la fédération avec un fournisseur d'identité pour accéder à AWS à l'aide d'informations d'identification temporaires dans le Guide d'utilisateur IAM.
6. Pour Type, si l'entrée d'accès concerne le rôle de nœud utilisé pour les nœuds Amazon EC2 autogérés, sélectionnez EC2 Linux ou EC2 Windows. Dans le cas contraire, acceptez le type par défaut (Standard).
7. Si le type que vous avez choisi est Standard et que vous souhaitez spécifier un nom d'utilisateur, saisissez-le.
8. Si le Type que vous avez sélectionné Standard et que vous souhaitez utiliser l'autorisation Kubernetes basée sur RBAC pour le principal IAM, spécifiez un ou plusieurs noms dans le champ Groupes. Si vous ne spécifiez aucun groupe et que vous préférez utiliser l'autorisation Amazon EKS, vous pourrez associer une stratégie d'accès à une étape ultérieure ou après la création de l'entrée d'accès.

9. (Facultatif) Pour Balises, attribuez des étiquettes à l'entrée d'accès. Par exemple, pour faciliter la recherche de toutes les ressources portant la même balise.
10. Choisissez Suivant.
11. Sur la page Ajouter une stratégie d'accès, si le type que vous avez sélectionné était Standard et que vous souhaitez qu'Amazon EKS autorise le principal IAM à accéder aux objets Kubernetes du cluster, suivez les étapes ci-dessous. Sinon, choisissez Next (Suivant).
    a. Pour Nom de la politique, choisissez une stratégie d'accès. Vous ne pouvez pas consulter les autorisations des stratégies d'accès, mais celles-ci incluent des autorisations similaires à celles des objets `ClusterRole` destinés aux utilisateurs Kubernetes. Pour plus d'informations, consultez la section Rôles destinés aux utilisateurs dans la documentation Kubernetes.
    b. Choisissez l'une des options suivantes :
        - Cluster : sélectionnez cette option si vous souhaitez qu'Amazon EKS autorise le principal IAM à disposer des autorisations définies dans la stratégie d'accès pour l'ensemble des objets Kubernetes du cluster.
        - Espace de noms Kubernetes : sélectionnez cette option si vous souhaitez qu'Amazon EKS autorise le principal IAM à disposer des autorisations définies dans la stratégie d'accès, mais uniquement pour les objets Kubernetes d'un espace de noms spécifique du cluster. Dans le champ Espace de noms, saisissez le nom de l'espace de noms Kubernetes présent sur votre cluster. Si vous souhaitez ajouter des espaces de noms supplémentaires, choisissez Ajouter un nouvel espace de noms et entrez son nom.
    c. Pour ajouter des politiques supplémentaires, Sélectionnez Ajouter une politique. Vous pouvez définir la portée de chaque politique différemment, mais vous ne pouvez ajouter chaque politique qu'une seule fois.
    d. Choisissez Suivant.
12. Vérifiez la configuration de votre entrée d'accès. Si quelque chose semble incorrect, choisissez Précédent pour revenir en arrière et corriger l'erreur. Si la configuration est correcte, choisissez Créer.

AWS CLI

1. Installez l'AWS CLI, comme indiqué dans la section Installation du Guide de l'utilisateur de l'interface de la ligne de commande AWS.
2. Vous pouvez utiliser n'importe lequel des exemples ci-dessous pour créer des entrées d'accès :

- Créez une entrée d'accès pour un groupe de nœuds Linux Amazon EC2 autogéré. Remplacez *my-cluster* par le nom de votre cluster, *111122223333* par votre ID de compte AWS, et *EKS-my-cluster-self-managed-ng-1* par le nom de votre rôle IAM de nœud. Si votre groupe de nœuds est un groupe de nœuds Windows, remplacez *EC2_LINUX* par EC2_Windows.

```
aws eks create-access-entry --cluster-name my-cluster --principal-arn
 arn:aws:iam::111122223333:role/EKS-my-cluster-self-managed-ng-1 --type EC2_LINUX
```

  Vous ne pouvez pas utiliser l'option --kubernetes-groups lorsque vous spécifiez un type autre que STANDARD. Vous ne pouvez pas associer de stratégie d'accès à cette entrée d'accès, car son type est une valeur autre que STANDARD.

- Créez une entrée d'accès permettant à un rôle IAM qui n'est pas utilisé pour un groupe de nœuds autogéré Amazon EC2 d'être autorisé par Kubernetes à accéder à votre cluster. Remplacez *my-cluster* par le nom de votre cluster, *111122223333* par votre ID de compte AWS, et *my-role* par le nom de votre rôle IAM. Remplacez *Utilisateurs* par le nom d'un groupe que vous avez défini dans un objet Kubernetes RoleBinding ou ClusterRoleBinding de votre cluster.

```
aws eks create-access-entry --cluster-name my-cluster --principal-arn
 arn:aws:iam::111122223333:role/my-role --type STANDARD --user Viewers --
kubernetes-groups Viewers
```

- Créez une entrée d'accès qui permet à un utilisateur IAM de s'authentifier auprès de votre cluster. Cet exemple est fourni car c'est possible, bien que les bonnes pratiques IAM recommandent d'accéder à votre cluster en utilisant des rôles IAM dotés d'informations d'identification à court terme, plutôt que des utilisateurs IAM dotés d'informations d'identification à long terme. Pour plus d'informations, consultez Exiger que les utilisateurs humains utilisent la fédération avec un fournisseur d'identité pour accéder à AWS à l'aide d'informations d'identification temporaires dans le Guide d'utilisateur IAM.

```
aws eks create-access-entry --cluster-name my-cluster --principal-arn
 arn:aws:iam::111122223333:user/my-user --type STANDARD --username my-user
```

  Si vous souhaitez que cet utilisateur ait davantage d'accès au cluster que les autorisations fournies par les rôles de découverte de l'API Kubernetes, vous devez associer une stratégie d'accès à l'entrée d'accès, puisque l'option --kubernetes-groups n'est pas utilisée. Pour

plus d'informations, consultez the section called "Association des stratégies d'accès" et Rôles de découverte d'API dans la documentation Kubernetes.

Mettre à jour les entrées d'accès

Vous pouvez mettre à jour une entrée d'accès à l'aide de la AWS Management Console ou de l'AWS CLI.

AWS Management Console

1. Ouvrez la console Amazon EKS.
2. Choisissez le nom du cluster pour lequel vous souhaitez créer une entrée d'accès.
3. Choisissez l'onglet Access.
4. Choisissez l'entrée d'accès que vous voulez mettre à jour.
5. Choisissez Modifier.
6. Pour Nom d'utilisateur, vous pouvez modifier la valeur existante.
7. Pour Groupes, vous pouvez supprimer les noms de groupes existants ou en ajouter de nouveaux. Si les noms de groupes suivants existent, veuillez ne pas les supprimer : system:nodes ou system:bootstrappers. La suppression de ces groupes peut entraîner un dysfonctionnement de votre cluster. Si vous ne spécifiez aucun nom de groupe et que vous voulez utiliser l'autorisation Amazon EKS, associez une stratégie d'accès dans une étape ultérieure.
8. Pour Balises, vous pouvez attribuer des étiquettes à l'entrée d'accès. Par exemple, pour faciliter la recherche de toutes les ressources portant la même balise. Vous pouvez également supprimer des balises existantes.
9. Sélectionnez Enregistrer les modifications.
10. Si vous souhaitez associer une stratégie d'accès à l'entrée, consultez the section called "Association des stratégies d'accès".

AWS CLI

1. Installez l'AWS CLI, comme indiqué dans la section Installation du Guide de l'utilisateur de l'interface de la ligne de commande AWS.
2. Pour mettre à jour une entrée d'accès, remplacez *my-cluster* par le nom de votre cluster, *111122223333* par votre ID de compte AWS et *EKS-my-cluster-my-namespace-Viewers* par le nom d'un rôle IAM.

```
aws eks update-access-entry --cluster-name my-cluster --principal-arn
 arn:aws:iam::111122223333:role/EKS-my-cluster-my-namespace-Viewers --kubernetes-
groups Viewers
```

Vous ne pouvez pas utiliser l'option `--kubernetes-groups` si le type de l'entrée d'accès est une valeur autre que `STANDARD`. Vous ne pouvez pas non plus associer une stratégie d'accès à une entrée d'accès dont le type est autre que `STANDARD`.

Supprimer les entrées d'accès

Si vous découvrez que vous avez supprimé une entrée d'accès par erreur, vous pouvez toujours la recréer. Si l'entrée d'accès que vous supprimez est associée à des stratégies d'accès, les associations sont automatiquement supprimées. Il n'est pas nécessaire de dissocier les stratégies d'accès d'une entrée d'accès avant de supprimer cette entrée d'accès.

Vous pouvez supprimer une entrée d'accès à l'aide de la AWS Management Console ou de la CLI AWS.

AWS Management Console

1. Ouvrez la console Amazon EKS.
2. Choisissez le nom du cluster dont vous souhaitez supprimer une entrée d'accès.
3. Choisissez l'onglet Access.
4. Dans la liste des entrées d'accès, choisissez l'entrée d'accès que vous souhaitez supprimer.
5. Sélectionnez Delete.
6. Dans la boîte de dialogue de confirmation, choisissez Delete (Supprimer).

AWS CLI

1. Installez l'AWS CLI, comme indiqué dans la section Installation du Guide de l'utilisateur de l'interface de la ligne de commande AWS.
2. Pour supprimer une entrée d'accès Remplacez *`my-cluster`* par le nom de votre cluster, *`111122223333`* par votre ID de compte AWS et *`my-role`* par le nom du rôle IAM auquel vous ne souhaitez plus donner accès à votre cluster.

```
aws eks delete-access-entry --cluster-name my-cluster --principal-arn
 arn:aws:iam::111122223333:role/my-role
```

Définir un nom d'utilisateur personnalisé pour les entrées d'accès Amazon EKS

Lors de la création d'entrées d'accès pour Amazon EKS, vous pouvez soit utiliser le nom d'utilisateur généré automatiquement, soit spécifier un nom d'utilisateur personnalisé. Cette page explique les deux options et vous guide dans la configuration d'un nom d'utilisateur personnalisé.

Présentation de

Le nom d'utilisateur dans une entrée d'accès est utilisé pour identifier le principal IAM dans les journaux et les pistes d'audit Kubernetes. Par défaut, Amazon EKS génère un nom d'utilisateur basé sur l'ARN de l'identité IAM, mais vous pouvez spécifier un nom d'utilisateur personnalisé si nécessaire.

Génération du nom d'utilisateur par défaut

Si vous ne spécifiez pas de valeur pour le nom d'utilisateur, Amazon EKS génère automatiquement un nom d'utilisateur basé sur l'identité IAM :

- Pour les utilisateurs IAM :
  - EKS définit le nom d'utilisateur Kubernetes sur l'ARN de l'utilisateur IAM
  - Exemple :

    ```
    {arn-aws}iam::<111122223333>:user/<my-user>
    ```
- Pour les rôles IAM :
  - EKS définit le nom d'utilisateur Kubernetes en fonction de l'ARN du rôle IAM
  - L'ARN STS du rôle lorsqu'il est assumé. Amazon EKS ajoute `{{SessionName}}` au rôle. Si l'ARN du rôle que vous avez spécifié contenait un chemin, Amazon EKS le supprime dans le nom d'utilisateur généré.
  - Exemple :

    ```
    {arn-aws}sts::<111122223333>:assumed-role/<my-role>/{{SessionName}}
    ```

À moins que vous n'ayez une raison particulière de spécifier votre propre nom d'utilisateur, nous vous recommandons de ne pas en spécifier et de laisser Amazon EKS le générer automatiquement pour vous.

Définition d'un nom d'utilisateur personnalisé

Lors de la création d'une entrée d'accès, vous pouvez spécifier un nom d'utilisateur personnalisé à l'aide du paramètre `--username` :

```
aws eks create-access-entry --cluster-name <cluster-name> --principal-arn <iam-
identity-arn> --type STANDARD --username <custom-username>
```

Exigences relatives aux noms d'utilisateur personnalisés

Si vous spécifiez un nom d'utilisateur personnalisé :

- Le nom d'utilisateur ne peut pas commencer par `system:`, `eks:`, `aws:`, `amazon:` ou `iam:`.
- Si le nom d'utilisateur s'applique à un rôle IAM, il est recommandé d'ajouter `{{SessionName}}` ou `{{SessionNameRaw}}` à la fin de votre nom d'utilisateur.
  - Si vous ajoutez l'un `{{SessionName}}` ou `{{SessionNameRaw}}` l'autre à votre nom d'utilisateur, celui-ci doit inclure deux points avant {{SessionName}}.

Création d'une entrée d'accès pour un rôle ou un utilisateur IAM à l'aide d'une stratégie d'accès et de la CLI AWS

Créez des entrées d'accès Amazon EKS qui utilisent des stratégies d'accès EKS gérées par AWS afin d'accorder aux identités IAM des autorisations standardisées pour accéder aux clusters Kubernetes et les gérer.

Présentation

Les entrées d'accès dans Amazon EKS définissent la manière dont les identités IAM (utilisateurs et rôles) peuvent accéder à vos clusters Kubernetes et interagir avec eux. En créant des entrées d'accès avec des stratégies d'accès EKS, vous pouvez :

- Accorder à des utilisateurs ou rôles IAM spécifiques l'accès à votre cluster EKS
- Contrôler les autorisations à l'aide des stratégies d'accès EKS gérées par AWS, qui offrent des ensembles d'autorisations standardisés prédéfinis

- Limiter les autorisations à des espaces de noms spécifiques ou à l'échelle du cluster
- Simplifier la gestion des accès sans modifier le ConfigMap `aws-auth` ni créer de ressources RBAC Kubernetes
- Utiliser une méthode d'accès à Kubernetes intégrée à AWS, couvrant les cas d'utilisation courants tout en respectant les bonnes pratiques de sécurité

Cette méthode est recommandée pour la plupart des cas d'utilisation, car elle fournit des autorisations standardisées gérées par AWS, sans nécessiter de configuration manuelle de RBAC Kubernetes. Les stratégies d'accès EKS éliminent la nécessité de configurer manuellement les ressources RBAC Kubernetes et offrent des ensembles d'autorisations prédéfinis couvrant les cas d'utilisation les plus fréquents.

Prérequis

- Le mode d'authentification de votre cluster doit être configuré pour autoriser les entrées d'accès. Pour de plus amples informations, consultez the section called "Mode d'authentification".
- Installez l'AWS CLI, comme indiqué dans la section Installation du Guide de l'utilisateur de l'interface de la ligne de commande AWS.

Étape 1 : définir l'entrée d'accès

1. Trouvez l'ARN de l'identité IAM (utilisateur ou rôle) à laquelle vous souhaitez accorder des autorisations.
   - Chaque identité IAM ne peut avoir qu'une seule entrée d'accès EKS.
2. Déterminez si vous souhaitez que les autorisations de la stratégie d'accès Amazon EKS s'appliquent uniquement à un espace de noms Kubernetes spécifique ou à l'ensemble du cluster.
   - Si vous souhaitez limiter les autorisations à un espace de noms spécifique, notez le nom de l'espace de noms.
3. Sélectionnez la stratégie d'accès EKS que vous souhaitez appliquer à l'identité IAM. Cette stratégie accorde des autorisations à l'intérieur du cluster. Notez l'ARN de la stratégie.
   - Pour obtenir la liste des stratégie, consultez les stratégies d'accès disponibles.
4. Déterminez si le nom d'utilisateur généré automatiquement convient à l'entrée d'accès ou si vous devez spécifier manuellement un nom d'utilisateur.
   - AWS génère automatiquement cette valeur en fonction de l'identité IAM. Vous pouvez définir un nom d'utilisateur personnalisé. Cela est visible dans les journaux Kubernetes.

- Pour de plus amples informations, consultez the section called "Définir un nom d'utilisateur personnalisé".

Étape 2 : créer une entrée d'accès

Une fois l'entrée d'accès planifiée, utilisez la CLI AWS pour la créer.

L'exemple ci-dessous couvre la majorité des cas d'utilisation. Affichez la référence de la CLI pour toutes les options de configuration.

Vous attacherez la stratégie d'accès à l'étape suivante.

```
aws eks create-access-entry --cluster-name <cluster-name> --principal-arn <iam-
identity-arn> --type STANDARD
```

Étape 3 : attacher la stratégie d'accès

La commande varie selon que vous souhaitiez ou non limiter la stratégie d'accès à un espace de noms Kubernetes.

Vous aurez besoin de l'ARN de la stratégie d'accès. Consultez les stratégies d'accès disponibles.

Création d'une stratégie sans limitation de la portée à un espace de noms

```
aws eks associate-access-policy --cluster-name <cluster-name> --principal-arn <iam-
identity-arn> --policy-arn <access-policy-arn>
```

Création d'une stratégie avec limitation de la portée à un espace de noms

```
aws eks associate-access-policy --cluster-name <cluster-name> --principal-arn <iam-
identity-arn> \
    --access-scope type=namespace,namespaces=my-namespace1,my-namespace2 --policy-arn
 <access-policy-arn>
```

Étapes suivantes

- Création d'un fichier kubeconfig pour utiliser kubectl avec une identité IAM

Création d'une entrée d'accès à l'aide des groupes Kubernetes avec la CLI AWS

Créer des entrées d'accès Amazon EKS qui utilisent les groupes Kubernetes pour l'autorisation et nécessitent une configuration RBAC manuelle.

> **Note**
>
> Dans la plupart des cas d'utilisation, nous recommandons d'utiliser les stratégies d'accès EKS plutôt que la méthode basée sur les groupes Kubernetes décrite dans cette page. Les stratégies d'accès EKS offrent une méthode plus simple et mieux intégrée à AWS pour gérer l'accès sans nécessiter de configuration RBAC manuelle. Utilisez l'approche basée sur les groupes Kubernetes uniquement lorsque vous avez besoin d'un contrôle plus granulaire que celui offert par les stratégies d'accès EKS.

Présentation

Les entrées d'accès définissent la manière dont les identités IAM (utilisateurs et rôles) accèdent à vos clusters Kubernetes. L'approche basée sur les groupes Kubernetes accorde aux utilisateurs ou rôles IAM l'autorisation d'accéder à votre cluster EKS via les groupes RBAC standard de Kubernetes. Cette méthode implique la création et la gestion des ressources RBAC Kubernetes (Roles, RoleBindings, ClusterRoles et ClusterRoleBindings) et est recommandée lorsque vous avez besoin d'ensembles d'autorisations très personnalisés, d'exigences d'autorisation complexes ou que vous souhaitez maintenir des modèles de contrôle d'accès cohérents dans des environnements Kubernetes hybrides.

Cette rubrique ne couvre pas la création d'entrées d'accès pour les identités IAM utilisées par les instances Amazon EC2 afin de rejoindre les clusters EKS.

Prérequis

- Le mode d'authentification de votre cluster doit être configuré pour autoriser les entrées d'accès. Pour de plus amples informations, consultez the section called "Mode d'authentification".
- Installez l'AWS CLI, comme indiqué dans la section Installation du Guide de l'utilisateur de l'interface de la ligne de commande AWS.
- Une bonne connaissance de la configuration RBAC Kubernetes est recommandée. Pour plus d'informations, consultez Utilisation de l'autorisation RBAC dans la documentation Kubernetes.

Étape 1 : définir l'entrée d'accès

1. Recherchez l'ARN de l'identité IAM, comme un utilisateur ou un rôle, à laquelle vous souhaitez accorder des autorisations.
   - Chaque identité IAM ne peut avoir qu'une seule entrée d'accès EKS.
2. Déterminez les groupes Kubernetes que vous souhaitez associer à cette identité IAM.
   - Vous devrez créer ou utiliser des ressources Kubernetes `Role`/`ClusterRole` et `RoleBinding`/`ClusterRoleBinding` qui référencent ces groupes.
3. Déterminez si le nom d'utilisateur généré automatiquement convient à l'entrée d'accès ou si vous devez spécifier manuellement un nom d'utilisateur.
   - AWS génère automatiquement cette valeur en fonction de l'identité IAM. Vous pouvez définir un nom d'utilisateur personnalisé. Cela est visible dans les journaux Kubernetes.
   - Pour de plus amples informations, consultez the section called "Définir un nom d'utilisateur personnalisé".

Étape 2 : créer une entrée d'accès avec des groupes Kubernetes

Après avoir planifié l'entrée d'accès, utilisez la CLI AWS pour la créer avec les groupes Kubernetes appropriés.

```
aws eks create-access-entry --cluster-name <cluster-name> --principal-arn <iam-identity-arn> --type STANDARD --kubernetes-groups <groups>
```

Remplacez :

- `<cluster-name>` par le nom de votre cluster EKS
- `<iam-identity-arn>` par l'ARN de l'utilisateur ou du rôle IAM
- `<groups>` par une liste de groupes Kubernetes séparés par des virgules (par exemple : « system:developers,system:readers »)

Affichez la référence de la CLI pour toutes les options de configuration.

Étape 3 : configurer RBAC Kubernetes

Pour permettre au principal IAM d'accéder aux objets Kubernetes de votre cluster, vous devez créer et gérer les objets de contrôle d'accès basé sur les rôles (RBAC) dans Kubernetes :

1. Créez des objets Kubernetes `Role` ou `ClusterRole` qui définissent les autorisations.
2. Créez des objets Kubernetes `RoleBinding` ou `ClusterRoleBinding` sur votre cluster qui spécifient le nom du groupe sous forme de `subject` pour `kind: Group`.

Pour des informations détaillées sur la configuration des groupes et des autorisations dans Kubernetes, consultez la section Utilisation de l'autorisation RBAC dans la documentation Kubernetes.

Étapes suivantes

- Création d'un fichier kubeconfig pour utiliser kubectl avec une identité IAM

Accordez aux utilisateurs IAM l'accès à Kubernetes avec un ConfigMap

> **Important**
>
> `aws-auth ConfigMap` est obsolète. Pour connaître la méthode recommandée pour gérer l'accès à Kubernetes APIs, consultez. the section called "Entrées d'accès"

L'accès à votre cluster à l'aide des principaux IAM est activé par l'authentificateur AWS IAM pour Kubernetes, qui s'exécute sur le plan de contrôle Amazon EKS. L'authentificateur obtient ses informations de configuration à partir du `aws-auth ConfigMap`. Pour tous les paramètres `aws-auth ConfigMap`, consultez Format de configuration complet sur GitHub.

Ajout de principaux IAM à votre cluster Amazon EKS

Lorsque vous créez un cluster Amazon EKS, le principal IAM qui crée le cluster se voit automatiquement accorder des autorisations `system:masters` dans la configuration du contrôle d'accès basé sur les rôles (RBAC) du cluster dans le plan de contrôle Amazon EKS. Ce principal n'apparaît dans aucune configuration visible. Souvenez-vous donc du principal qui a créé le cluster à l'origine. Pour accorder à d'autres principaux IAM la possibilité d'interagir avec votre cluster, modifiez le `aws-auth ConfigMap` dans Kubernetes et créez un `rolebinding` ou `clusterrolebinding` Kubernetes associé au nom d'un `group` que vous spécifiez dans le `aws-auth ConfigMap`.

> **Note**
>
> Pour plus d'informations sur la configuration du contrôle d'accès basé sur les rôles (RBAC) de Kubernetes, consultez la section Utilisation de l'autorisation RBAC dans la documentation Kubernetes.

1. Déterminer les informations d'identification que `kubectl` utilise pour accéder à votre cluster. Sur votre ordinateur, vous pouvez voir quelles informations d'identification `kubectl` utilise avec la commande suivante. *`~/.kube/config`*Remplacez-le par le chemin de votre `kubeconfig` fichier si vous n'utilisez pas le chemin par défaut.

   ```
   cat ~/.kube/config
   ```

   L'exemple qui suit illustre un résultat.

   ```
   [...]
   contexts:
   - context:
       cluster: my-cluster.region-code.eksctl.io
       user: admin@my-cluster.region-code.eksctl.io
     name: admin@my-cluster.region-code.eksctl.io
   current-context: admin@my-cluster.region-code.eksctl.io
   [...]
   ```

   Dans l'exemple de sortie précédent, les informations d'identification d'un utilisateur nommé *`admin`* sont configurées pour un cluster nommé *`my-cluster`*. Si c'est l'utilisateur qui a créé le cluster, alors il a déjà accès à votre cluster. S'il ne s'agit pas de l'utilisateur qui a créé le cluster, suivez les étapes restantes pour permettre à d'autres principaux IAM d'accéder au cluster. Les bonnes pratiques IAM recommandent d'accorder des autorisations à des rôles plutôt qu'à des utilisateurs. Vous pouvez voir quels autres principaux ont actuellement accès à votre cluster à l'aide de la commande suivante :

   ```
   kubectl describe -n kube-system configmap/aws-auth
   ```

   L'exemple qui suit illustre un résultat.

   ```
   Name:         aws-auth
   ```

```
Namespace:    kube-system
Labels:       <none>
Annotations:  <none>

Data
====
mapRoles:
----
- groups:
  - system:bootstrappers
  - system:nodes
  rolearn: arn:aws: iam::111122223333:role/my-node-role
  username: system:node:{{EC2PrivateDNSName}}


BinaryData
====

Events:  <none>
```

L'exemple précédent est un `aws-auth` `ConfigMap` par défaut. Seul le rôle d'instance du nœud a accès au cluster.

2. Assurez-vous de disposer de `roles` et `rolebindings` ou de `clusterroles` et `clusterrolebindings` existants auxquels vous pouvez associer des principaux IAM. Pour plus d'informations sur ces ressources, consultez Utilisation de l'autorisation RBAC dans la documentation Kubernetes.

   a. Affichez vos `roles` ou `clusterroles` Kubernetes existants. Les `Roles` sont associés à un `namespace`, mais les `clusterroles` sont associés au cluster.

   ```
   kubectl get roles -A
   ```

   ```
   kubectl get clusterroles
   ```

   b. Affichez les détails de tout `role` ou `clusterrole` renvoyé dans la sortie précédente et assurez-vous qu'il dispose des autorisations (`rules`) dont vous souhaitez que vos principaux IAM disposent dans votre cluster.

   Remplacez *`role-name`* par un nom de `role` renvoyé dans la sortie de la commande précédente. Remplacez *`kube-system`* par l'espace de noms du `role`.

```
kubectl describe role role-name -n kube-system
```

Remplacez *cluster-role-name* par un nom de `clusterrole` renvoyé dans la sortie de la commande précédente.

```
kubectl describe clusterrole cluster-role-name
```

c. Affichez vos `rolebindings` ou `clusterrolebindings` Kubernetes existants. Les `Rolebindings` sont associés à un `namespace`, mais les `clusterrolebindings` sont associés au cluster.

```
kubectl get rolebindings -A
```

```
kubectl get clusterrolebindings
```

d. Affichez les détails de n'importe quel `rolebinding` ou `clusterrolebinding` et confirmez qu'il possède un `role` ou un `clusterrole` de l'étape précédente répertorié comme un `roleRef` et un nom de groupe répertorié pour `subjects`.

Remplacez *role-binding-name* par un nom de `rolebinding` renvoyé dans la sortie de la commande précédente. Remplacez *kube-system* avec le `namespace` du `rolebinding`.

```
kubectl describe rolebinding role-binding-name -n kube-system
```

L'exemple qui suit illustre un résultat.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: eks-console-dashboard-restricted-access-role-binding
  namespace: default
subjects:
- kind: Group
  name: eks-console-dashboard-restricted-access-group
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role
  name: eks-console-dashboard-restricted-access-role
```

```
  apiGroup: rbac.authorization.k8s.io
```

Remplacez *cluster-role-binding-name* par un nom de `clusterrolebinding` renvoyé dans la sortie de la commande précédente.

```
kubectl describe clusterrolebinding cluster-role-binding-name
```

L'exemple qui suit illustre un résultat.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: eks-console-dashboard-full-access-binding
subjects:
- kind: Group
  name: eks-console-dashboard-full-access-group
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: eks-console-dashboard-full-access-clusterrole
  apiGroup: rbac.authorization.k8s.io
```

3. Modifiez le `aws-auth` `ConfigMap`. Vous pouvez utiliser un outil tel que `eksctl` pour mettre à jour le `ConfigMap` ou vous pouvez le mettre à jour manuellement en le modifiant.

> **Important**
>
> Nous vous recommandons d'utiliser `eksctl`, ou un autre outil, pour modifier le `ConfigMap`. Pour plus d'informations sur les autres outils que vous pouvez utiliser, consultez la section Utiliser des outils pour apporter des modifications à l'aws-authConfigMap dans les guides de bonnes pratiques Amazon EKS. Un `aws-auth` `ConfigMap` mis en forme incorrectement peut entraîner la perte de l'accès à votre cluster.

- Consultez les étapes pour modifier configmap avec eksctl.
- Consultez les étapes pour modifier configmap manuellement.

Modification du ConfigMap avec eksctl

1. Vous avez besoin d'`0.215.0`une version ou d'une version ultérieure de l'outil de ligne de `eksctl` commande installée sur votre appareil ou AWS CloudShell. Pour installer ou mettre à jour `eksctl`, veuillez consulter Installation dans la documentation de `eksctl`.
2. Affichez les mappages actuels dans le `ConfigMap`. Remplacez *`my-cluster`* par le nom de votre cluster. Remplacez *`region-code`* par la AWS région dans laquelle se trouve votre cluster.

   ```
   eksctl get iamidentitymapping --cluster my-cluster --region=region-code
   ```

   L'exemple qui suit illustre un résultat.

   ```
   ARN
                USERNAME                                GROUPS
    ACCOUNT
   arn:aws: iam::111122223333:role/eksctl-my-cluster-my-nodegroup-
   NodeInstanceRole-1XLS7754U3ZPA    system:node:{{EC2PrivateDNSName}}
    system:bootstrappers,system:nodes
   ```

3. Ajoutez un mappage pour un rôle. Remplacez *`my-role`* par le nom de votre rôle. Remplacez *`eks-console-dashboard-full-access-group`* par le nom du groupe spécifié dans votre Kubernetes `RoleBinding` ou votre objet. `ClusterRoleBinding` Remplacez *`111122223333`* par votre ID de compte. Vous pouvez remplacer *`admin`* par n'importe quel nom que vous choisissez.

   ```
   eksctl create iamidentitymapping --cluster my-cluster --region=region-code \
       --arn arn:aws: iam::111122223333:role/my-role --username admin --group eks-
   console-dashboard-full-access-group \
       --no-duplicate-arns
   ```

   > **Important**
   >
   > L'ARN de rôle ne peut pas inclure de chemin d'accès tel que `role/my-team/developers/my-role`. Le format de l'ARN doit être `arn:aws: iam::`*`111122223333`*`:role/`*`my-role`* . Dans cet exemple, `my-team/developers/` doit être supprimé.

L'exemple qui suit illustre un résultat.

```
[...]
2022-05-09 14:51:20 [#]  adding identity "{arn-aws}iam::111122223333:role/my-role" to
 auth ConfigMap
```

4. Ajoutez un mappage pour un utilisateur. Les bonnes pratiques IAM recommandent d'accorder des autorisations à des rôles plutôt qu'à des utilisateurs. Remplacez *my-user* par votre nom d'utilisateur. Remplacez *eks-console-dashboard-restricted-access-group* par le nom du groupe spécifié dans votre Kubernetes `RoleBinding` ou votre objet. `ClusterRoleBinding` Remplacez *111122223333* par votre ID de compte. Vous pouvez remplacer *my-user* par n'importe quel nom que vous choisissez.

```
eksctl create iamidentitymapping --cluster my-cluster --region=region-code \
    --arn arn:aws: iam::111122223333:user/my-user --username my-user --group eks-
console-dashboard-restricted-access-group \
    --no-duplicate-arns
```

L'exemple qui suit illustre un résultat.

```
[...]
2022-05-09 14:53:48 [#]  adding identity "arn:aws: iam::111122223333:user/my-user" to
 auth ConfigMap
```

5. Affichez de nouveau les mappages dans le `ConfigMap`.

```
eksctl get iamidentitymapping --cluster my-cluster --region=region-code
```

L'exemple qui suit illustre un résultat.

```
ARN
           USERNAME                                GROUPS
      ACCOUNT
arn:aws: iam::111122223333:role/eksctl-my-cluster-my-nodegroup-
NodeInstanceRole-1XLS7754U3ZPA    system:node:{{EC2PrivateDNSName}}
 system:bootstrappers,system:nodes
arn:aws: iam::111122223333:role/admin
              my-role                                 eks-console-dashboard-full-
access-group
```

```
arn:aws: iam::111122223333:user/my-user
            my-user                                    eks-console-dashboard-restricted-
access-group
```

Modification manuelle du ConfigMap

1. Ouvrez le `ConfigMap` pour le modifier.

```
kubectl edit -n kube-system configmap/aws-auth
```

> **Note**
>
> Si vous recevez un message d'erreur indiquant `Error from server (NotFound): configmaps "aws-auth" not found` « », utilisez la procédure décrite dans Appliquer l'aws-auth ConfigMap à votre cluster pour appliquer le stock. `ConfigMap`

2. Ajoutez vos principaux IAM à la `ConfigMap`. Un groupe IAM n'est pas un principal IAM et ne peut donc pas être ajouté au `ConfigMap`.
   - Pour ajouter un rôle IAM (par exemple, pour les utilisateurs fédérés) : ajoutez les détails du rôle à la section `mapRoles` du `ConfigMap`, sous `data`. Ajoutez cette section si elle n'existe pas déjà dans le fichier. Chaque entrée prend en charge les paramètres suivants :
     - rolearn : ARN du rôle IAM à ajouter. Cette valeur ne peut pas contenir de chemin. Par exemple, vous ne pouvez pas spécifier un ARN tel que `arn:aws: iam::111122223333:role/my-team/developers/role-name` . L'ARN doit être `arn:aws: iam::111122223333:role/role-name` .
     - username : nom d'utilisateur dans Kubernetes à mapper au rôle IAM.
     - groups : groupe ou liste des groupes Kubernetes auxquels mapper le rôle. Le groupe peut être un groupe par défaut ou un groupe spécifié dans un `clusterrolebinding` ou `rolebinding`. Pour plus d'informations, consultez Rôles par défaut et liaisons de rôles dans la documentation Kubernetes.
   - Pour ajouter un utilisateur IAM : les bonnes pratiques IAM recommandent d'accorder des autorisations à des rôles plutôt qu'à des utilisateurs. Ajoutez les détails de l'utilisateur dans la section `mapUsers` de la `ConfigMap`, sous `data`. Ajoutez cette section si elle n'existe pas déjà dans le fichier. Chaque entrée prend en charge les paramètres suivants :
     - userarn : ARN de l'utilisateur IAM à ajouter.

- username : nom d'utilisateur dans Kubernetes à mapper à l'utilisateur IAM.
- groups : groupe ou liste des groupes Kubernetes auxquels mapper l'utilisateur. Le groupe peut être un groupe par défaut ou un groupe spécifié dans un `clusterrolebinding` ou `rolebinding`. Pour plus d'informations, consultez Rôles par défaut et liaisons de rôles dans la documentation Kubernetes.

3. Par exemple, le bloc YAML ci-dessous contient :

- Une section `mapRoles` qui mappe l'instance du nœud IAM avec des groupes Kubernetes afin que les nœuds puissent s'inscrire eux-mêmes avec le cluster et le rôle IAM `my-console-viewer-role` mappé à un groupe Kubernetes pouvant afficher toutes les ressources Kubernetes pour tous les clusters. Pour obtenir la liste des autorisations de groupe IAM et Kubernetes requises pour le rôle IAM `my-console-viewer-role`, voir the section called "Autorisations requises".
- `mapUsers`Section qui fait correspondre l'utilisateur `admin` IAM du AWS compte par défaut au groupe `system:masters` Kubernetes et l'`my-user`utilisateur d'un autre AWS compte mappé à un groupe Kubernetes qui peut afficher les ressources Kubernetes pour un espace de noms spécifique. Pour obtenir la liste des autorisations de groupe IAM et Kubernetes requises pour l'utilisateur IAM `my-user`, voir the section called "Autorisations requises".

  Ajoutez ou supprimez des lignes si nécessaire et remplacez toutes les valeurs d'exemple par vos propres valeurs.

```
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this file
 will be
# reopened with the relevant failures.
#
apiVersion: v1
data:
  mapRoles: |
    - groups:
      - system:bootstrappers
      - system:nodes
      rolearn: arn:aws: iam::111122223333:role/my-role
      username: system:node:{{EC2PrivateDNSName}}
    - groups:
      - eks-console-dashboard-full-access-group
      rolearn: arn:aws: iam::111122223333:role/my-console-viewer-role
      username: my-console-viewer-role
  mapUsers: |
```

```
    - groups:
      - system:masters
      userarn: arn:aws: iam::111122223333:user/admin
      username: admin
    - groups:
      - eks-console-dashboard-restricted-access-group
      userarn: arn:aws: iam::444455556666:user/my-user
      username: my-user
```

4. Enregistrez le fichier et quittez votre éditeur de texte.

Appliquer la **ConfigMap aws-auth** à votre cluster

Le `aws-auth` `ConfigMap` est automatiquement créé et appliqué à votre cluster lorsque vous créez un groupe de nœuds géré ou lorsque vous créez un groupe de nœuds à l'aide de `eksctl`. Il est initialement créé pour permettre aux nœuds de rejoindre votre cluster, mais vous utilisez également cette `ConfigMap` pour ajouter un accès RBAC aux principaux IAM. Si vous avez lancé des nœuds autogérés et que vous n'avez pas appliqué le `ConfigMap` `aws-auth` à votre cluster, vous pouvez le faire à l'aide de la procédure suivante.

1. Vérifiez si vous avez déjà appliqué le `ConfigMap` `aws-auth`.

   ```
   kubectl describe configmap -n kube-system aws-auth
   ```

   Si vous recevez une erreur indiquant « `Error from server (NotFound): configmaps "aws-auth" not found` », appliquez le `ConfigMap` par défaut en suivant les étapes ci-dessous.
2. Téléchargez, modifiez et appliquez le plan de configuration de l' AWS authentificateur.
   a. Téléchargez la mappe de configuration.

      ```
      curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/
      aws-auth-cm.yaml
      ```

   b. Dans le fichier `aws-auth-cm.yaml`, remplacez `rolearn` par l'Amazon Resource Name (ARN) du rôle IAM associé par vos nœuds. Pour ce faire, utilisez un éditeur de texte ou remplacez *`my-node-instance-role`* et exécutez la commande suivante :

      ```
      sed -i.bak -e 's|<ARN of instance role (not instance profile)>|my-node-instance-
      role|' aws-auth-cm.yaml
      ```

Ne modifiez aucune autre ligne de ce fichier.

> **Important**
>
> L'ARN de rôle ne peut pas inclure de chemin d'accès tel que `role/my-team/developers/my-role`. Le format de l'ARN doit être `arn:aws:iam::111122223333:role/my-role` . Dans cet exemple, `my-team/developers/` doit être supprimé.

Vous pouvez inspecter les sorties de la AWS CloudFormation pile pour vos groupes de nœuds et rechercher les valeurs suivantes :

- InstanceRoleARN — Pour les groupes de nœuds créés avec `eksctl`
- NodeInstanceRole— Pour les groupes de nœuds créés à l'aide de AWS CloudFormation modèles Amazon EKS vendus dans le AWS Management Console

c. Appliquez la configuration. L'exécution de cette commande peut prendre quelques minutes.

```
kubectl apply -f aws-auth-cm.yaml
```

> **Note**
>
> Si vous recevez d'autres erreurs concernant les types d'autorisations ou de ressources, consultez the section called "Accès non autorisé ou refusé (kubectl)" dans la rubrique relative à la résolution des problèmes.

3. Observez le statut de vos nœuds et attendez qu'ils obtiennent le statut `Ready`.

```
kubectl get nodes --watch
```

Saisissez `Ctrl+C` pour revenir à une invite de shell.

Accès des utilisateurs à Kubernetes via un fournisseur OIDC externe

Amazon EKS prend en charge l'utilisation des fournisseurs d'identité OpenID Connect (OIDC) comme méthode d'authentification des utilisateurs de votre cluster. Les fournisseurs d'identité

OIDC peuvent être utilisés avec ou comme alternative à AWS Identity and Access Management (IAM). Pour plus d'informations sur l'utilisation d'IAM, consultez the section called “Accès à l'API Kubernetes”. Après avoir configuré l'authentification dans votre cluster, vous pouvez créer des `roles` et des `clusterroles` Kubernetes pour attribuer des autorisations aux rôles, puis lier les rôles aux identités à l'aide de liaisons `rolebindings` et `clusterrolebindings` Kubernetes. Pour plus d'informations, consultez Utilisation de l'autorisation RBAC dans la documentation Kubernetes.

- Vous pouvez associer un fournisseur d'identité OIDC à votre cluster.
- Kubernetes ne fournit pas de fournisseur d'identité OIDC. Vous pouvez utiliser un fournisseur d'identité OIDC public existant ou vous exécuter votre propre fournisseur d'identité. Pour obtenir la liste des fournisseurs certifiés, consultez Certification OpenID sur le site OpenID.
- L'URL de l'émetteur du fournisseur d'identité OIDC doit être accessible publiquement, afin qu'Amazon EKS puisse découvrir les clés de signature. Amazon EKS ne prend pas en charge les fournisseurs d'identité OIDC utilisant des certificats autosignés.
- Vous ne pouvez pas désactiver l'authentification IAM de votre cluster, car elle reste nécessaire pour joindre des nœuds au cluster.
- Un cluster Amazon EKS doit cependant être créé par un principal IAM AWS plutôt que par un utilisateur de fournisseur d'identité OIDC. Cela est dû au fait que le créateur du cluster interagit avec Amazon EKS APIs plutôt qu'avec APIs Kubernetes.
- Les utilisateurs authentifiés par le fournisseur d'identité OIDC sont répertoriés dans le journal d'audit du cluster si les CloudWatch journaux sont activés pour le plan de contrôle. Pour de plus amples informations, veuillez consulter the section called “Activer ou désactiver les journaux du plan de contrôle”.
- Vous ne pouvez pas vous connecter au AWS Management Console avec un compte auprès d'un fournisseur OIDC. Vous ne pouvez le faire qu'en vous the section called “Accéder aux ressources du cluster” connectant AWS Management Console à l'aide d'un compte AWS Identity and Access Management.

Associer un fournisseur d'identité OIDC

Pour pouvoir associer un fournisseur d'identité OIDC à votre cluster, vous devez obtenir les informations suivantes de votre fournisseur :

URL de l'émetteur

L'URL du fournisseur d'identité OIDC qui permet au serveur d'API de découvrir les clés de signature publiques pour vérifier les jetons. L'URL doit commencer par `https://` et correspondre à la revendication `iss` figurant dans les jetons d'ID OIDC du fournisseur. Conformément à la norme OIDC, les composants du chemin sont autorisés, mais pas les paramètres de requête. Généralement, l'URL se compose uniquement d'un nom d'hôte, comme `https://server.example.org` ou `https://example.com`. Cette URL doit pointer vers le niveau sous `.well-known/openid-configuration` et doit être accessible publiquement sur Internet.

ID client (également appelé audience)

L'ID de l'application cliente qui envoie les demandes d'authentification au fournisseur d'identité OIDC.

Vous pouvez associer un fournisseur d'identité à l'aide de `eksctl` ou de la AWS Management Console.

Association d'un fournisseur d'identité à l'aide d'eksctl

1. Créez un fichier nommé `associate-identity-provider.yaml` avec les contenus suivants. Remplacez les exemples de valeurs par les vôtres. Les valeurs de la section `identityProviders` sont obtenues de votre fournisseur d'identité OIDC. Des valeurs ne sont requises que pour les paramètres `name`, `type`, `issuerUrl` et `clientId` sous `identityProviders`.

```
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-cluster
  region: your-region-code

identityProviders:
  - name: my-provider
    type: oidc
    issuerUrl: https://example.com
    clientId: kubernetes
    usernameClaim: email
```

```
    usernamePrefix: my-username-prefix
    groupsClaim: my-claim
    groupsPrefix: my-groups-prefix
    requiredClaims:
      string: string
    tags:
      env: dev
```

> **Important**
>
> Ne spécifiez pas `system:`, ni aucune portion de cette chaîne, pour `groupsPrefix` ou `usernamePrefix`.

2. Créez le fournisseur.

```
eksctl associate identityprovider -f associate-identity-provider.yaml
```

3. Pour utiliser `kubectl` avec votre cluster et un fournisseur d'identité OIDC, consultez Utilisation de kubectl dans la documentation Kubernetes.

Associer un fournisseur d'identité à l'aide de la AWS console

1. Ouvrez la console Amazon EKS.
2. Sélectionnez votre cluster, puis sélectionnez l'onglet Accès.
3. Dans la section Fournisseurs d'identité OIDC, sélectionnez Associer un fournisseur d'identité.
4. Sur la page Associate OIDC Identity Provider (Associer un fournisseur d'identité OIDC), saisissez ou sélectionnez les options suivantes, puis sélectionnez Associate (Associer).
   - Dans Nom, saisissez un nom unique pour le fournisseur.
   - Pour URL de l'émetteur, saisissez l'URL de votre fournisseur. Cette URL doit être accessible via Internet.
   - Pour ID client, saisissez l'ID client du fournisseur d'identité OIDC (également appelé audience).
   - Pour Nom d'utilisateur, saisissez la revendication à utiliser comme nom d'utilisateur.
   - Pour Revendication de groupes, saisissez la revendication à utiliser comme groupe de l'utilisateur.
   - (Facultatif) Sélectionnez Options avancées, puis saisissez ou sélectionnez les informations suivantes :

- Nom d'utilisateur : saisissez un préfixe à ajouter aux revendications de nom d'utilisateur. Le préfixe est ajouté aux revendications de nom d'utilisateur pour éviter les conflits avec les noms existants. Si vous ne fournissez aucune valeur et que le nom d'utilisateur est une valeur autre que `email`, le préfixe correspond par défaut à la valeur URL de l'émetteur. Vous pouvez utiliser la valeur `-` pour désactiver toutes les préfixes. Ne spécifiez pas `system:` ni aucune partie de cette chaîne.
- Préfixe de groupes : saisissez un préfixe à ajouter aux revendications de groupes. Le préfixe est ajouté aux revendications de groupe pour éviter les conflits avec des noms existants (tels que `system: groups`). Par exemple, la valeur `oidc:` crée des noms de groupe comme `oidc:engineering` et `oidc:infra`. Ne spécifiez pas `system:` ni aucune partie de cette chaîne.
- Revendications requises : sélectionnez Ajouter une revendication et saisissez une ou plusieurs paires de valeurs clés qui décrivent les revendications requises dans le jeton d'ID de client. Les paires décrivent les revendications requises dans le jeton d'ID. Si elle est définie, la présence de chaque revendication est vérifiée dans le jeton d'ID avec une valeur correspondante.
  a. Pour utiliser `kubectl` avec votre cluster et un fournisseur d'identité OIDC, consultez Utilisation de kubectl dans la documentation Kubernetes.

Exemple de politique IAM

Si vous voulez empêcher un fournisseur d'identité OIDC d'être associé à un cluster, créez et associez la politique IAM suivante aux comptes IAM de vos administrateurs Amazon EKS. Pour plus d'informations, consultez Création de politiques IAM et Ajout d'autorisations d'identité IAM dans le Guide de l'utilisateur IAM, ainsi que Actions dans la Référence d'autorisation de service.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "denyOIDC",
            "Effect": "Deny",
            "Action": [
                "eks:AssociateIdentityProviderConfig"
            ],
            "Resource": "arn:aws:eks:us-west-2:111122223333:cluster/*"

        },
```

```
        {
            "Sid": "eksAdmin",
            "Effect": "Allow",
            "Action": [
                "eks:*"
            ],
            "Resource": "*"
        }
    ]
}
```

L'exemple de politique suivant autorise l'association de fournisseur d'identité OIDC si `clientID` est `kubernetes` et `issuerUrl` est `https://cognito-idp.us-west-2amazonaws.com/*`.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowCognitoOnly",
            "Effect": "Deny",
            "Action": "eks:AssociateIdentityProviderConfig",
            "Resource": "arn:aws:eks:us-west-2:111122223333:cluster/my-instance",
            "Condition": {
                "StringNotLikeIfExists": {
                    "eks:issuerUrl": "https://cognito-idp.us-west-2.amazonaws.com/*"
                }
            }
        },
        {
            "Sid": "DenyOtherClients",
            "Effect": "Deny",
            "Action": "eks:AssociateIdentityProviderConfig",
            "Resource": "arn:aws:eks:us-west-2:111122223333:cluster/my-instance",
            "Condition": {
                "StringNotEquals": {
                    "eks:clientId": "kubernetes"
                }
            }
        },
        {
            "Sid": "AllowOthers",
            "Effect": "Allow",
            "Action": "eks:*",
```

```
            "Resource": "*"
        }
    ]
}
```

Dissocier un fournisseur d'identité OIDC de votre cluster

Si vous dissociez un fournisseur d'identité OIDC de votre cluster, les utilisateurs inclus dans le fournisseur ne peuvent plus accéder au cluster. Cependant, vous pouvez toujours accéder au cluster avec les principaux IAM.

1. Ouvrez la console Amazon EKS.
2. Dans la section Fournisseurs d'identité OIDC, sélectionnez Dissocier, saisissez le nom du fournisseur d'identité, puis sélectionnez `Disassociate`.

Consultez les ressources Kubernetes dans le AWS Management Console

Vous pouvez afficher les ressources Kubernetes déployées sur votre cluster avec la AWS Management Console. Vous ne pouvez pas afficher les ressources Kubernetes avec la AWS CLI ou eksctl. Pour afficher les ressources Kubernetes à l'aide d'un outil en ligne de commande, utilisez kubectl.

> **Note**
>
> Pour afficher l'onglet Ressources et la section Nœuds de l'onglet Compute du AWS Management Console, le principal IAM que vous utilisez doit disposer d'autorisations IAM et Kubernetes spécifiques. Pour de plus amples informations, veuillez consulter the section called "Autorisations requises".

1. Ouvrez la console Amazon EKS.
2. Dans la liste Clusters, sélectionnez le cluster qui contient les ressources Kubernetes que vous souhaitez afficher.
3. Sélectionnez l'onglet Resources (Ressources).

4. Sélectionnez un groupe Resource type (Type de ressource) pour lequel vous souhaitez afficher les ressources, comme Workloads (Charges de travail). Vous voyez une liste des types de ressources dans ce groupe.
5. Sélectionnez un type de ressource, tel que Deployments (Déploiements), dans le groupe Workloads (Charges de travail). Vous voyez une description du type de ressource, un lien vers la documentation Kubernetes pour plus d'informations sur le type de ressource et une liste des ressources de ce type qui sont déployées sur votre cluster. Si la liste est vide, aucune ressource de ce type n'est déployée sur votre cluster.
6. Sélectionnez une ressource pour afficher plus d'informations à son sujet. Essayez les exemples suivants :
   - Sélectionnez le groupe Workloads (Charges de travail), sélectionnez le type de ressource Deployments (Déploiements), puis sélectionnez la ressource coredns. Lorsque vous sélectionnez une ressource, vous êtes dans Structured view (Vue structurée), par défaut. Pour certains types de ressources, vous voyez une section Pod dans Structured view (Vue structurée). Cette section répertorie les pods gérés par la charge de travail. Vous pouvez sélectionner n'importe quel pod répertorié pour afficher des informations à son sujet. Tous les types de ressources n'affichent pas d'informations dans Structured View (Vue structurée). Si vous sélectionnez Raw view (Vue brute) dans le coin supérieur droit de la page de la ressource, vous voyez la réponse JSON complète de l'API Kubernetes pour la ressource.
   - Sélectionnez le groupe Cluster, puis sélectionnez le type de ressource Nodes (Nœuds). Vous voyez une liste de tous les nœuds de votre cluster. Les nœuds peuvent être n'importe quel type de nœud Amazon EKS. Il s'agit de la même liste que celle que vous voyez dans la section Nodes (Nœuds) lorsque vous sélectionnez l'onglet Compute (Calcul) pour votre cluster. Sélectionnez une ressource de nœud dans la liste. Dans Structured view (Vue structurée), vous voyez également une section Pod. Cette section vous montre tous les pods s'exécutant sur le nœud.

Autorisations requises

Pour afficher l'onglet Ressources et la section Nœuds de l'onglet Compute du AWS Management Console, le principal IAM que vous utilisez doit disposer d'autorisations IAM et Kubernetes minimales spécifiques. Les autorisations IAM et Kubernetes RBAC doivent être correctement configurées. Procédez comme suit pour accorder les autorisations requises à vos principaux IAM.

1. Veuillez vous assurer que les autorisations `eks:AccessKubernetesApi` et autres autorisations IAM nécessaires pour afficher les ressources Kubernetes sont attribuées au principal IAM que

vous utilisez. Pour plus d'informations sur la modification des autorisations d'un principal IAM, consultez la rubrique Contrôle de l'accès des principaux du Guide de l'utilisateur IAM. Pour plus d'informations sur la modification des autorisations pour un rôle, consultez Modification d'une politique d'autorisations de rôle (console) dans le guide de l'utilisateur IAM.

L'exemple de politique suivant inclut les autorisations nécessaires pour qu'un principal puisse afficher les ressources Kubernetes pour tous les clusters de votre compte. Remplacez *111122223333* par votre identifiant de AWS compte.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "eks:ListFargateProfiles",
                "eks:DescribeNodegroup",
                "eks:ListNodegroups",
                "eks:ListUpdates",
                "eks:AccessKubernetesApi",
                "eks:ListAddons",
                "eks:DescribeCluster",
                "eks:DescribeAddonVersions",
                "eks:ListClusters",
                "eks:ListIdentityProviderConfigs",
                "iam:ListRoles"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "ssm:GetParameter",
            "Resource": "arn:aws:ssm:*:111122223333:parameter/*"
        }
    ]
}
```

Pour afficher les nœuds des clusters connectés, le rôle IAM du connecteur Amazon EKS doit être en mesure d'emprunter l'identité du principal dans le cluster. Cela permet à Amazon EKS Connector de mapper le principal à un utilisateur Kubernetes.

2. Configurez les autorisations RBAC Kubernetes à l'aide des entrées d'accès EKS.

Que sont les entrées EKS Access ?

Les entrées d'accès EKS constituent un moyen simplifié d'accorder aux principaux IAM (utilisateurs et rôles) l'accès à votre cluster Kubernetes. Au lieu de gérer manuellement les ressources Kubernetes RBAC et les entrées d'accès gèrent automatiquement le `aws-auth` ConfigMap mappage entre les autorisations IAM et Kubernetes à l'aide des politiques gérées fournies par. AWS Pour des informations détaillées sur les entrées d'accès, voirthe section called “Entrées d’accès”. Pour plus d'informations sur les politiques d'accès disponibles et leurs autorisations, consultez la section Autorisations relatives aux politiques d'accès.

Vous pouvez attacher des autorisations Kubernetes aux entrées d’accès de deux façons :

- Utiliser une politique d'accès : les politiques d'accès sont des modèles d'autorisations Kubernetes prédéfinis gérés par. AWS Ils fournissent des ensembles d'autorisations standardisés pour les cas d'utilisation courants.
- Référencer un groupe Kubernetes : si vous associez une identité IAM à un groupe Kubernetes, vous pouvez créer des ressources Kubernetes qui accordent des autorisations au groupe. Pour plus d'informations, consultez Utilisation de l'autorisation RBAC dans la documentation Kubernetes.

  a. Créez une entrée d'accès pour votre principal IAM à l'aide de la AWS CLI. Remplacez *my-cluster* par le nom de votre cluster. Remplacez *111122223333* par votre ID de compte.

  ```
  aws eks create-access-entry \
      --cluster-name my-cluster \
      --principal-arn arn:aws: iam::111122223333:role/my-console-viewer-role
  ```

  L'exemple qui suit illustre un résultat.

  ```
  {
      "accessEntry": {
          "clusterName": "my-cluster",
          "principalArn": "arn:aws: iam::111122223333:role/my-console-viewer-
  role",
          "kubernetesGroups": [],
          "accessEntryArn": "arn:aws: eks:region-code:111122223333:access-
  entry/my-cluster/role/111122223333/my-console-viewer-role/
  abc12345-1234-1234-1234-123456789012",
          "createdAt": "2024-03-15T10:30:45.123000-07:00",
          "modifiedAt": "2024-03-15T10:30:45.123000-07:00",
  ```

```
        "tags": {},
        "username": "arn:aws: iam::111122223333:role/my-console-viewer-role",
        "type": "STANDARD"
    }
}
```

b. Associez une politique à l'entrée d'accès. Pour consulter les ressources Kubernetes, utilisez : `AmazonEKSViewPolicy`

```
aws eks associate-access-policy \
    --cluster-name my-cluster \
    --principal-arn arn:aws: iam::111122223333:role/my-console-viewer-role \
    --policy-arn arn:aws: eks::aws:cluster-access-policy/AmazonEKSViewPolicy \
    --access-scope type=cluster
```

L'exemple qui suit illustre un résultat.

```
{
    "clusterName": "my-cluster",
    "principalArn": "arn:aws: iam::111122223333:role/my-console-viewer-role",
    "associatedAt": "2024-03-15T10:31:15.456000-07:00"
}
```

Pour un accès spécifique à un espace de noms, vous pouvez étendre la politique à des espaces de noms spécifiques :

```
aws eks associate-access-policy \
    --cluster-name my-cluster \
    --principal-arn arn:aws: iam::111122223333:role/my-console-viewer-role \
    --policy-arn arn:aws: eks::aws:cluster-access-policy/AmazonEKSViewPolicy \
    --access-scope type=namespace,namespaces=default,kube-system
```

c. Vérifiez que l'entrée d'accès a bien été créée :

```
aws eks describe-access-entry \
    --cluster-name my-cluster \
    --principal-arn arn:aws: iam::111122223333:role/my-console-viewer-role
```

d. Répertoriez les politiques associées pour confirmer l'association des politiques :

```
aws eks list-associated-access-policies \
```

```
    --cluster-name my-cluster \
    --principal-arn arn:aws: iam::111122223333:role/my-console-viewer-role
```

L'exemple qui suit illustre un résultat.

```
{
    "associatedAccessPolicies": [
        {
            "policyArn": "arn:aws: eks::aws:cluster-access-policy/
AmazonEKSViewPolicy",
            "accessScope": {
                "type": "cluster"
            },
            "associatedAt": "2024-03-15T10:31:15.456000-07:00",
            "modifiedAt": "2024-03-15T10:31:15.456000-07:00"
        }
    ]
}
```

CloudTrail visibilité

Lorsque vous consultez les ressources Kubernetes, le nom de l'opération suivant apparaît dans vos journaux : CloudTrail

- `AccessKubernetesApi`- Lorsque vous lisez ou consultez des ressources

Cet CloudTrail événement fournit une piste d'audit de l'accès en lecture à vos ressources Kubernetes.

> **Note**
>
> Ce nom d'opération apparaît dans les CloudTrail journaux à des fins d'audit uniquement. Il ne s'agit pas d'une action IAM et ne peut pas être utilisée dans les déclarations de politique IAM. Pour contrôler l'accès en lecture aux ressources Kubernetes par le biais de politiques IAM, utilisez l'`eks:AccessKubernetesApi`autorisation indiquée dans la section. the section called “Autorisations requises”

Accorder AWS aux services l'accès en écriture à Kubernetes APIs

Autorisations requises

Pour permettre aux AWS services d'effectuer des opérations d'écriture sur les ressources Kubernetes de votre cluster Amazon EKS, vous devez accorder à la fois les autorisations IAM `eks:AccessKubernetesApi` et `eks:MutateViaKubernetesApi` IAM.

Par exemple, Amazon SageMaker HyperPod utilise ces autorisations pour permettre le déploiement de modèles depuis SageMaker AI Studio. Pour plus d'informations, consultez la section Configurer les autorisations facultatives du JavaScript SDK dans le manuel Amazon SageMaker AI Developer Guide.

> **Important**
>
> Les opérations d'écriture telles que la création, la mise à jour et la suppression nécessitent les deux autorisations. Si l'une d'entre elles est absente, les opérations d'écriture échoueront.

CloudTrail visibilité

Lorsque vous effectuez des opérations d'écriture sur les ressources Kubernetes, vous verrez des noms d'opérations spécifiques dans vos journaux : CloudTrail

- `createKubernetesObject`- Lors de la création de nouvelles ressources
- `updateKubernetesObject`- Lors de la modification de ressources existantes
- `deleteKubernetesObject`- Lors de la suppression de ressources

Ces CloudTrail événements fournissent des pistes d'audit détaillées de toutes les modifications apportées à vos ressources Kubernetes.

> **Note**
>
> Ces noms d'opérations apparaissent dans les CloudTrail journaux à des fins d'audit uniquement. Il ne s'agit pas d'actions IAM et ne peuvent pas être utilisées dans les déclarations de politique IAM. Pour contrôler l'accès en écriture aux ressources Kubernetes

par le biais de politiques IAM, utilisez l'`eks:MutateViaKubernetesApi`autorisation indiquée dans la section. the section called "Autorisations requises"

Connexion de kubectl à un cluster EKS en créant un fichier kubeconfig

Tip

Inscrivez-vous aux prochains ateliers Amazon EKS.

Dans cette rubrique, vous allez créer un fichier `kubeconfig` pour votre cluster (ou mettre à jour un fichier existant).

L'outil de ligne de commande `kubectl` utilise les informations de configuration dans les fichiers `kubeconfig` pour communiquer avec le serveur API d'un cluster. Pour plus d'informations, consultez Organisation de l'accès au cluster à l'aide des fichiers kubeconfig dans la documentation Kubernetes.

Amazon EKS utilise la commande `aws eks get-token` avec `kubectl` pour l'authentification du cluster. Par défaut, la AWS CLI utilise les mêmes informations d'identification que celles renvoyées par la commande suivante :

```
aws sts get-caller-identity
```

- Un cluster Amazon EKS existant. Pour en déployer un, consultez Démarrer.
- L'outil de ligne de commande `kubectl` est installé sur votre appareil ou AWS CloudShell. La version peut correspondre à celle utilisée par votre cluster Kubernetes, ou différer d'au plus une version mineure, qu'elle soit antérieure ou plus récente. Par exemple, si la version de votre cluster est `1.29`, vous pouvez utiliser la version `kubectl` `1.28`, `1.29` ou `1.30`. Pour installer ou mettre à niveau `kubectl`, veuillez consulter the section called "Configurer kubectl et eksctl".
- Version `2.12.3` ou version ultérieure `1.27.160` ou version ultérieure de l'interface de ligne de AWS commande (AWS CLI) installée et configurée sur votre appareil ou AWS CloudShell. Pour vérifier votre version actuelle, utilisez `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Les gestionnaires de packages tels que `yum` Homebrew pour macOS ont souvent plusieurs versions de retard sur la dernière version de la AWS CLI. `apt-get` Pour installer la dernière version, consultez la section Installation et configuration rapide avec aws configure dans le Guide

de l'utilisateur de l'interface de ligne de AWS commande. La version de la AWS CLI installée AWS CloudShell peut également avoir plusieurs versions de retard par rapport à la dernière version. Pour le mettre à jour, consultez la section Installation de la AWS CLI dans votre répertoire de base dans le guide de AWS CloudShell l'utilisateur.

- Un utilisateur IAM ou un rôle IAM avec l'autorisation d'utiliser l'action d'API `eks:DescribeCluster` pour le cluster que vous spécifiez. Pour de plus amples informations, veuillez consulter the section called "Politiques basées sur l'identité". Si vous utilisez une identité provenant de votre propre fournisseur OpenID Connect pour accéder à votre cluster, consultez Utilisation de kubectl dans la documentation Kubernetes afin de créer ou de mettre à jour votre fichier `kube config`.

Créer le fichier **`kubeconfig`** automatiquement

- Version `2.12.3` ou version ultérieure `1.27.160` ou version ultérieure de l'interface de ligne de AWS commande (AWS CLI) installée et configurée sur votre appareil ou AWS CloudShell. Pour vérifier votre version actuelle, utilisez `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Les gestionnaires de packages tels que `yum` Homebrew pour macOS ont souvent plusieurs versions de retard sur la dernière version de la AWS CLI. `apt-get` Pour installer la dernière version, consultez la section Installation et configuration rapide avec aws configure dans le Guide de l'utilisateur de l'interface de ligne de AWS commande. La version de la AWS CLI installée AWS CloudShell peut également avoir plusieurs versions de retard par rapport à la dernière version. Pour le mettre à jour, consultez la section Installation de la AWS CLI dans votre répertoire de base dans le guide de AWS CloudShell l'utilisateur.
- Autorisation d'utiliser l'action d'API `eks:DescribeCluster` pour le cluster que vous spécifiez. Pour de plus amples informations, veuillez consulter the section called "Politiques basées sur l'identité".

1. Créez ou mettez à jour un fichier `kubeconfig` pour votre cluster. Remplacez *`region-code`* par la AWS région dans laquelle se trouve votre cluster et remplacez *`my-cluster`* par le nom de votre cluster.

   ```
   aws eks update-kubeconfig --region region-code --name my-cluster
   ```

   Par défaut, le fichier de configuration résultant est créé dans le chemin `kubeconfig` par défaut (`.kube`) dans votre répertoire de base ou fusionné avec un fichier `config` existant à cet emplacement. Vous pouvez spécifier un autre chemin avec l'option `--kubeconfig`.

Vous pouvez spécifier un ARN de rôle IAM avec l'option `--role-arn` à utiliser pour l'authentification lorsque vous émettez des commandes `kubectl`. Dans le cas contraire, le principal IAM de votre chaîne d'identification par défaut de la AWS CLI ou du SDK est utilisé. Vous pouvez consulter l'identité de votre AWS CLI ou de votre SDK par défaut en exécutant la `aws sts get-caller-identity` commande.

Pour connaître toutes les options disponibles, exécutez la *`aws eks update-kubeconfig help`* commande ou consultez update-kubeconfig dans le manuel de référence des commandes de la CLI AWS .

2. Testez votre configuration.

```
kubectl get svc
```

L'exemple qui suit illustre un résultat.

```
NAME             TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
svc/kubernetes   ClusterIP   10.100.0.1   <none>        443/TCP   1m
```

Si vous recevez d'autres erreurs concernant les types d'autorisations ou de ressources, consultez the section called “Accès non autorisé ou refusé (`kubectl`)” dans la rubrique relative à la résolution des problèmes.

Accorder aux charges de travail Kubernetes l'accès à AWS à l'aide des comptes de service Kubernetes

A Kubernetes service account provides an identity for processes that run in a Pod. For more information see Gestion des comptes de service in the Kubernetes documentation. If your Pod needs access to AWS services, you can map the service account to an AWS Identity and Access Management identity to grant that access. For more information, see the section called “Informations d'identification auprès de l'IRSA” or the section called “Pod Identity”.

Jetons de compte de service

La fonctionnalité BoundServiceAccountTokenVolume est activée par défaut dans les versions Kubernetes. Cette fonctionnalité améliore la sécurité des jetons de compte de service en permettant aux charges de travail exécutées sur Kubernetes de demander des jetons Web JSON liés à

l'audience, à l'heure et à la clé. Les jetons de compte de service ont une expiration d'une heure. Dans les versions antérieures de Kubernetes, les jetons n'avaient pas de date d'expiration. Cela signifie que les clients qui s'appuient sur ces jetons doivent actualiser les jetons en moins d'une heure. Les kits SDK client Kubernetes suivants actualisent automatiquement les jetons dans le délai requis :

- Go version `0.15.7` et ultérieure
- Python version `12.0.0` et ultérieure
- Java version `9.0.0` et ultérieure
- JavaScript version `0.10.3` et ultérieure
- Branche `master` Ruby
- Haskell version `0.3.0.0`
- Version C# `7.0.5` et ultérieure

Si votre charge de travail utilise une version client antérieure, vous devez la mettre à jour. Afin de permettre une migration fluide des clients vers les nouveaux jetons de compte de service à durée limitée, Kubernetes ajoute une période d'expiration prolongée au jeton de compte de service, au-delà de l'heure par défaut. Pour les clusters Amazon EKS, la période d'expiration prolongée est de 90 jours. Le serveur API Kubernetes de votre cluster Amazon EKS rejette les demandes dont les jetons ont plus de 90 jours. Nous vous recommandons de vérifier vos applications et leurs dépendances pour vous assurer que les SDK des clients Kubernetes sont de versions identiques ou ultérieures aux versions répertoriées ci-dessus.

Lorsque le serveur API reçoit des requêtes avec des jetons de plus d'une heure, il annote l'événement du journal d'audit de l'API avec `annotations.authentication.k8s.io/stale-token`. La valeur de l'annotation ressemble à l'exemple suivant :

```
subject: system:serviceaccount:common:fluent-bit, seconds after warning threshold:
 4185802.
```

Si votre cluster a une journalisation de plan de contrôle activée, les annotations se trouvent dans les journaux d'audit. Vous pouvez utiliser la requête CloudWatch Logs Insights suivante pour identifier tous les pods de votre cluster Amazon EKS qui utilisent des jetons périmés :

```
fields @timestamp
|filter @logStream like /kube-apiserver-audit/
|filter @message like /seconds after warning threshold/
```

```
|parse @message "subject: *, seconds after warning threshold:*\"" as subject,
 elapsedtime
```

Le `subject` fait référence au compte de service utilisé par le pod. Le `elapsedtime` indique le temps écoulé (en secondes) après la lecture du dernier jeton. Les demandes adressées au serveur d'API sont refusées lorsque le `elapsedtime` dépasse 90 jours (7 776 000 secondes). Il est recommandé de mettre à jour de manière proactive le kit SDK client Kubernetes de vos applications afin d'utiliser l'une des versions répertoriées précédemment qui actualisent automatiquement le jeton. Si le jeton de compte de service utilisé approche les 90 jours et que vous ne disposez pas de suffisamment de temps pour mettre à jour les versions de votre kit SDK client avant l'expiration du jeton, vous pouvez résilier les pods existants et en créer de nouveaux. Il en résulte une récupération du jeton de compte de service, ce qui vous donne 90 jours supplémentaires pour mettre à jour les kits SDK de votre version client.

Si le pod fait partie d'un déploiement, la suggestion pour résilier les pods tout en conservant une haute disponibilité consiste à effectuer un déploiement à l'aide de la commande suivante. Remplacez *`my-deployment`* par le nom de votre déploiement.

```
kubectl rollout restart deployment/my-deployment
```

Modules complémentaires de cluster

Les modules complémentaires de cluster suivants ont été mis à jour afin d'utiliser les SDK clients Kubernetes qui récupèrent automatiquement les jetons de compte de service. Nous vous recommandons de vous assurer que les versions répertoriées, ou des versions ultérieures, sont installées sur votre cluster.

- Plug-in CNI Amazon VPC pour Kubernetes et plug-ins d'aide aux métriques version `1.8.0` et ultérieure. Pour vérifier votre version actuelle ou la mettre à jour, consultez the section called "CNI Amazon VPC" et cni-metrics-helper.
- CoreDNS version `1.8.4` et ultérieure. Pour vérifier votre version actuelle ou la mettre à jour, consultez the section called "CoreDNS".
- AWS Load Balancer Controller version `2.0.0` et ultérieure. Pour vérifier votre version actuelle ou la mettre à jour, consultez the section called " AWS Contrôleur Load Balancer".
- Une version actuelle de `kube-proxy`. Pour vérifier votre version actuelle ou la mettre à jour, consultez the section called "kube-proxy".

- AWS pour Fluent Bit version `2.25.0` ou ultérieure. Pour mettre à jour votre version actuelle, consultez Versions sur GitHub.
- Version d'image Fluentd 1.14.6-1.2 ou version ultérieure et plug-in de filtre Fluentd pour les métadonnées de Kubernetes version 2.11.1 ou ultérieure.

Octroi d'autorisations de gestion des identités et des accès AWS aux charges de travail sur les clusters Amazon Elastic Kubernetes Service

Amazon EKS propose deux méthodes pour octroyer des autorisations de gestion des identités et des accès AWS aux charges de travail qui s'exécutent dans les clusters Amazon EKS : Rôles IAM pour les comptes de service et Identités du pod EKS.

Rôles IAM pour les comptes de service

Les rôles IAM des comptes de service (IRSA) configurent les applications Kubernetes s'exécutant sur AWS avec des autorisations IAM détaillées pour accéder à diverses autres ressources AWS, telles que les compartiments Amazon S3, les tables Amazon DynamoDB, etc. Vous pouvez exécuter plusieurs applications ensemble dans le même cluster Amazon EKS, et vous assurer que chaque application ne dispose que de l'ensemble minimum d'autorisations dont elle a besoin. IRSA a été créé pour prendre en charge diverses options de déploiement Kubernetes prises en charge par AWS, telles qu'Amazon EKS, Amazon EKS Anywhere, Red Hat OpenShift Service sur AWS et les clusters Kubernetes autogérés sur les instances Amazon EC2. Ainsi, IRSA a été créé en utilisant un service AWS fondamental comme IAM, et n'a pas pris de dépendance directe sur le service Amazon EKS et l'API EKS. Pour de plus amples informations, consultez the section called "Informations d'identification auprès de l'IRSA".

Identités des pods EKS

L'identité du pod EKS offre aux administrateurs de cluster un flux simplifié pour l'authentification des applications afin d'accéder à diverses autres ressources AWS telles que les compartiments Amazon S3, les tables Amazon DynamoDB, etc. L'identité du pod EKS est réservée uniquement à EKS et, par conséquent, elle simplifie la manière dont les administrateurs de cluster peuvent configurer les applications Kubernetes pour obtenir des autorisations IAM. Ces autorisations peuvent désormais être facilement configurées en quelques étapes directement via AWS Management Console, l'API EKS et l'interface AWS CLI, et aucune action n'est requise à l'intérieur du cluster dans les objets Kubernetes. Les administrateurs de clusters n'ont pas besoin de basculer entre les services EKS et IAM, ni d'utiliser des opérations IAM privilégiées pour configurer les autorisations requises par vos applications. Les rôles IAM peuvent désormais

être utilisés dans plusieurs clusters sans qu'il soit nécessaire de mettre à jour la politique d'approbation des rôles lors de la création de nouveaux clusters. Les informations d'identification IAM fournies par l'identité du pod EKS comprennent des balises de session de rôle, avec des attributs tels que le nom du cluster, l'espace de noms, le nom du compte de service. Les balises de session de rôle permettent aux administrateurs de créer un rôle unique qui peut fonctionner sur plusieurs comptes de service en autorisant l'accès aux ressources AWS sur la base des balises correspondantes. Pour de plus amples informations, consultez the section called "Pod Identity".

Comparaison de l'identité du pod EKS et de l'IRSA

À un niveau élevé, l'identité du pod EKS et l'IRSA vous permettent tous deux d'accorder des autorisations IAM aux applications exécutées sur des clusters Kubernetes. Mais ils sont fondamentalement différents dans la façon dont vous les configurez, les limites prises en charge et les fonctionnalités activées. Ci-dessous, nous comparons certains des aspects clés des deux solutions.

> **Note**
>
> AWS recommande d'utiliser les identités du pod EKS pour accorder l'accès aux ressources AWS à vos pods dans la mesure du possible. Pour de plus amples informations, consultez the section called "Pod Identity".

Attribut	Identité du pod EKS	IRSA
Extensibilité des rôles	Vous devez configurer chaque rôle une fois pour établir la confiance avec le nouveau principal de service Amazon EKS `pods.eks.amazonaws.com` . Après cette étape unique, vous n'avez pas besoin de mettre à jour la politique d'approbation du rôle à chaque fois qu'il est utilisé dans un nouveau cluster.	Vous devez mettre à jour la politique d'approbation du rôle IAM avec le nouveau point de terminaison du fournisseur OIDC du cluster EKS chaque fois que vous voulez utiliser le rôle dans un nouveau cluster.

Attribut	Identité du pod EKS	IRSA
Capacité de mise à l'échelle des clusters	L'identité du pod EKS n'exige pas des utilisateurs qu'ils configurent le fournisseur IAM OIDC, cette limite ne s'appliqu e donc pas.	Chaque cluster EKS est associé à une URL d'émetteur OpenID Connect (OIDC). Pour utiliser IRSA, un fournisseur OpenID Connect unique doit être créé pour chaque cluster EKS dans IAM. IAM a une limite globale par défaut de 100 fournisseurs OIDC pour chaque compte AWS. Si vous prévoyez d'avoir plus de 100 clusters EKS pour chaque compte AWS avec IRSA, vous atteindrez la limite du fournisseur IAM OIDC.

Attribut	Identité du pod EKS	IRSA
Capacité de mise à l'échelle des rôles	L'identité du pod EKS n'exige pas des utilisateurs qu'ils définissent une relation d'approbation entre le rôle IAM et le compte de service dans la politique d'approba tion, cette limite ne s'applique donc pas.	Dans IRSA, vous définissez la relation d'approbation entre un rôle IAM et un compte de service dans la politique d'approbation du rôle. Par défaut, la longueur de la politique d'approbation est `2048`. Cela signifie que vous pouvez généralement définir 4 relations d'approbation dans une seule politique d'approbation. Bien qu'il soit possible d'augmenter la limite de longueur de la politique d'approbation, vous êtes généralement limité à un maximum de 8 relations d'approbation au sein d'une même politique d'approbation.

Attribut	Identité du pod EKS	IRSA
Utilisation du quota d'API STS	L'identité du pod EKS simplifie la fourniture d'informations d'identification AWS à vos pods et ne nécessite pas que votre code effectue des appels directement avec le service de jetons de sécurité AWS (STS). Le service EKS gère l'assomption de rôle et fournit des informations d'identif ication aux applications écrites à l'aide du kit SDK AWS dans vos pods sans que ceux-ci communiquent avec AWS STS ou utilisent le quota d'API STS.	Dans IRSA, les applications écrites à l'aide du kit SDK AWS dans vos pods utilisent des jetons pour appeler l'API `AssumeRoleWithWebI dentity` sur le service de jetons de sécurité AWS (STS). En fonction de la logique de votre code sur le kit SDK AWS, il est possible que votre code effectue des appels inutiles vers AWS STS et reçoive des erreurs de limitatio n.

Attribut	Identité du pod EKS	IRSA
Réutilisation des rôles	Les informations d'identif ication temporaires AWS STS fournies par l'identité du pod Amazon EKS comprennent des balises de session de rôle, telles que le nom du cluster, l'espace de noms et le nom du compte de service. Les balises de session de rôle permetten t aux administrateurs de créer un rôle IAM unique qui peut être utilisé avec plusieurs comptes de service, avec différentes autorisat ions effectives, en autorisant l'accès aux ressources AWS en fonction des balises qui leur sont attachées. C'est ce qu'on appelle le contrôle d'accès par attributs (ABAC). Pour de plus amples informati ons, consultez the section called "Accorder l'accès aux pods".	Les balises de session AWS STS ne sont pas prises en charge. Vous pouvez réutiliser un rôle entre les clusters, mais chaque pod reçoit toutes les autorisations du rôle.
Environnements pris en charge	L'identité du pod EKS est uniquement disponible sur Amazon EKS.	IRSA peut être utilisé avec Amazon EKS, Amazon EKS Anywhere, Red Hat OpenShift Service sur AWS et les clusters Kubernetes autogérés sur les instances Amazon EC2.

Attribut	Identité du pod EKS	IRSA
Versions EKS prises en charge	Toutes les versions de clusters EKS prises en charge. Pour les versions de plateforme spécifiques, consultez the section called "Versions du cluster de l'identité du pod EKS".	Toutes les versions de clusters EKS prises en charge.

Rôles IAM pour les comptes de service

> **Tip**
>
> Inscrivez-vous aux prochains ateliers Amazon EKS.

Les applications situées dans les conteneurs d'un pod peuvent utiliser un AWS SDK ou la AWS CLI pour envoyer des demandes d'API aux AWS services à l'aide des AWS autorisations Identity and Access Management (IAM). Les applications doivent signer leurs demandes AWS d'API avec des AWS informations d'identification. Les rôles IAM pour les comptes de service (IRSA) permettent de gérer les informations d'identification de vos applications, de la même manière que les profils d' EC2 instance Amazon fournissent des informations d'identification aux instances Amazon EC2 . Au lieu de créer et de distribuer vos AWS informations d'identification aux conteneurs ou d'utiliser le rôle de l' EC2 instance Amazon, vous associez un rôle IAM à un compte de service Kubernetes et configurez vos Pods pour utiliser le compte de service. Vous ne pouvez pas utiliser de rôles IAM pour des comptes de service dotés de clusters locaux pour Amazon EKS on AWS Outposts.

Les rôles IAM pour les comptes de service offrent les bénéfices suivants :

- Moindre privilège : vous pouvez définir les autorisations IAM sur un compte de service et seuls les pods qui utilisent ce compte de service ont accès à ces autorisations. Cette fonctionnalité élimine également le besoin de solutions tierces telles que `kiam` ou `kube2iam`.
- Isolation des informations d'identification : lorsque l'accès à l'Amazon EC2 Instance Metadata Service (IMDS) est restreint, les conteneurs d'un pod peuvent uniquement récupérer les informations d'identification pour le rôle IAM associé au compte de service utilisé par le conteneur. Un conteneur n'a jamais accès aux informations d'identification qui sont utilisées par d'autres

conteneurs dans d'autres pods. Si IMDS n'est pas restreint, les conteneurs du Pod ont également accès au rôle IAM du nœud Amazon EKS et les conteneurs peuvent être en mesure d'accéder aux informations d'identification des rôles IAM d'autres Pods sur le même nœud. Pour plus d'informations, consultez Restreindre l'accès au profil d'instance affecté au composant master.

> **Note**
>
> Les pods configurés avec `hostNetwork: true` auront toujours un accès IMDS, mais la CLI AWS SDKs and utilisera les informations d'identification IRSA lorsqu'elle est activée.

- Auditabilité — La journalisation des accès et des événements est disponible AWS CloudTrail pour garantir un audit rétrospectif.

> **Important**
>
> Les conteneurs ne constituent pas une limite de sécurité, et l'utilisation de rôles IAM pour les comptes de service ne change rien à cela. Les pods attribués au même nœud partageront un noyau et éventuellement d'autres ressources en fonction de la configuration de votre pod. Bien que les pods s'exécutant sur des nœuds distincts soient isolés au niveau de la couche de calcul, certaines applications de nœuds disposent d'autorisations supplémentaires dans l'API Kubernetes qui dépassent le champ d'application d'une instance individuelle. Quelques exemples : `kubelet`, `kube-proxy`, les pilotes de stockage CSI ou vos propres applications Kubernetes.

Activez les rôles IAM pour les comptes de service en suivant les procédures suivantes :

1. Créez un fournisseur IAM OIDC pour votre cluster : vous ne devez effectuer cette procédure qu'une seule fois pour chaque cluster.

   > **Note**
   >
   > Si vous avez activé le point de terminaison d'un VPC EKS, le point de terminaison du service OIDC EKS n'était pas accessible depuis l'intérieur de ce VPC. Par conséquent, les opérations telles que la création d'un fournisseur OIDC avec `eksctl` dans le VPC n'aboutiront pas et entraîneront une expiration lors de la tentative de demande de

`https://oidc.eks.region.amazonaws.com`. Voici un exemple de message d'erreur :

```
server cant find oidc.eks.region.amazonaws.com: NXDOMAIN
```

Pour terminer cette étape, vous pouvez exécuter la commande en dehors du VPC, par exemple dans AWS CloudShell ou sur un ordinateur connecté à Internet. Vous pouvez également créer un résolveur conditionnel à horizon divisé dans le VPC, tel que Route 53 Resolver, afin d'utiliser un résolveur différent pour l'URL de l'émetteur OIDC et de ne pas utiliser le DNS VPC pour celle-ci. Pour un exemple de transfert conditionnel dans CoreDNS, consultez la demande de fonctionnalité Amazon EKS sur. GitHub

2. Attribuer des rôles IAM aux comptes de service Kubernetes : suivez cette procédure pour chaque ensemble unique d'autorisations que vous souhaitez attribuer à une application.
3. Configurer les pods pour utiliser un compte de service Kubernetes : effectuez cette procédure pour chaque pod devant accéder aux services. AWS
4. Utiliser IRSA avec le AWS SDK : vérifiez que la charge de travail utilise un AWS SDK d'une version prise en charge et qu'elle utilise la chaîne d'identification par défaut.

Informations générales sur IAM, Kubernetes et OpenID Connect (OIDC)

En 2014, AWS Identity and Access Management a ajouté la prise en charge des identités fédérées à l'aide d'OpenID Connect (OIDC). Cette fonctionnalité vous permet d'authentifier les appels d' AWS API auprès des fournisseurs d'identité pris en charge et de recevoir un jeton Web OIDC JSON (JWT) valide. Vous pouvez transmettre ce jeton à l'opération de l'`AssumeRoleWithWebIdentity`API AWS STS et recevoir des informations d'identification de rôle temporaires IAM. Vous pouvez utiliser ces informations d'identification pour interagir avec n'importe quel AWS service, y compris Amazon S3 et DynamoDB.

Chaque jeton JWT est signé par une paire de clés de signature. Les clés sont servies sur le fournisseur OIDC géré par Amazon EKS et la clé privée change tous les 7 jours. Amazon EKS conserve les clés publiques jusqu'à leur expiration. Si vous connectez des clients OIDC externes, sachez que vous devez actualiser les clés de signature avant l'expiration de la clé publique. Apprenez à récupérer les clés de signature pour valider les jetons OIDC.

Kubernetes utilise depuis longtemps les comptes de service comme son propre système d'identité interne. Les pods peuvent s'authentifier auprès du serveur d'API Kubernetes à l'aide d'un jeton

monté automatiquement (qui était un JWT non OIDC) que seul le serveur d'API Kubernetes a pu valider. Ces jetons de compte de service hérités n'expirent pas et la rotation de la clé de signature est un processus difficile. Dans la version Kubernetes `1.12`, une nouvelle fonctionnalité `ProjectedServiceAccountToken` a été ajoutée. Cette fonctionnalité est un jeton Web JSON OIDC qui contient également l'identité du compte de service et prend en charge une audience configurable.

Amazon EKS héberge un point de terminaison de découverte OIDC public pour chaque cluster qui contient les clés de signature des jetons Web JSON `ProjectedServiceAccountToken` afin que les systèmes externes, tels que IAM, puissent valider et accepter les jetons OIDC émis par Kubernetes.

Créer un fournisseur d'identité OIDC IAM pour votre cluster

Votre cluster a une URL de diffuseur OpenID Connect (OIDC) qui lui est associée. Pour utiliser les rôles AWS Identity and Access Management (IAM) pour les comptes de service, un fournisseur IAM OIDC doit exister pour l'URL de l'émetteur OIDC de votre cluster.

Conditions préalables

- Un cluster Amazon EKS existant. Pour en déployer un, consultez Démarrer.
- Version `2.12.3` ou version ultérieure `1.27.160` ou version ultérieure de l'interface de ligne de AWS commande (AWS CLI) installée et configurée sur votre appareil ou AWS CloudShell. Pour vérifier votre version actuelle, utilisez `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Les gestionnaires de packages tels que `yum` Homebrew pour macOS ont souvent plusieurs versions de retard sur la dernière version de la AWS CLI. `apt-get` Pour installer la dernière version, consultez la section Installation et configuration rapide avec aws configure dans le Guide de l'utilisateur de l'interface de ligne de AWS commande. La version de la AWS CLI installée AWS CloudShell peut également avoir plusieurs versions de retard par rapport à la dernière version. Pour le mettre à jour, consultez la section Installation de la AWS CLI dans votre répertoire de base dans le guide de AWS CloudShell l'utilisateur.
- L'outil de ligne de commande `kubectl` est installé sur votre appareil ou AWS CloudShell. La version peut correspondre à celle utilisée par votre cluster Kubernetes, ou différer d'au plus une version mineure, qu'elle soit antérieure ou plus récente. Par exemple, si la version de votre cluster est `1.29`, vous pouvez utiliser la version `kubectl` `1.28`, `1.29` ou `1.30`. Pour installer ou mettre à niveau `kubectl`, veuillez consulter the section called "Configurer kubectl et eksctl".
- Un fichier existant `kubectl config` qui contient la configuration de votre cluster. Pour créer un fichier `kubectl config`, consultez the section called "Accès au cluster avec kubectl".

Vous pouvez créer un fournisseur IAM OIDC pour votre cluster en utilisant `eksctl` ou la AWS Management Console.

Créer un fournisseur OIDC (eksctl)

1. Version `0.215.0` ou ultérieure de l'outil de ligne de commande `eksctl` installée sur votre appareil ou AWS CloudShell. Pour installer ou mettre à jour `eksctl`, veuillez consulter Installation dans la documentation de `eksctl`.
2. Déterminez l'identifiant de l'émetteur OIDC pour votre cluster.

   Récupérez l'ID émetteur OIDC de votre cluster et stockez-le dans une variable. Remplacez `<my-cluster>` par votre propre valeur.

   ```
   cluster_name=<my-cluster>
   oidc_id=$(aws eks describe-cluster --name $cluster_name --query
    "cluster.identity.oidc.issuer" --output text | cut -d '/' -f 5)
   echo $oidc_id
   ```

3. Déterminez si un fournisseur IAM OIDC avec l'ID d'émetteur de votre cluster figure déjà dans votre compte.

   ```
   aws iam list-open-id-connect-providers | grep $oidc_id | cut -d "/" -f4
   ```

   Si la sortie est renvoyée, cela signifie que vous disposez déjà d'un fournisseur IAM OIDC pour votre cluster. Vous pouvez donc ignorer l'étape suivante. Si aucune sortie n'est renvoyée, vous devez créer un fournisseur OIDC IAM pour votre cluster.
4. Créez votre fournisseur d'identité OIDC IAM pour votre cluster avec la commande suivante.

   ```
   eksctl utils associate-iam-oidc-provider --cluster $cluster_name --approve
   ```

   > **Note**
   >
   > Si vous avez activé le point de terminaison d'un VPC EKS, le point de terminaison du service OIDC EKS n'était pas accessible depuis l'intérieur de ce VPC. Par conséquent, vos opérations telles que la création d'un fournisseur OIDC avec `eksctl` dans le VPC ne fonctionneront pas et entraîneront un délai d'expiration. Voici un exemple de message d'erreur :

```
** server cant find oidc.eks.<region-code>.amazonaws.com: NXDOMAIN
```

Pour terminer cette étape, vous pouvez exécuter la commande en dehors du VPC, par exemple dans AWS CloudShell ou sur un ordinateur connecté à Internet. Vous pouvez également créer un résolveur conditionnel à horizon divisé dans le VPC, tel que Route 53 Resolver, afin d'utiliser un résolveur différent pour l'URL de l'émetteur OIDC et de ne pas utiliser le DNS VPC pour celle-ci. Pour un exemple de transfert conditionnel dans CoreDNS, consultez la demande de fonctionnalité Amazon EKS sur. GitHub

Créer un fournisseur OIDC (AWS console)

1. Ouvrez la console Amazon EKS.
2. Dans le volet de gauche, sélectionnez Clusters, puis sélectionnez le nom de votre cluster sur la page Clusters.
3. Dans la section Détails de l'onglet Présentation, notez la valeur de l'URL du fournisseur d'OpenID Connect.
4. Ouvrez la console IAM à https://console.aws.amazon.com/iam/ l'adresse.
5. Dans le panneau de navigation, sélectionnez Identity Providers (Fournisseurs d'identité) sous Access management (Gestion des accès). Si un fournisseur est répertorié et correspond à l'URL de votre cluster, alors vous avez déjà un fournisseur pour votre cluster. Si aucun fournisseur ne correspond à l'URL de votre cluster, vous devez en créer un.
6. Pour créer un fournisseur, cliquez sur Ajouter un fournisseur.
7. Pour le type de fournisseur, sélectionnez OpenID Connect.
8. Pour l'URL du fournisseur, saisissez l'URL du fournisseur OIDC pour votre cluster.
9. Pour Audience, saisissez `sts.amazonaws.com`.
10. (Facultatif) Ajoutez des balises, par exemple une balise permettant d'identifier le cluster associé à ce fournisseur.
11. Choisissez Ajouter un fournisseur.

Étape suivante : the section called "Attribution d'un rôle IAM"

Attribution de rôles IAM aux comptes de service Kubernetes

Cette rubrique explique comment configurer un compte de service Kubernetes pour qu'il assume un rôle AWS Identity and Access Management (IAM). Tous les pods configurés pour utiliser le compte de service peuvent ensuite accéder à tout AWS service auquel le rôle est autorisé à accéder.

Conditions préalables

- Un cluster existant. Si vous n'en avez pas, vous pouvez en créer un en suivant l'un des guides Démarrer.
- Un fournisseur IAM OpenID Connect (OIDC) existant pour votre cluster. Pour déterminer si vous en avez déjà un ou comment en créer un, consultez the section called "Fournisseur IAM OIDC".
- Version `2.12.3` ou version ultérieure `1.27.160` ou version ultérieure de l'interface de ligne de AWS commande (AWS CLI) installée et configurée sur votre appareil ou AWS CloudShell. Pour vérifier votre version actuelle, utilisez `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Les gestionnaires de packages tels que `yum` Homebrew pour macOS ont souvent plusieurs versions de retard sur la dernière version de la AWS CLI. `apt-get` Pour installer la dernière version, consultez la section Installation et configuration rapide avec aws configure dans le Guide de l'utilisateur de l'interface de ligne de AWS commande. La version de la AWS CLI installée AWS CloudShell peut également avoir plusieurs versions de retard par rapport à la dernière version. Pour le mettre à jour, consultez la section Installation de la AWS CLI dans votre répertoire de base dans le guide de AWS CloudShell l'utilisateur.
- L'outil de ligne de commande `kubectl` est installé sur votre appareil ou AWS CloudShell. La version peut correspondre à celle utilisée par votre cluster Kubernetes, ou différer d'au plus une version mineure, qu'elle soit antérieure ou plus récente. Par exemple, si la version de votre cluster est `1.29`, vous pouvez utiliser la version `kubectl` `1.28`, `1.29` ou `1.30`. Pour installer ou mettre à niveau `kubectl`, veuillez consulter the section called "Configurer kubectl et eksctl".
- Un fichier existant `kubectl config` qui contient la configuration de votre cluster. Pour créer un fichier `kubectl config`, consultez the section called "Accès au cluster avec kubectl".

Étape 1 : créer une politique IAM

Si vous souhaitez associer une politique IAM existante à votre rôle IAM, passez à l'étape suivante.

1. Créez une politique IAM. Vous pouvez créer votre propre politique ou copier une politique AWS gérée qui accorde déjà certaines des autorisations dont vous avez besoin et la personnaliser en

fonction de vos besoins spécifiques. Pour plus d'informations, consultez Création de politiques IAM dans le Guide de l'utilisateur IAM.

2. Créez un fichier qui inclut les autorisations pour les AWS services auxquels vous souhaitez que vos pods accèdent. Pour obtenir la liste de toutes les actions pour tous les AWS services, consultez la référence d'autorisation des services.

   Vous pouvez exécuter la commande suivante pour créer un exemple de fichier de politique autorisant l'accès en lecture seule à un compartiment Amazon S3. Vous pouvez, si vous le souhaitez, stocker des informations de configuration ou un script d'amorçage dans ce compartiment et les conteneurs de votre pod pourront lire le fichier dans le compartiment et le charger dans votre application. Si vous souhaitez créer cet exemple de politique, copiez le contenu suivant sur votre appareil. *my-pod-secrets-bucket*Remplacez-le par le nom de votre compartiment et exécutez la commande.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::my-pod-secrets-bucket"
        }
    ]
}
```

3. Créez la politique IAM.

```
aws iam create-policy --policy-name my-policy --policy-document file://my-policy.json
```

Étape 2 : créer et associer un rôle IAM

Créez un rôle IAM, puis associez-le à un compte de service Kubernetes. Vous pouvez utiliser l'une ou l'autre `eksctl` option ou la AWS CLI.

Création et association du rôle (eksctl)

Cette commande `eksctl` crée un compte de service Kubernetes dans l'espace de noms spécifié, crée un rôle IAM (s'il n'existe pas) portant le nom indiqué, attache une politique IAM existante au rôle, puis annote le compte de service avec l'ARN du rôle IAM. Veillez à remplacer les valeurs

d'exemple de cette commande par vos propres valeurs. Pour installer ou mettre à jour `eksctl`, veuillez consulter Installation dans la documentation de `eksctl`.

```
eksctl create iamserviceaccount --name my-service-account --namespace default --cluster
 my-cluster --role-name my-role \
    --attach-policy-arn arn:aws: iam::111122223333:policy/my-policy --approve
```

> **Important**
>
> Si le rôle ou le compte de service existe déjà, la commande précédente peut échouer. `eksctl` propose différentes options que vous pouvez fournir dans ces situations. Pour de plus amples informations, exécutez `eksctl create iamserviceaccount --help`.

Créer et associer un rôle (AWS CLI)

Si vous avez déjà un compte de service Kubernetes devant assumer un rôle IAM, vous pouvez ignorer cette étape.

1. Créez un compte de service Kubernetes. Copiez les contenus suivants sur votre appareil. *my-service-account*Remplacez-le par le nom de votre choix et *default* par un autre espace de noms, si nécessaire. Si vous modifiez*default*, l'espace de noms doit déjà exister.

   ```
   cat >my-service-account.yaml <<EOF
   apiVersion: v1
   kind: ServiceAccount
   metadata:
     name: my-service-account
     namespace: default
   EOF
   kubectl apply -f my-service-account.yaml
   ```

2. Définissez votre identifiant de AWS compte sur une variable d'environnement à l'aide de la commande suivante.

   ```
   account_id=$(aws sts get-caller-identity --query "Account" --output text)
   ```

3. Définissez le fournisseur d'identité OIDC de votre cluster dans une variable d'environnement à l'aide de la commande suivante. Remplacez *my-cluster* par le nom de votre cluster.

```
oidc_provider=$(aws eks describe-cluster --name my-cluster --region $AWS_REGION --
query "cluster.identity.oidc.issuer" --output text | sed -e "s/^https:\/\///")
```

4. Définissez des variables pour l'espace de noms et le nom du compte de service. *my-service-account*Remplacez-le par le compte de service Kubernetes dont vous souhaitez assumer le rôle. Remplacez *default* par l'espace de noms du compte de service.

```
export namespace=default
export service_account=my-service-account
```

5. Exécutez la commande suivante pour créer un fichier de stratégie d'approbation pour le rôle IAM. Si vous souhaitez autoriser tous les comptes de service d'un espace de noms à utiliser le rôle, copiez le contenu suivant sur votre appareil. Remplacez *StringEquals* par `StringLike` et remplacez *$service_account* par`*`. Vous pouvez ajouter plusieurs entrées dans les conditions `StringEquals` ou `StringLike` pour autoriser plusieurs comptes de service ou espaces de noms à endosser le rôle. Pour autoriser les rôles provenant d'un AWS compte différent de celui dans lequel se trouve votre cluster à assumer le rôle, consultez the section called "IAM inter-comptes" pour plus d'informations.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::123456789012:oidc-provider/$oidc_provider"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "$oidc_provider:aud": "sts.amazonaws.com",
          "$oidc_provider:sub": "system:serviceaccount:$namespace:$service_account"
        }
      }
    }
  ]
}
```

6. Créez le rôle. Remplacez *my-role* avec un nom pour votre rôle IAM, et *my-role-description* avec une description de votre rôle.

```
aws iam create-role --role-name my-role --assume-role-policy-document file://trust-
relationship.json --description "my-role-description"
```

7. Liez une politique IAM à votre rôle. Remplacez *my-role* par le nom de votre rôle IAM et *my-policy* par le nom d'une politique existante que vous avez créée.

```
aws iam attach-role-policy --role-name my-role --policy-arn=arn:aws: iam::
$account_id:policy/my-policy
```

8. Annotez votre compte de service avec l'Amazon Resource Name (ARN) du rôle IAM que vous souhaitez que le compte de service endosse. Remplacez *my-role* par le nom de votre rôle IAM existant. Supposons que vous ayez autorisé un rôle provenant d'un AWS compte différent de celui dans lequel se trouve votre cluster à assumer ce rôle lors d'une étape précédente. Assurez-vous ensuite de spécifier le AWS compte et le rôle de l'autre compte. Pour de plus amples informations, veuillez consulter the section called "IAM inter-comptes".

```
kubectl annotate serviceaccount -n $namespace $service_account eks.amazonaws.com/
role-arn=arn:aws: iam::$account_id:role/my-role
```

9. (Facultatif) Configurez le point de terminaison du service AWS Security Token pour un compte de service. AWS recommande d'utiliser un point de terminaison AWS STS régional au lieu du point de terminaison global. Cela réduit la latence, fournit une redondance intégrée et augmente la validité des jetons de session.

Étape 3 : confirmer la configuration

1. Vérifiez que la stratégie d'approbation du rôle IAM est correctement configurée.

```
aws iam get-role --role-name my-role --query Role.AssumeRolePolicyDocument
```

L'exemple qui suit illustre un résultat.

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

```
                "Federated": "arn:aws:iam::111122223333:oidc-provider/oidc.eks.us-
east-1.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
            },
            "Action": "sts:AssumeRoleWithWebIdentity",
            "Condition": {
                "StringEquals": {
                    "oidc.eks.us-east-1.amazonaws.com/id/
EXAMPLED539D4633E53DE1B71EXAMPLE:sub": "system:serviceaccount:default:my-service-
account",
                    "oidc.eks.us-east-1.amazonaws.com/id/
EXAMPLED539D4633E53DE1B71EXAMPLE:aud": "sts.amazonaws.com"
                }
            }
        }
    ]
}
```

2. Vérifiez que la stratégie que vous avez associée à votre rôle lors d'une étape précédente est associée au rôle.

```
aws iam list-attached-role-policies --role-name my-role --query
 "AttachedPolicies[].PolicyArn" --output text
```

   L'exemple qui suit illustre un résultat.

```
 arn:aws: iam::111122223333:policy/my-policy
```

3. Définissez une variable pour stocker l'Amazon Resource Name (ARN) de la stratégie que vous souhaitez utiliser. *my-policy*Remplacez-le par le nom de la politique pour laquelle vous souhaitez confirmer les autorisations.

```
export policy_arn=arn:aws: iam::111122223333:policy/my-policy
```

4. Affichez la version par défaut de la stratégie.

```
aws iam get-policy --policy-arn $policy_arn
```

   L'exemple qui suit illustre un résultat.

```
{
    "Policy": {
```

```
        "PolicyName": "my-policy",
        "PolicyId": "EXAMPLEBIOWGLDEXAMPLE",
        "Arn": "arn:aws: iam::111122223333:policy/my-policy",
        "Path": "/",
        "DefaultVersionId": "v1",
        [...]
    }
}
```

5. Consultez le contenu de la stratégie pour vous assurer qu'elle inclut toutes les autorisations nécessaires pour votre pod. Si nécessaire, remplacez *1* la commande suivante par la version renvoyée dans la sortie précédente.

```
aws iam get-policy-version --policy-arn $policy_arn --version-id v1
```

L'exemple qui suit illustre un résultat.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::my-pod-secrets-bucket"
        }
    ]
}
```

Si vous avez créé l'exemple de stratégie lors d'une étape précédente, le résultat est le même. Si vous avez créé une politique différente, le *example* contenu est différent.

6. Confirmez que le compte de service Kubernetes est annoté avec le rôle.

```
kubectl describe serviceaccount my-service-account -n default
```

L'exemple qui suit illustre un résultat.

```
Name:                my-service-account
Namespace:           default
Annotations:         eks.amazonaws.com/role-arn: arn:aws: iam::111122223333:role/my-
role
```

```
Image pull secrets:  <none>
Mountable secrets:   my-service-account-token-qqjfl
Tokens:              my-service-account-token-qqjfl
[...]
```

Étapes suivantes

- the section called "Attribuer à un pod"

Configurer les pods pour utiliser un compte de service Kubernetes

Si un Pod doit accéder à AWS des services, vous devez le configurer pour utiliser un compte de service Kubernetes. Le compte de service doit être associé à un rôle AWS Identity and Access Management (IAM) autorisé à accéder aux services. AWS

- Un cluster existant. Si vous n'en avez pas, vous pouvez en créer un à l'aide de l'un des guides disponibles dans Démarrer.
- Un fournisseur IAM OpenID Connect (OIDC) existant pour votre cluster. Pour déterminer si vous en avez déjà un ou comment en créer un, consultez the section called "Fournisseur IAM OIDC".
- Un compte de service Kubernetes existant associé à un rôle IAM. Le compte de service doit être annoté avec l'Amazon Resource Name (ARN) du rôle IAM. Le rôle doit être associé à une politique IAM contenant les autorisations dont vous souhaitez que vos pods disposent pour utiliser les AWS services. Pour plus d'informations sur la façon de créer et de configurer le compte de service et le rôle, consultez the section called "Attribution d'un rôle IAM".
- Version `2.12.3` ou version ultérieure `1.27.160` ou version ultérieure de l'interface de ligne de AWS commande (AWS CLI) installée et configurée sur votre appareil ou AWS CloudShell. Pour vérifier votre version actuelle, utilisez `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Les gestionnaires de packages tels que `yum` Homebrew pour macOS ont souvent plusieurs versions de retard sur la dernière version de la AWS CLI. `apt-get` Pour installer la dernière version, consultez la section Installation et configuration rapide avec aws configure dans le Guide de l'utilisateur de l'interface de ligne de AWS commande. La version de la AWS CLI installée AWS CloudShell peut également avoir plusieurs versions de retard par rapport à la dernière version. Pour le mettre à jour, consultez la section Installation de la AWS CLI dans votre répertoire de base dans le guide de AWS CloudShell l'utilisateur.
- L'outil de ligne de commande `kubectl` est installé sur votre appareil ou AWS CloudShell. La version peut correspondre à celle utilisée par votre cluster Kubernetes, ou différer d'au plus une

version mineure, qu'elle soit antérieure ou plus récente. Par exemple, si la version de votre cluster est `1.29`, vous pouvez utiliser la version `kubectl` `1.28`, `1.29` ou `1.30`. Pour installer ou mettre à niveau `kubectl`, veuillez consulter the section called "Configurer kubectl et eksctl".

- Un fichier existant `kubectl config` qui contient la configuration de votre cluster. Pour créer un fichier `kubectl config`, consultez the section called "Accès au cluster avec kubectl".

1. Utilisez la commande suivante pour créer un manifeste de déploiement que vous pouvez déployer sur un pod afin de confirmer la configuration. Remplacez les exemples de valeurs par vos propres valeurs.

```
cat >my-deployment.yaml <<EOF
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-app
spec:
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
    spec:
      serviceAccountName: my-service-account
      containers:
      - name: my-app
        image: public.ecr.aws/nginx/nginx:X.XX
EOF
```

2. Appliquez le manifeste à votre cluster.

```
kubectl apply -f my-deployment.yaml
```

3. Vérifiez que les variables d'environnement requises existent pour votre pod.

   a. Affichez les pods qui ont été déployés avec le déploiement à l'étape précédente.

```
kubectl get pods | grep my-app
```

   L'exemple qui suit illustre un résultat.

```
my-app-6f4dfff6cb-76cv9   1/1     Running   0          3m28s
```

b. Affichez l'ARN du rôle IAM utilisé par le pod.

```
kubectl describe pod my-app-6f4dfff6cb-76cv9 | grep AWS_ROLE_ARN:
```

L'exemple qui suit illustre un résultat.

```
AWS_ROLE_ARN: arn:aws: iam::111122223333:role/my-role
```

L'ARN du rôle doit correspondre à l'ARN du rôle avec lequel vous avez annoté le compte de service existant. Pour en savoir plus sur l'annotation du compte de service, consultez the section called "Attribution d'un rôle IAM".

c. Vérifiez que le pod dispose d'un fichier de jeton d'identité Web monté.

```
kubectl describe pod my-app-6f4dfff6cb-76cv9 | grep AWS_WEB_IDENTITY_TOKEN_FILE:
```

L'exemple qui suit illustre un résultat.

```
AWS_WEB_IDENTITY_TOKEN_FILE:  /var/run/secrets/eks.amazonaws.com/serviceaccount/
token
```

Le `kubelet` demande et stocke le jeton au nom du pod. Par défaut, `kubelet` actualise le jeton si ce dernier est supérieur à 80 % de son temps total avant le live, ou si le jeton est âgé de plus de 24 heures. Vous pouvez modifier la durée d'expiration de tout compte autre que le compte de service par défaut à l'aide des paramètres de votre spécification de pod. Pour plus d'informations, consultez la section Projection du volume des jetons de compte de service dans la documentation Kubernetes.

Le webhook d'identité du pod Amazon EKS sur le cluster surveille les pods qui utilisent un compte de service avec l'annotation suivante :

```
eks.amazonaws.com/role-arn: arn:aws: iam::111122223333:role/my-role
```

Le webhook applique les variables d'environnement précédentes à ces pods. Votre cluster n'a pas besoin d'utiliser le webhook pour configurer les variables d'environnement et

les montages de fichiers de jeton. Vous pouvez configurer manuellement les pods pour qu'ils disposent de ces variables d'environnement. Les versions prises en charge du AWS SDK recherchent d'abord ces variables d'environnement dans le fournisseur de chaîne d'informations d'identification. Les informations d'identification du rôle sont utilisées pour les pods qui répondent à ces critères.

4. Vérifiez que vos Pods peuvent interagir avec les AWS services à l'aide des autorisations que vous avez attribuées dans la politique IAM associée à votre rôle.

> **Note**
>
> Lorsqu'un pod utilise les AWS informations d'identification d'un rôle IAM associé à un compte de service, la AWS CLI ou un autre SDKs élément des conteneurs de ce pod utilisent les informations d'identification fournies par ce rôle. Si vous ne restreignez pas l'accès aux informations d'identification fournies au rôle IAM du nœud Amazon EKS, le pod a toujours accès à ces informations d'identification. Pour plus d'informations, consultez Restreindre l'accès au profil d'instance affecté au composant master.

Si vos pods ne peuvent pas interagir avec les services comme prévu, suivez les étapes suivantes pour vérifier que tout est correctement configuré.

a. Vérifiez que vos Pods utilisent une version du AWS SDK qui permet d'assumer un rôle IAM via un fichier de jeton d'identité Web OpenID Connect. Pour de plus amples informations, veuillez consulter the section called "Kits SDK pris en charge".

b. Vérifiez que le déploiement utilise le compte de service.

```
kubectl describe deployment my-app | grep "Service Account"
```

L'exemple qui suit illustre un résultat.

```
Service Account:  my-service-account
```

c. Si vos pods ne peuvent toujours pas accéder aux services, passez en revue les étapes décrites dans Attribuer des rôles IAM aux comptes de service Kubernetes pour vérifier que votre rôle et votre compte de service sont correctement configurés.

Configurer le point de terminaison du service de jetons de sécurité AWS pour un compte de service

Si vous utilisez un compte de service Kubernetes avec des rôles IAM pour les comptes de service, vous pouvez configurer le type de point de terminaison du service de jetons de sécurité AWS utilisé par le compte de service.

AWS recommande d'utiliser les points d'arrivée régionaux STS AWS plutôt que le point d'arrivée global. Cela réduit la latence, fournit une redondance intégrée et augmente la validité des jetons de session. Le service de jetons de sécurité AWS doit être actif dans la région AWS où le pod est exécuté. De plus, votre application doit disposer d'une redondance intégrée pour une autre région AWS en cas de défaillance du service dans la région AWS. Pour plus d'informations, consultez Gestion d'AWS STS dans une région AWS dans le Guide de l'utilisateur IAM.

- Un cluster existant. Si vous n'en avez pas, vous pouvez en créer un à l'aide de l'un des guides disponibles dans Démarrer.
- Un fournisseur IAM OIDC existant pour votre cluster. Pour de plus amples informations, consultez the section called "Fournisseur IAM OIDC".
- Un compte de service Kubernetes existant configuré pour être utilisé avec la fonction Amazon EKS IAM pour les comptes de service.

Les exemples suivants utilisent tous le compte de service Kubernetes aws-node utilisé par le plug-in Amazon VPC CNI. Vous pouvez remplacer les *valeurs d'exemple* par vos propres comptes de service, pods, espaces de noms et autres ressources.

1. Sélectionnez un pod qui utilise un compte de service dont vous souhaitez modifier le point de terminaison. Déterminez dans quelle région AWS le Pod fonctionne. Remplacez *aws-node-6mfgv* par le nom de votre pod et *kube-system* par l'espace de noms de votre pod.

   ```
   kubectl describe pod aws-node-6mfgv -n kube-system |grep Node:
   ```

   L'exemple qui suit illustre un résultat.

   ```
   ip-192-168-79-166.us-west-2/192.168.79.166
   ```

   Dans la sortie précédente, le Pod s'exécute sur un nœud dans la région AWS us-west-2.
2. Déterminez le type de point de terminaison utilisé par le compte de service du pod.

```
kubectl describe pod aws-node-6mfgv -n kube-system |grep AWS_STS_REGIONAL_ENDPOINTS
```

L'exemple qui suit illustre un résultat.

```
AWS_STS_REGIONAL_ENDPOINTS: regional
```

Si le point de terminaison actuel est international, alors `global` est renvoyé en sortie. Si aucune sortie n'est renvoyée, le type de point de terminaison par défaut est en cours d'utilisation et n'a pas été remplacé.

3. Si la version de votre cluster ou celle de votre plateforme est identique ou ultérieure à celles répertoriées dans le tableau, vous pouvez modifier le type de point de terminaison utilisé par votre compte de service pour le faire passer du type par défaut à un type différent à l'aide de l'une des commandes suivantes. Remplacez *`aws-node`* par le nom de votre compte de service et *`kube-system`* par l'espace de noms de votre compte de service.

   - Si votre type de point de terminaison par défaut ou actuel est international et que vous souhaitez le remplacer par régional :

     ```
     kubectl annotate serviceaccount -n kube-system aws-node eks.amazonaws.com/sts-regional-endpoints=true
     ```

     Si vous utilisez des rôles IAM pour les comptes de service afin de générer des URL S3 pré-signées dans votre application exécutée dans les conteneurs Pods, le format de l'URL pour les points de terminaison régionaux est similaire à l'exemple suivant :

     ```
     https://bucket.s3.us-west-2.amazonaws.com/path?...&X-Amz-Credential=your-access-key-id/date/us-west-2/s3/aws4_request&...
     ```

   - Si votre type de point de terminaison par défaut ou actuel est régional et que vous souhaitez le remplacer par international :

     ```
     kubectl annotate serviceaccount -n kube-system aws-node eks.amazonaws.com/sts-regional-endpoints=false
     ```

     Si votre application envoie explicitement des requêtes aux points de terminaison globaux STS AWS et que vous ne remplacez pas le comportement par défaut consistant à utiliser les points de terminaison régionaux dans les clusters Amazon EKS, les requêtes échoueront et généreront

une erreur. Pour de plus amples informations, consultez the section called "Les conteneurs Pod reçoivent l'erreur suivante : `An error occurred (SignatureDoesNotMatch) when calling the GetCallerIdentity operation: Credential should be scoped to a valid region`".

Si vous utilisez des rôles IAM pour les comptes de service afin de générer des URL S3 pré-signées dans votre application exécutée dans les conteneurs Pods, le format de l'URL pour les points de terminaison globaux est similaire à l'exemple suivant :

```
https://bucket.s3.amazonaws.com/path?...&X-Amz-Credential=your-access-key-id/date/
us-west-2/s3/aws4_request&...
```

Si votre automatisation nécessite une URL pré-signée dans un certain format ou si votre application ou les dépendances en aval qui utilisent des URL pré-signées ont des attentes concernant la région AWS ciblée, effectuez les modifications nécessaires pour utiliser le point de terminaison STS AWS approprié.

4. Supprimez et recréez tous les pods existants associés au compte de service pour appliquer les variables d'environnement des informations d'identification. Le hook web en mutation ne les applique pas aux qui sont déjà en cours d'exécution. Vous pouvez remplacer *pods*, *kube-system* et *-l k8s-app=aws-node* par les informations relatives aux pods pour lesquels vous avez défini votre annotation.

```
kubectl delete Pods -n kube-system -l k8s-app=aws-node
```

5. Vérifiez que les pods ont tous redémarré.

```
kubectl get Pods -n kube-system -l k8s-app=aws-node
```

6. Affichez les variables d'environnement pour l'un des pods. Vérifiez que la valeur `AWS_STS_REGIONAL_ENDPOINTS` est celle que vous avez définie à une étape précédente.

```
kubectl describe pod aws-node-kzbtr -n kube-system |grep AWS_STS_REGIONAL_ENDPOINTS
```

L'exemple qui suit illustre un résultat.

```
AWS_STS_REGIONAL_ENDPOINTS=regional
```

Authentifiez-vous sur un autre compte avec IRSA

Vous pouvez configurer les autorisations IAM entre comptes en créant un fournisseur d'identité à partir du cluster d'un autre compte ou en utilisant les opérations `AssumeRole` chaînées. Dans les exemples suivants, le compte A possède un cluster Amazon EKS qui prend en charge les rôles IAM pour les comptes de service. Les pods qui s'exécutent sur ce cluster doivent assumer les autorisations IAM du compte B.

Example Créer un fournisseur d'identité à partir du cluster d'un autre compte

Example

Dans cet exemple, le Compte A fournit au Compte B l'URL de l'émetteur OpenID Connect (OIDC) à partir de son cluster. Le compte B suit les instructions de la section Créer un fournisseur IAM OIDC pour votre cluster et the section called "Attribution d'un rôle IAM" utilise l'URL d'émetteur OIDC du cluster du compte A. Ensuite, un administrateur de cluster annote le compte de service dans le cluster du compte A pour utiliser le rôle du compte B (*444455556666*).

```
apiVersion: v1
kind: ServiceAccount
metadata:
  annotations:
    eks.amazonaws.com/role-arn: arn:aws: iam::444455556666:role/account-b-role
```

Example Utiliser des opérations **`AssumeRole`** chaînées

Example

Dans cet exemple, le compte B crée une politique IAM avec les autorisations à accorder aux pods du cluster du Compte A. Le compte B (*444455556666*) associe cette politique à un rôle IAM doté d'une relation de confiance qui autorise l'`AssumeRole`accès au compte A (*111122223333*).

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:root"
      },
      "Action": "sts:AssumeRole",
```

```
      "Condition": {}
    }
  ]
}
```

Le compte A crée un rôle avec une politique d'approbation qui récupère les informations d'identification du fournisseur d'identité créé avec l'adresse de l'émetteur OIDC.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::111122223333:oidc-provider/oidc.eks.us-
east-1.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "oidc.eks.us-east-1.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE:aud":
 "sts.amazonaws.com"
        }
      }
    }
  ]
}
```

Si votre cluster se trouve dans une autre , ou si vous avez copié les images dans votre propre référentiel lors d'une étape précédente, effectuez les étapes suivantes.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource": "arn:aws:iam::444455556666:role/account-b-role"
        }
    ]
}
```

Le code d'application permettant aux pods d'assumer le rôle du compte B utilise deux profils : `account_b_role` et `account_a_role`. Le profil `account_b_role` utilise le profil `account_a_role` comme source. Pour la AWS CLI, le `~/.aws/config` fichier est similaire au suivant.

```
[profile account_b_role]
source_profile = account_a_role
role_arn=arn:aws: iam::444455556666:role/account-b-role

[profile account_a_role]
web_identity_token_file = /var/run/secrets/eks.amazonaws.com/serviceaccount/token
role_arn=arn:aws: iam::111122223333:role/account-a-role
```

Pour spécifier des profils chaînés pour d'autres AWS SDKs, consultez la documentation du SDK que vous utilisez. Pour plus d'informations, consultez la section Outils sur lesquels vous pouvez vous appuyer AWS.

Utiliser IRSA avec le SDK AWS

Utilisation des informations d'identification

Pour utiliser les informations d'identification des rôles IAM des comptes de service (IRSA), votre code peut utiliser n'importe quel kit SDK AWS pour créer un client pour un service AWS avec un kit SDK, et par défaut, le kit SDK recherche dans une chaîne d'emplacements les informations d'identification de gestion des identités et des accès AWS à utiliser. Les informations d'identification des rôles IAM des comptes de service seront utilisées si vous ne spécifiez pas de fournisseur d'informations d'identification lors de la création du client ou de l'initialisation du kit SDK.

Cela fonctionne parce que les rôles IAM des comptes de service ont été ajoutés en tant qu'étape dans la chaîne d'informations d'identification par défaut. Si vos charges de travail utilisent actuellement des informations d'identification qui se trouvent plus haut dans la chaîne d'informations d'identification, ces informations d'identification continueront d'être utilisées même si vous configurez un rôle IAM des comptes de service pour la même charge de travail.

Le SDK échange automatiquement le jeton OIDC du compte de service contre des informations d'identification temporaires provenant du service de jetons de sécurité AWS à l'aide de l'action `AssumeRoleWithWebIdentity`. Amazon EKS et cette action du kit SDK poursuivent la rotation des informations d'identification temporaires en les renouvelant avant qu'elles n'expirent.

Lorsque vous utilisez des rôles IAM pour les comptes de service, les conteneurs de vos pods doivent utiliser une version du SDK AWS qui prend en charge l'attribution d'un rôle IAM via un fichier de jetons d'identité Web OpenID Connect. Veillez à utiliser les versions suivantes, ou ultérieures, pour votre SDK AWS :

- Java (version 2) – 2.10.11
- Java : 1.12.782
- Kit SDK AWS pour Go v1 : 1.23.13
- Kit SDK AWS pour Go v2 : toutes les versions sont prises en charge
- Python (Boto3) – 1.9.220
- Python (botocore) – 1.12.200
- AWS CLI : 1.16.232
- Node – 2.525.0 et 3.27.0
- Ruby – 3.58.0
- C++ – 1.7.174
- .NET – 3.3.659.1 – Vous devez également inclure `AWSSDK.SecurityToken`.
- PHP – 3.110.7

De nombreux modules complémentaires Kubernetes populaires, tels que Cluster Autoscaler, Routage du trafic Internet avec AWS Load Balancer Controller et le plug-in CNI Amazon VPC pour Kubernetes, prennent en charge les rôles IAM pour les comptes de service.

Pour vous assurer d'utiliser un kit SDK pris en charge, suivez les instructions d'installation relatives à votre SDK préféré de la page Tools to Build on AWS lorsque vous créez vos conteneurs.

Considérations

Java

Lorsque vous utilisez Java, vous devez inclure le module `sts` dans le chemin de classe. Pour plus d'informations, consultez WebIdentityTokenFileCredentialsProvider dans la documentation du SDK Java.

Récupérer les clés de signature pour valider les jetons OIDC

Kubernetes délivre un `ProjectedServiceAccountToken` à chaque compte de service Kubernetes. Ce jeton est un jeton OIDC, qui est lui-même un type de jeton web JSON (JWT).

Amazon EKS héberge un point de terminaison OIDC public pour chaque cluster qui contient les clés de signature du jeton afin que les systèmes externes puissent le valider.

Pour valider un `ProjectedServiceAccountToken`, vous devez récupérer les clés de signature publiques OIDC, également appelées JSON Web Key Set (JWKS). Utilisez ces clés dans votre application pour valider le jeton. Par exemple, vous pouvez utiliser la bibliothèque Python PyJWT pour valider les jetons à l'aide de ces clés. Pour plus d'informations sur le `ProjectedServiceAccountToken`, consultez the section called "Informations générales sur IAM, Kubernetes et OpenID Connect (OIDC)".

Prérequis

- Un fournisseur OpenID Connect (OIDC) existant pour la gestion des identités et des accès (IAM) AWS pour votre cluster. Pour déterminer si vous en avez déjà un, ou pour en créer un, consultez the section called "Fournisseur IAM OIDC".
- AWS CLI : Un outil de ligne de commande pour travailler avec les services AWS, y compris Amazon EKS. Pour plus d'informations, consultez la section Installation du Guide de l'utilisateur de l'interface de la ligne de commande AWS. Après avoir installé CLI AWS, nous vous recommandons de le configurer également. Pour plus d'informations, consultez la section Configuration rapide avec aws configure du Guide de l'utilisateur de l'interface de la ligne de commande AWS.

Procédure

1. Récupérez l'URL OIDC de votre cluster Amazon EKS à l'aide de la CLI AWS.

```
$ aws eks describe-cluster --name my-cluster --query 'cluster.identity.oidc.issuer'
"https://oidc.eks.us-west-2.amazonaws.com/id/8EBDXXXX00BAE"
```

2. Récupérez la clé de signature publique à l'aide de curl ou d'un outil similaire. Le résultat est un ensemble de clés web JSON (JWKS).

> **Important**
>
> Amazon EKS limite les appels vers le point de terminaison OIDC. Vous devez mettre en cache la clé de signature publique. Respectez l'en-tête `cache-control` inclus dans la réponse.

> **Important**
>
> Amazon EKS fait pivoter la clé de signature OIDC tous les sept jours.

```
$ curl https://oidc.eks.us-west-2.amazonaws.com/id/8EBDXXXX00BAE/keys
{"keys":
[{"kty":"RSA","kid":"2284XXXX4a40","use":"sig","alg":"RS256","n":"wklbXXXXMVfQ","e":"AQAB"}]
```

Découvrir comment l'identité du pod Amazon EKS accorde aux pods l'accès aux services AWS

Les applications dans les conteneurs d'un Pod peuvent utiliser un SDK AWS ou l'interface CLI AWS pour envoyer des requêtes API aux services AWS à l'aide des autorisations de la gestion des identités et des accès AWS (IAM). Les applications doivent signer leurs demandes d'API AWS avec les informations d'identification AWS.

Les identités du pod EKS permettent de gérer les informations d'identification de vos applications, de la même manière que les profils d'instance Amazon EC2 fournissent des informations d'identification aux instances Amazon EC2. Au lieu de créer et distribuer vos informations d'identification AWS aux conteneurs ou d'utiliser le rôle de l'instance Amazon EC2, vous associez un rôle IAM à un compte de service Kubernetes et vous configurez vos pods pour qu'ils utilisent le compte de service.

Chaque identité du pod EKS mappe un rôle à un compte de service dans un espace de noms dans le cluster spécifié. Si vous avez la même application dans plusieurs clusters, vous pouvez faire des associations identiques dans chaque cluster sans modifier la politique d'approbation du rôle.

Si un pod utilise un compte de service qui a une association, Amazon EKS définit des variables d'environnement dans les conteneurs du pod. Les variables d'environnement configurent les kits SDK AWS, y compris l'AWS CLI, pour utiliser les informations d'identification d'identité du pod EKS.

Avantages des identités du pod EKS

Les identités du pod EKS offrent les avantages suivants :

- Moindre privilège : vous pouvez définir les autorisations IAM sur un compte de service et seuls les pods qui utilisent ce compte de service ont accès à ces autorisations. Cette fonctionnalité élimine également le besoin de solutions tierces telles que `kiam` ou `kube2iam`.
- Isolation des informations d'identification : lorsque l'accès au service de métadonnées d'instance (IMDS) Amazon EC2 est restreint, les conteneurs d'un pod ne peuvent récupérer que les informations d'identification du rôle IAM associé au compte de service utilisé par le conteneur. Un conteneur n'a jamais accès aux informations d'identification qui sont utilisées par d'autres conteneurs dans d'autres pods. Si IMDS n'est pas restreint, les conteneurs du Pod ont également accès au rôle IAM du nœud Amazon EKS et les conteneurs peuvent être en mesure d'accéder aux informations d'identification des rôles IAM d'autres Pods sur le même nœud. Pour plus d'informations, consultez Restreindre l'accès au profil d'instance affecté au composant master.

> **Note**
>
> Les pods configurés avec `hostNetwork: true` auront toujours accès à IMDS, mais les kits SDK et l'AWS CLI utiliseront les informations d'identification du pod lorsque celles-ci seront activées.

- Auditabilité : l'accès et la journalisation des événements sont disponibles via AWS CloudTrail afin de faciliter l'audit rétrospectif.

> **Important**
>
> Les conteneurs ne constituent pas une frontière de sécurité, et l'utilisation de l'identité du pod ne change rien à cela. Les pods attribués au même nœud partageront un noyau et éventuellement d'autres ressources en fonction de la configuration de votre pod. Bien que les pods s'exécutant sur des nœuds distincts soient isolés au niveau de la couche de calcul, certaines applications de nœuds disposent d'autorisations supplémentaires dans l'API Kubernetes qui dépassent le champ d'application d'une instance individuelle. Quelques exemples : `kubelet`, `kube-proxy`, les pilotes de stockage CSI ou vos propres applications Kubernetes.

L'identité du pod EKS est une méthode plus simple que the section called "Informations d'identification auprès de l'IRSA", car elle n'utilise pas de fournisseurs d'identité OIDC. L'identité du pod EKS présente les améliorations suivantes :

- Opérations indépendantes : dans de nombreuses organisations, la création de fournisseurs d'identité OIDC relève de la responsabilité d'équipes différentes de celles qui administrent les clusters Kubernetes. L'identité du pod EKS présente une séparation nette des tâches, où toute la configuration des associations de l'identité du pod EKS est effectuée dans Amazon EKS et toute la configuration des autorisations IAM est effectuée dans IAM.
- Réutilisabilité : l'identité du pod EKS utilise un seul principal IAM au lieu des principaux distincts pour chaque cluster que les rôles IAM des comptes de service utilisent. Votre administrateur IAM ajoute le principal suivant à la politique d'approbation de n'importe quel rôle pour le rendre utilisable par les identités du pod EKS.

```
            "Principal": {
                "Service": "pods.eks.amazonaws.com"
            }
```

- Capacité de mise à l'échelle : chaque ensemble d'informations d'identification temporaires est pris en charge par le service d'authentification EKS dans l'identité du pod EKS, au lieu de chaque kit SDK AWS que vous exécutez dans chaque pod. Ensuite, l'agent de l'identité du pod Amazon EKS qui s'exécute sur chaque nœud émet les informations d'identification aux kits SDK. La charge est ainsi réduite à une seule fois pour chaque nœud et n'est pas dupliquée dans chaque pod. Pour plus de détails sur le processus, consultez the section called "Fonctionnement".

Pour plus d'informations sur la comparaison des deux solutions, consultez the section called "Accès des charges de travail à AWS ".

Présentation de la configuration des identités du pod EKS

Activez les identités du pod EKS en effectuant les procédures suivantes :

1. the section called "Configurer l'agent" : vous n'effectuez cette procédure qu'une seule fois pour chaque cluster. Vous n'avez pas besoin d'effectuer cette étape si le mode automatique EKS est activé sur votre cluster.
2. the section called "Attribution d'un rôle IAM" : effectuez cette procédure pour chaque ensemble unique d'autorisations que vous voulez attribuer à une application.

3. the section called "Compte de service de pod" : effectuez cette procédure pour chaque pod qui doit accéder aux services AWS.
4. the section called "Kits SDK pris en charge" : vérifiez que la charge de travail utilise un kit SDK AWS d'une version prise en charge et qu'elle utilise la chaîne d'informations d'identification par défaut.

Limites

- Jusqu'à 5 000 associations d'identité du pod EKS par cluster sont prises en charge pour mapper les rôles IAM aux comptes de service Kubernetes.

Considérations

- Association de rôles IAM : chaque compte de service Kubernetes d'un cluster peut être associé à un rôle IAM du même compte AWS que le cluster. Pour modifier le rôle, modifiez l'association d'identité du pod EKS. Pour l'accès intercompte, déléguez l'accès au rôle à l'aide des rôles IAM. Pour en savoir plus, consultez Déléguer l'accès entre les comptes AWS à l'aide des rôles IAM dans le Guide de l'utilisateur IAM.
- Agent d'identité du pod EKS : l'agent d'identité du pod est nécessaire pour utiliser l'identité du pod EKS. L'agent s'exécute en tant que `DaemonSet` Kubernetes sur les nœuds du cluster, fournissant des informations d'identification uniquement aux pods du même nœud. Il utilise le `hostNetwork` du nœud, occupant les ports `80` et `2703` sur l'adresse locale de liaison (`169.254.170.23` pour IPv4, `[fd00:ec2::23]` pour IPv6). Si IPv6 est désactivé dans votre cluster, désactivez IPv6 pour l'agent d'identité du pod EKS. Pour en savoir plus, consultez Désactiver IPv6 dans l'agent d'identité du pod EKS.
- Cohérence à terme : les associations d'identité du pod EKS sont cohérentes à terme, avec des retards potentiels de plusieurs secondes après les appels d'API. Évitez de créer ou de mettre à jour des associations dans des chemins de code critiques et à haute disponibilité. Effectuez plutôt ces actions dans des routines d'initialisation ou de configuration distinctes et moins fréquentes. Pour en savoir plus, consultez Groupes de sécurité par pod dans le Guide des bonnes pratiques EKS.
- Considérations relatives au proxy et au groupe de sécurité : pour les pods utilisant un proxy, ajoutez `169.254.170.23` (IPv4) et `[fd00:ec2::23]` (IPv6) aux variables d'environnement `no_proxy`/`NO_PROXY` afin d'éviter les échecs de demandes vers l'agent d'identité du pod EKS. Si vous utilisez des groupes de sécurité pour les pods avec le CNI VPC AWS, définissez l'indicateur `ENABLE_POD_ENI` sur « true » et l'indicateur `POD_SECURITY_GROUP_ENFORCING_MODE`

sur « standard ». Pour en savoir plus, consultez Attribuer des groupes de sécurité à des pods individuels.

Versions du cluster de l'identité du pod EKS

Pour utiliser l'identité du pod EKS, le cluster doit disposer d'une version de plateforme identique ou supérieure à celle indiquée dans le tableau suivant, ou d'une version Kubernetes supérieure à celles indiquées dans le tableau. Pour connaître la version recommandée de l'Agent de l'identité du pod Amazon EKS pour une version Kubernetes, consultez the section called "Vérification de la compatibilité".

Version de Kubernetes	Version de la plateforme
Versions Kubernetes non répertoriées	Toutes les versions de plateforme prennent en charge
`1.28`	`eks.4`

Restrictions relatives à l'identité du pod EKS

Les identités du pod EKS sont disponibles sur les éléments suivants :

- Les versions du cluster Amazon EKS répertoriées dans la rubrique précédente the section called "Versions du cluster de l'identité du pod EKS".
- Les composants master du cluster qui sont des instances Amazon EC2 Linux.

Les identités du pod Amazon EKS ne sont pas disponibles dans les cas suivants :

- AWS Outposts.
- Amazon EKS Anywhere.
- Clusters Kubernetes que vous créez et exécutez sur Amazon EC2. Les composants de l'identité du pod Amazon EKS sont uniquement disponibles sur Amazon EKS.

Vous ne pouvez pas utiliser les identités du pod Amazon EKS avec :

- Les pods qui s'exécutent n'importe où, à l'exception des instances Amazon EC2 Linux. Les pods Linux et Windows qui s'exécutent sur AWS Fargate (Fargate) ne sont pas pris en charge. Les pods qui s'exécutent sur des instances Amazon EC2 Windows ne sont pas pris en charge.

Comprendre le fonctionnement de l'identité du pod Amazon EKS

Les associations de l'identité du pod Amazon EKS offrent une capacité de gestion des informations d'identification à utiliser pour les applications, de la même façon que les profils d'instance Amazon EC2 fournissent des informations d'identification aux instances EC2.

L'identité du pod Amazon EKS fournit des informations d'identification à vos charges de travail avec une API d'authentification EKS supplémentaire et un pod d'agent qui s'exécute sur chaque nœud.

Dans vos modules complémentaires, tels que les modules complémentaires Amazon EKS et le contrôleur autogéré, les opérateurs et autres modules complémentaires, l'auteur doit mettre à jour son logiciel pour utiliser les derniers kits SDK AWS. Pour la liste de compatibilité entre l'identité du pod EKS et les modules complémentaires produits par Amazon EKS, reportez-vous à la section précédente the section called "Restrictions relatives à l'identité du pod EKS".

Utilisation des identités du pod EKS dans votre code

Dans votre code, vous pouvez utiliser les kits SDK AWS pour accéder aux services AWS. Vous écrivez du code pour créer un client pour un service AWS avec un kit SDK, et par défaut, le kit SDK recherche dans une chaîne d'emplacements les informations d'identification de la gestion des identités et des accès AWS à utiliser. Une fois les informations d'identification valides trouvées, la recherche s'arrête. Pour plus d'informations sur les emplacements par défaut utilisés, consultez Chaîne de fournisseurs d'informations d'identification dans le Guide de référence des kits SDK et des outils AWS.

Les identités du pod EKS ont été ajoutées au fournisseur d'informations d'identification du conteneur qui est recherché dans une étape de la chaîne d'informations d'identification par défaut. Si vos charges de travail utilisent actuellement des informations d'identification antérieures dans la chaîne de fournisseurs d'informations d'identification, ces informations continueront d'être utilisées même si vous configurez une association d'identité du pod EKS pour la même charge de travail. Ainsi, vous pouvez migrer en toute sécurité à partir d'autres types d'informations d'identification en créant d'abord l'association, avant de supprimer les anciennes informations d'identification.

Le fournisseur d'informations d'identification de conteneur fournit des informations d'identification temporaires à partir d'un agent qui s'exécute sur chaque nœud. Dans Amazon EKS, l'agent est

l'agent d'identité du pod Amazon EKS et sur Amazon Elastic Container Service, l'agent est le `amazon-ecs-agent`. Les kits SDK utilisent des variables d'environnement pour localiser l'agent auquel se connecter.

En revanche, les rôles IAM pour les comptes de service fournissent un jeton d'identité Web que le kit SDK AWS doit échanger avec le service de jetons de sécurité AWS à l'aide d'`AssumeRoleWithWebIdentity`.

Fonctionnement de l'agent de l'identité du pod EKS avec un pod

1. Lorsque Amazon EKS démarre un nouveau pod qui utilise un compte de service associé à une identité du pod EKS, le cluster ajoute le contenu suivant au manifeste du pod :

```
    env:
    - name: AWS_CONTAINER_AUTHORIZATION_TOKEN_FILE
      value: "/var/run/secrets/pods.eks.amazonaws.com/serviceaccount/eks-pod-
identity-token"
    - name: AWS_CONTAINER_CREDENTIALS_FULL_URI
      value: "http://169.254.170.23/v1/credentials"
    volumeMounts:
    - mountPath: "/var/run/secrets/pods.eks.amazonaws.com/serviceaccount/"
      name: eks-pod-identity-token
  volumes:
  - name: eks-pod-identity-token
    projected:
      defaultMode: 420
      sources:
      - serviceAccountToken:
          audience: pods.eks.amazonaws.com
          expirationSeconds: 86400 # 24 hours
          path: eks-pod-identity-token
```

2. Kubernetes sélectionne le nœud sur lequel exécuter le pod. Ensuite, l'agent d'identité du pod Amazon EKS sur le nœud utilise l'action AssumeRoleForPodIdentity pour récupérer des informations d'identification temporaires auprès de l'API d'authentification EKS.
3. L'agent d'identité du pod EKS met ces identités à la disposition des kits SDK AWS que vous exécutez à l'intérieur de vos conteneurs.
4. Vous utilisez le kit SDK dans votre application sans spécifier de fournisseur d'informations d'identification pour utiliser la chaîne d'informations d'identification par défaut. Vous pouvez également spécifier le fournisseur d'informations d'identification du conteneur. Pour plus

d'informations sur les emplacements par défaut utilisés, consultez Chaîne de fournisseurs d'informations d'identification dans le Guide de référence des kits SDK et des outils AWS.

5. Le kit SDK utilise les variables d'environnement pour se connecter à l'agent d'identité du pod EKS et récupérer les informations d'identification.

> **Note**
>
> Si vos charges de travail utilisent actuellement des informations d'identification antérieures dans la chaîne de fournisseurs d'informations d'identification, ces informations continueront d'être utilisées même si vous configurez une association d'identité du pod EKS pour la même charge de travail.

Configurer l'agent de l'identité du pod Amazon EKS

Les associations Amazon EKS Pod Identity permettent de gérer les informations d'identification de vos applications, de la même manière que les profils d' EC2 instance Amazon fournissent des informations d'identification aux EC2 instances Amazon.

L'identité du pod Amazon EKS fournit des informations d'identification à vos charges de travail avec une API d'authentification EKS supplémentaire et un pod d'agent qui s'exécute sur chaque nœud.

> **Tip**
>
> Vous n'avez pas besoin d'installer l'agent de l'identité du pod Amazon EKS sur les clusters du mode automatique EKS. Cette fonctionnalité est créée dans le mode automatique EKS.

Considérations

- Par défaut, l'agent de l'identité du pod Amazon EKS est préinstallé sur les clusters du mode automatique EKS. Pour en savoir plus, veuillez consulter la section Mode automatique EKS.
- Par défaut, l'agent d'identité du pod EKS écoute sur une adresse `IPv4` et `IPv6` pour que les pods puissent demander des informations d'identification. L'agent utilise l'adresse IP de bouclage (localhost) `169.254.170.23` pour `IPv4` et l'adresse IP localhost `[fd00:ec2::23]` pour `IPv6`.
- Si vous désactivez les adresses `IPv6` ou empêchez d'une autre manière les adresses IP `IPv6` localhost, l'agent ne peut pas démarrer. Pour démarrer l'agent sur des nœuds qui ne peuvent

pas utiliser `IPv6`, suivez les étapes décrites dans the section called "Désactivation d'IPv6" pour désactiver la configuration `IPv6`.

Création d'un agent d'identité du pod Amazon EKS

Conditions préalables pour l'agent

- Un cluster Amazon EKS existant. Pour en déployer un, consultez Démarrer. La version du cluster et la version de plateforme doivent être identiques ou supérieures aux versions répertoriées dans Versions du cluster de l'identité du pod EKS.
- Le rôle de nœud dispose d'autorisations permettant à l'agent d'effectuer l'action `AssumeRoleForPodIdentity` dans l'API d'authentification EKS. Vous pouvez utiliser la politique AWS gérée : Amazon EKSWorker NodePolicy ou ajouter une politique personnalisée similaire à la suivante :

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "eks-auth:AssumeRoleForPodIdentity"
            ],
            "Resource": "*"
        }
    ]
}
```

  Cette action peut être limitée par des balises pour restreindre les rôles qui peuvent être assumés par les pods qui utilisent l'agent.
- Les nœuds peuvent atteindre et télécharger des images à partir d'Amazon ECR. L'image de conteneur pour le module complémentaire se trouve dans les registres répertoriés dans Afficher les registres d'images de conteneur Amazon pour les modules complémentaires Amazon EKS.

  Notez que vous pouvez modifier l'emplacement de l'image et `imagePullSecrets` prévoir des modules complémentaires EKS dans les paramètres de configuration facultatifs de et `--configuration-values` dans la AWS CLI. AWS Management Console

- Les nœuds peuvent accéder à l'API d'authentification Amazon EKS. Pour les clusters privés, le `eks-auth` point de terminaison AWS PrivateLink est obligatoire.

Agent d'installation avec AWS console

1. Ouvrez la console Amazon EKS.
2. Dans le panneau de navigation de gauche, sélectionnez Clusters, puis sélectionnez le nom du cluster pour lequel vous souhaitez configurer le module complémentaire l'agent d'identité du pod EKS.
3. Choisissez l'onglet Modules complémentaires.
4. Choisissez Obtenez plus de modules complémentaires.
5. Cochez la case en haut à droite du module complémentaire de l'agent d'identité du pod EKS, puis sélectionnez Suivant.
6. Sur la page Configurer les paramètres de modules complémentaires sélectionnés, sélectionnez n'importe quelle Version dans la liste déroulante Version.
7. (Facultatif) Développez les Paramètres de configuration facultatifs pour entrer une configuration supplémentaire. Par exemple, vous pouvez fournir un autre emplacement d'image de conteneur et `ImagePullSecrets`. Le schéma JSON avec les clés acceptées est présenté dans Schéma de configuration du module complémentaire.

   Saisissez les clés et les valeurs de configuration dans Valeurs de configuration.
8. Choisissez Suivant.
9. Confirmez que les pods de l'agent d'identité du pod EKS sont en cours d'exécution sur votre cluster.

   ```
   kubectl get pods -n kube-system | grep 'eks-pod-identity-agent'
   ```

   L'exemple qui suit illustre un résultat.

   ```
   eks-pod-identity-agent-gmqp7                                          1/1     Running
      1 (24h ago)   24h
   eks-pod-identity-agent-prnsh                                          1/1     Running
      1 (24h ago)   24h
   ```

   Vous pouvez maintenant utiliser les associations d'identité du pod EKS dans votre cluster. Pour de plus amples informations, veuillez consulter the section called "Attribution d'un rôle IAM".

Agent de configuration avec AWS CLI

1. Exécutez la commande AWS CLI suivante. Remplacez `my-cluster` par le nom de votre cluster.

```
aws eks create-addon --cluster-name my-cluster --addon-name eks-pod-identity-agent --
addon-version v1.0.0-eksbuild.1
```

> **Note**
>
> L'agent de l'identité du pod Amazon EKS n'utilise pas le `service-account-role-arn` pour les rôles IAM pour les comptes de service. Vous devez fournir à l'agent d'identité du pod EKS des autorisations dans le rôle de nœud.

2. Confirmez que les pods de l'agent d'identité du pod EKS sont en cours d'exécution sur votre cluster.

```
kubectl get pods -n kube-system | grep 'eks-pod-identity-agent'
```

L'exemple qui suit illustre un résultat.

```
eks-pod-identity-agent-gmqp7                                          1/1     Running
   1 (24h ago)   24h
eks-pod-identity-agent-prnsh                                          1/1     Running
   1 (24h ago)   24h
```

Vous pouvez maintenant utiliser les associations d'identité du pod EKS dans votre cluster. Pour de plus amples informations, veuillez consulter the section called "Attribution d'un rôle IAM".

Attribuer un rôle IAM à un compte de service Kubernetes

Cette rubrique explique comment configurer un compte de service Kubernetes pour qu'il assume un rôle de gestion des AWS identités et des accès (IAM) avec EKS Pod Identity. Tous les pods configurés pour utiliser le compte de service peuvent ensuite accéder à tout AWS service auquel le rôle est autorisé à accéder.

Pour créer une association EKS Pod Identity, il n'y a qu'une seule étape : vous créez l'association dans EKS via la AWS Management Console AWS CLI AWS CloudFormation et d'autres outils. AWS

SDKs Il n'y a aucune donnée ou métadonnée concernant les associations à l'intérieur du cluster dans les objets Kubernetes et vous n'ajoutez aucune annotation aux comptes de service.

Conditions préalables

- Un cluster existant. Si vous n'en avez pas, vous pouvez en créer un en suivant l'un des guides Démarrer.
- Le principal IAM qui crée l'association doit avoir le `iam:PassRole`.
- La dernière version de la AWS CLI installée et configurée sur votre appareil ou AWS CloudShell. Vous pouvez vérifier votre version actuelle avec `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Les gestionnaires de packages tels que `yum` Homebrew pour macOS ont souvent plusieurs versions de retard sur la dernière version de la AWS CLI. `apt-get` Pour installer la dernière version, consultez la section Installation et configuration rapide avec aws configure dans le Guide de l'utilisateur de l'interface de ligne de AWS commande. La version de la AWS CLI installée dans le AWS CloudShell peut également avoir plusieurs versions de retard par rapport à la dernière version. Pour le mettre à jour, consultez la section Installation de la AWS CLI dans votre répertoire de base dans le guide de AWS CloudShell l'utilisateur.
- L'outil de ligne de commande `kubectl` est installé sur votre appareil ou AWS CloudShell. La version peut correspondre à celle utilisée par votre cluster Kubernetes, ou différer d'au plus une version mineure, qu'elle soit antérieure ou plus récente. Par exemple, si la version de votre cluster est `1.29`, vous pouvez utiliser la version `kubectl` `1.28`, `1.29` ou `1.30`. Pour installer ou mettre à niveau `kubectl`, veuillez consulter the section called "Configurer kubectl et eksctl".
- Un fichier existant `kubectl config` qui contient la configuration de votre cluster. Pour créer un fichier `kubectl config`, consultez the section called "Accès au cluster avec kubectl".

Création d'une association Pod Identity (AWS console)

1. Ouvrez la console Amazon EKS.
2. Dans le panneau de navigation de gauche, sélectionnez Clusters, puis sélectionnez le nom du cluster pour lequel vous souhaitez configurer le module complémentaire l'agent d'identité du pod EKS.
3. Choisissez l'onglet Access.
4. Dans Associations d'identité du pod, sélectionnez Créer.
5. Pour Rôle IAM, sélectionnez le rôle IAM avec les autorisations dont a besoin la charge de travail.

> **Note**
>
> La liste ne contient que les rôles qui ont la politique d'approbation suivante qui permet à l'identité du pod EKS de les utiliser.

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

`sts:AssumeRole` : l'identité du pod EKS utilise `AssumeRole` pour assumer le rôle IAM avant de transmettre les informations d'identification temporaires à vos pods.

`sts:TagSession`— EKS Pod Identity est utilisé `TagSession` pour inclure des balises de session dans les demandes adressées à AWS STS.

Vous pouvez utiliser ces balises dans les clés de condition de la politique d'approbation pour restreindre les comptes de service, les espaces de noms et les clusters pouvant utiliser ce rôle.

Pour obtenir la liste des clés de condition Amazon EKS, consultez la rubrique Conditions définies par Amazon Elastic Kubernetes Service de la Référence de l'autorisation de service. Pour savoir avec quelles actions et ressources vous pouvez utiliser une clé de condition, consultez la rubrique Actions définies par Amazon Elastic Kubernetes Service.

6. Pour l'espace de noms Kubernetes, sélectionnez l'espace de noms Kubernetes qui contient le compte de service et la charge de travail. Vous pouvez également spécifier un espace de noms par son nom qui n'existe pas dans le cluster.

7. Pour le compte de service Kubernetes, sélectionnez le compte de service Kubernetes à utiliser. Le manifeste de votre charge de travail Kubernetes doit spécifier ce compte de service. Vous pouvez également spécifier un compte de service par son nom qui n'existe pas dans le cluster.
8. (Facultatif) Sélectionnez Désactiver les balises de session pour désactiver les balises de session par défaut que Pod Identity ajoute automatiquement lorsqu'il assume le rôle.
9. (Facultatif) Activez l'option Configurer la politique de session pour configurer une politique IAM afin d'appliquer des restrictions supplémentaires à cette association Pod Identity au-delà des autorisations définies dans la stratégie IAM attachée au rôle IAM.

> **Note**
>
> Une politique de session ne peut être appliquée que lorsque le paramètre Désactiver les balises de session est coché.

10. (Facultatif) Pour Balises, choisissez Ajouter une balise pour ajouter des métadonnées dans une paire clé et valeur. Ces balises sont appliquées à l'association et peuvent être utilisées dans les politiques IAM.

    Vous pouvez répéter cette étape pour ajouter plusieurs balises.
11. Choisissez Créer.

Création d'une association Pod Identity (AWS CLI)

1. Si vous souhaitez associer une politique IAM existante à votre rôle IAM, passez à l'étape suivante.

   Créez une politique IAM. Vous pouvez créer votre propre politique ou copier une politique AWS gérée qui accorde déjà certaines des autorisations dont vous avez besoin et la personnaliser en fonction de vos besoins spécifiques. Pour plus d'informations, consultez Création de politiques IAM dans le Guide de l'utilisateur IAM.

   a. Créez un fichier qui inclut les autorisations pour les AWS services auxquels vous souhaitez que vos pods accèdent. Pour obtenir la liste de toutes les actions pour tous les AWS services, consultez la référence d'autorisation des services.

      Vous pouvez exécuter la commande suivante pour créer un exemple de fichier de politique autorisant l'accès en lecture seule à un compartiment Amazon S3. Vous pouvez, si vous le souhaitez, stocker des informations de configuration ou un script d'amorçage dans ce compartiment et les conteneurs de votre pod pourront lire le fichier dans le compartiment et

le charger dans votre application. Si vous souhaitez créer cet exemple de politique, copiez le contenu suivant sur votre appareil. *my-pod-secrets-bucket*Remplacez-le par le nom de votre compartiment et exécutez la commande.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::my-pod-secrets-bucket"
        }
    ]
}
```

b. Créez la politique IAM.

```
aws iam create-policy --policy-name my-policy --policy-document file://my-
policy.json
```

2. Créez un rôle IAM, puis associez-le à un compte de service Kubernetes.

a. Si vous avez déjà un compte de service Kubernetes devant assumer un rôle IAM, vous pouvez ignorer cette étape.

Créez un compte de service Kubernetes. Copiez les contenus suivants sur votre appareil. *my-service-account*Remplacez-le par le nom de votre choix et *default* par un autre espace de noms, si nécessaire. Si vous modifiez*default*, l'espace de noms doit déjà exister.

```
cat >my-service-account.yaml <<EOF
apiVersion: v1
kind: ServiceAccount
metadata:
  name: my-service-account
  namespace: default
EOF
kubectl apply -f my-service-account.yaml
```

Exécutez la commande suivante.

```
kubectl apply -f my-service-account.yaml
```

b. Exécutez la commande suivante pour créer un fichier de stratégie d'approbation pour le rôle IAM.

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

c. Créez le rôle. Remplacez *my-role* avec un nom pour votre rôle IAM, et *my-role-description* avec une description de votre rôle.

```
aws iam create-role --role-name my-role --assume-role-policy-document file://
trust-relationship.json --description "my-role-description"
```

d. Liez une politique IAM à votre rôle. Remplacez *my-role* par le nom de votre rôle IAM et *my-policy* par le nom d'une politique existante que vous avez créée.

```
aws iam attach-role-policy --role-name my-role --policy-arn=arn:aws:
 iam::111122223333:policy/my-policy
```

> **Note**
>
> Contrairement aux rôles IAM pour les comptes de service, l'identité du pod EKS n'utilise pas d'annotation sur le compte de service.

e. Exécutez la commande suivante pour créer l'association. `my-cluster`Remplacez-le par le nom du cluster, remplacez-le *my-service-account* par le nom de votre choix et *default* par un autre espace de noms, si nécessaire.

```
aws eks create-pod-identity-association --cluster-name my-cluster --role-arn
 arn:aws: iam::111122223333:role/my-role --namespace default --service-account my-
service-account
```

L'exemple qui suit illustre un résultat.

```
{
    "association": {
        "clusterName": "my-cluster",
        "namespace": "default",
        "serviceAccount": "my-service-account",
        "roleArn": "arn:aws: iam::111122223333:role/my-role",
        "associationArn": "arn:aws::111122223333:podidentityassociation/my-
cluster/a-abcdefghijklmnop1",
        "associationId": "a-abcdefghijklmnop1",
        "tags": {},
        "createdAt": 1700862734.922,
        "modifiedAt": 1700862734.922
    }
}
```

> **Note**
>
> Vous pouvez spécifier un espace de noms et un compte de service par leur nom qui n'existent pas dans le cluster. Vous devez créer l'espace de noms, le compte de service et la charge de travail qui utilise le compte de service pour que l'association de l'identité du pod EKS fonctionne.

Confirmer la configuration

1. Vérifiez que la stratégie d'approbation du rôle IAM est correctement configurée.

```
aws iam get-role --role-name my-role --query Role.AssumeRolePolicyDocument
```

L'exemple qui suit illustre un résultat.

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

```
    "Statement": [
        {
            "Sid": "Allow EKS Auth service to assume this role for Pod Identities",
            "Effect": "Allow",
            "Principal": {
                "Service": "pods.eks.amazonaws.com"
            },
            "Action": [
                "sts:AssumeRole",
                "sts:TagSession"
            ]
        }
    ]
}
```

2. Vérifiez que la stratégie que vous avez associée à votre rôle lors d'une étape précédente est associée au rôle.

```
aws iam list-attached-role-policies --role-name my-role --query
 'AttachedPolicies[].PolicyArn' --output text
```

L'exemple qui suit illustre un résultat.

```
 arn:aws: iam::111122223333:policy/my-policy
```

3. Définissez une variable pour stocker l'Amazon Resource Name (ARN) de la stratégie que vous souhaitez utiliser. *my-policy*Remplacez-le par le nom de la politique pour laquelle vous souhaitez confirmer les autorisations.

```
export policy_arn=arn:aws: iam::111122223333:policy/my-policy
```

4. Affichez la version par défaut de la stratégie.

```
aws iam get-policy --policy-arn $policy_arn
```

L'exemple qui suit illustre un résultat.

```
{
    "Policy": {
        "PolicyName": "my-policy",
        "PolicyId": "EXAMPLEBIOWGLDEXAMPLE",
```

```
        "Arn": "arn:aws: iam::111122223333:policy/my-policy",
        "Path": "/",
        "DefaultVersionId": "v1",
        [...]
    }
}
```

5. Consultez le contenu de la stratégie pour vous assurer qu'elle inclut toutes les autorisations nécessaires pour votre pod. Si nécessaire, remplacez *1* la commande suivante par la version renvoyée dans la sortie précédente.

```
aws iam get-policy-version --policy-arn $policy_arn --version-id v1
```

L'exemple qui suit illustre un résultat.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::my-pod-secrets-bucket"
        }
    ]
}
```

Si vous avez créé l'exemple de stratégie lors d'une étape précédente, le résultat est le même. Si vous avez créé une politique différente, le *example* contenu est différent.

Étapes suivantes

the section called "Compte de service de pod"

Accédez aux AWS ressources à l'aide des rôles IAM EKS Pod Identity Target

Lorsque vous exécutez des applications sur Amazon Elastic Kubernetes Service (Amazon EKS), il se peut que vous deviez accéder AWS à des ressources qui existent dans différents comptes. AWS Ce guide explique comment configurer l'accès entre comptes à l'aide d'EKS Pod Identity, qui permet à vos pods Kubernetes d'accéder à d'autres AWS ressources à l'aide de rôles cibles.

Conditions préalables

Avant de commencer, assurez-vous d'avoir effectué les étapes suivantes :

- Configurer l'agent de l'identité du pod Amazon EKS
- Créer un rôle identité du pod EKS

Comment ça marche

Pod Identity permet aux applications de votre cluster EKS d'accéder aux AWS ressources de différents comptes par le biais d'un processus appelé chaînage des rôles.

Lorsque vous créez une association Pod Identity, vous pouvez fournir deux rôles IAM : un rôle EKS Pod Identity dans le même compte que votre cluster EKS et un rôle IAM cible provenant du compte contenant les AWS ressources auxquelles vous souhaitez accéder (comme les compartiments S3 ou les bases de données RDS). Le rôle EKS Pod Identity doit figurer dans le compte de votre cluster EKS en raison des PassRole exigences IAM, tandis que le rôle IAM cible peut se trouver dans n'importe quel AWS compte. PassRole permet à une AWS entité de déléguer l'attribution de rôles à un autre service. EKS Pod Identity PassRole permet de connecter un rôle à un compte de service Kubernetes, ce qui nécessite que le rôle et l'identité qui le transmettent se trouvent dans le même AWS compte que le cluster EKS. Lorsque votre pod d'application a besoin d'accéder à AWS des ressources, il demande des informations d'identification à Pod Identity. L'identité du pod effectue alors automatiquement deux prises de rôle consécutives : elle prend d'abord le rôle d'identité du pod EKS, puis utilise ces informations d'identification pour prendre le rôle IAM cible. Ce processus fournit à votre pod des informations d'identification temporaires dotées des autorisations définies dans le rôle cible, ce qui permet un accès sécurisé aux ressources d'autres AWS comptes.

Considérations relatives à la mise en cache

En raison des mécanismes de mise en cache, les mises à jour d'un rôle IAM dans une association identité du pod EKS peuvent ne pas prendre effet immédiatement dans les pods s'exécutant sur votre cluster EKS. L'agent identité du pod met en cache les informations d'identification IAM en fonction de la configuration de l'association au moment où les informations d'identification sont récupérées. Si l'association comprend uniquement un rôle d'identité du pod EKS et aucun rôle IAM cible, les informations d'identification mises en cache sont valables pendant 6 heurs. Si l'association comprend à la fois l'ARN du rôle d'identité du pod EKS et un rôle IAM cible, les informations d'identification mises en cache sont valables pendant 59 minutes. La modification d'une association existante, telle que la mise à jour de l'ARN du rôle d'identité du pod EKS ou l'ajout d'un rôle IAM cible, ne réinitialise pas le cache existant. Par conséquent, l'agent ne reconnaîtra pas les mises à jour tant

que les informations d'identification mises en cache n'auront pas été actualisées. Pour appliquer les modifications plus rapidement, vous pouvez recréer les pods existants ; sinon, vous devrez attendre l'expiration du cache.

Étape 1 : créer et associer un rôle IAM cible

Au cours de cette étape, vous allez établir une chaîne de confiance sécurisée en créant et en configurant un rôle IAM cible. À titre de démonstration, nous allons créer un nouveau rôle IAM cible pour établir une chaîne de confiance entre deux AWS comptes : le rôle EKS Pod Identity (par exemple`eks-pod-identity-primary-role`) du AWS compte du cluster EKS est autorisé à assumer le rôle IAM cible (par exemple`eks-pod-identity-aws-resources`) sur votre compte cible, permettant ainsi d'accéder à des AWS ressources telles que les compartiments Amazon S3.

Créer le rôle IAM cible

1. Ouvrez la console Amazon IAM.
2. Dans la barre de navigation supérieure, vérifiez que vous êtes connecté au compte contenant les AWS ressources (telles que des compartiments S3 ou des tables DynamoDB) pour votre rôle IAM cible.
3. Dans le volet de navigation de gauche, choisissez Rôles.
4. Cliquez sur le bouton Créer un rôle, puis sur le AWS compte sous « Type d'entité fiable ».
5. Choisissez Un autre AWS compte, entrez votre numéro de AWS compte (le compte sur lequel se trouve votre rôle EKS Pod Identity), puis choisissez Next.
6. Ajoutez les politiques d'autorisation que vous souhaitez associer au rôle (par exemple, AmazonS3FullAccess), puis choisissez Next.
7. Saisissez un nom de rôle, tel que `MyCustomIAMTargetRole`, puis sélectionnez Créer un rôle.

Mettre à jour la politique d'approbation du rôle IAM cible

1. Après avoir créé le rôle, vous serez redirigé vers la liste Rôles. Recherchez et sélectionnez le nouveau rôle que vous avez créé à l'étape précédente (par exemple, `MyCustomIAMTargetRole`).
2. Sélectionnez l'onglet Relations d'approbation.
3. Cliquez sur Modifier la politique d'approbation à droite.
4. Dans l'éditeur de politique, remplacez le JSON par défaut par votre politique d'approbation. Remplacez les valeurs d'espace réservé pour le nom du rôle et `111122223333` dans l'ARN

du rôle IAM par l'ID du AWS compte hébergeant votre cluster EKS. Vous pouvez également éventuellement utiliser la politique de confiance PrincipalTags dans les rôles pour autoriser uniquement des comptes de service spécifiques d'un cluster et d'un espace de noms donnés à assumer votre rôle cible. Par exemple :

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:root"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ],
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/eks-cluster-arn": "arn:aws:eks:us-
east-1:111122223333:cluster/example-cluster",
          "aws:RequestTag/kubernetes-namespace": "ExampleNameSpace",
          "aws:RequestTag/kubernetes-service-account": "ExampleServiceAccountName"
        },
        "ArnEquals": {
          "aws:PrincipalARN": "arn:aws:iam::111122223333:role/eks-pod-identity-primary-
role"
        }
      }
    }
  ]
}
```

La politique ci-dessus permet au rôle `eks-pod-identeity-primary-role` du AWS compte 111122223333 doté des tags de session EKS Pod Identity appropriés d'assumer ce rôle.

Si vous avez désactivé les balises de session dans votre identité du pod EKS, l'identité du pod EKS définit également le `sts:ExternalId` avec des informations sur le cluster, l'espace de noms et le compte de service d'un pod lors de l'assomption d'un rôle cible.

```
{
```

```
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:root"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "sts:ExternalId": "region/111122223333/cluster-name/namespace/service-
account-name"
        },
        "ArnEquals": {
          "aws:PrincipalARN": "arn:aws:iam::111122223333:role/eks-pod-identity-primary-
role"
        }
      }
    },
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:root"
      },
      "Action": "sts:TagSession"
    }
  ]
}
```

La politique ci-dessus permet de garantir que seuls le cluster, l'espace de noms et le compte de service attendus peuvent assumer le rôle cible.

Mettre à jour la politique d'autorisation pour le rôle d'identité du pod EKS

Au cours de cette étape, vous allez mettre à jour la politique d'autorisation du rôle d'identité du pod EKS associé à votre cluster Amazon EKS en ajoutant l'ARN du rôle IAM cible en tant que ressource.

1. Ouvrez la console Amazon EKS.
2. Dans le volet de navigation gauche, sélectionnez Clusters, puis sélectionnez le nom de votre cluster EKS.
3. Choisissez l'onglet Access.
4. Sous Associations d'identité du pod, sélectionnez votre rôle d'identité du pod EKS.

5. Sélectionnez Autorisations, Ajouter des autorisations, puis Créer une politique en ligne.
6. Sélectionnez JSON sur le côté droit.
7. Dans l'éditeur de politique, remplacez le JSON par défaut par votre politique d'autorisation. Remplacez la valeur de remplacement pour le nom du rôle et `222233334444` dans l'ARN du rôle IAM par votre rôle IAM cible. Par exemple :

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sts:AssumeRole",
                "sts:TagSession"
            ],
            "Resource": "arn:aws:iam::222233334444:role/eks-pod-identity-aws-resources"
        }
    ]
}
```

Étape 2 : associer le rôle IAM cible à un compte de service Kubernetes

Au cours de cette étape, vous allez créer une association entre le rôle IAM cible et le compte de service Kubernetes dans votre cluster EKS.

1. Ouvrez la console Amazon EKS.
2. Dans le panneau de navigation gauche, sélectionnez Clusters, puis le nom du cluster auquel vous voulez ajouter l'association.
3. Choisissez l'onglet Access.
4. Dans Associations d'identité du pod, sélectionnez Créer.
5. Sélectionnez le Rôle d'identité du pod EKS dans Rôle IAM pour vos charges de travail.
6. Sélectionnez le rôle IAM cible dans Rôle IAM cible qui sera assumé par le Rôle d'identité du pod EKS.
7. Dans le champ Espace de noms Kubernetes, saisissez le nom de l'espace de noms dans lequel vous voulez créer l'association (par exemple, `my-app-namespace`). Cela permet de définir l'emplacement du compte de service.

8. Dans le champ Compte de service Kubernetes, saisissez le nom du compte de service (par exemple, `my-service-account`) qui utilisera les informations d'identification IAM. Cela permet de lier le rôle IAM au compte de service.
9. (Facultatif) Sélectionnez Désactiver les balises de session pour désactiver les balises de session par défaut que Pod Identity ajoute automatiquement lorsqu'il assume le rôle.
10. (Facultatif) Activez l'option Configurer la politique de session pour configurer une politique IAM afin d'appliquer des restrictions supplémentaires à cette association Pod Identity au-delà des autorisations définies dans la stratégie IAM attachée au rôle IAM cible.

> **Note**
>
> 1. Une politique de session ne peut être appliquée que lorsque le paramètre Désactiver les balises de session est coché. 2. Si vous spécifiez une politique de session, les restrictions de politique s'appliquent aux autorisations du rôle Target IAM et non au rôle IAM associé à cette association Pod Identity.

11. Sélectionnez Créer pour créer l'association.

Configurer les pods pour accéder aux AWS services avec des comptes de service

Si un Pod doit accéder à AWS des services, vous devez le configurer pour utiliser un compte de service Kubernetes. Le compte de service doit être associé à un rôle AWS Identity and Access Management (IAM) autorisé à accéder aux services. AWS

- Un cluster existant. Si vous n'en avez pas, vous pouvez en créer un à l'aide de l'un des guides disponibles dans Démarrer.
- Un compte de service Kubernetes existant et une association d'identité du pod EKS qui associe le compte de service à un rôle IAM. Le rôle doit être associé à une politique IAM contenant les autorisations dont vous souhaitez que vos pods disposent pour utiliser les AWS services. Pour plus d'informations sur la façon de créer et de configurer le compte de service et le rôle, consultez the section called "Attribution d'un rôle IAM".
- La dernière version de la AWS CLI installée et configurée sur votre appareil ou AWS CloudShell. Vous pouvez vérifier votre version actuelle avec `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Les gestionnaires de packages tels que `yum` Homebrew pour macOS ont souvent plusieurs versions de retard sur la dernière version de la AWS CLI. `apt-get` Pour installer la dernière version, consultez la section Installation et configuration rapide avec aws configure dans

le Guide de l'utilisateur de l'interface de ligne de AWS commande. La version de la AWS CLI installée dans le AWS CloudShell peut également avoir plusieurs versions de retard par rapport à la dernière version. Pour le mettre à jour, consultez la section Installation de la AWS CLI dans votre répertoire de base dans le guide de AWS CloudShell l'utilisateur.

- L'outil de ligne de commande `kubectl` est installé sur votre appareil ou AWS CloudShell. La version peut correspondre à celle utilisée par votre cluster Kubernetes, ou différer d'au plus une version mineure, qu'elle soit antérieure ou plus récente. Par exemple, si la version de votre cluster est `1.29`, vous pouvez utiliser la version `kubectl` `1.28`, `1.29` ou `1.30`. Pour installer ou mettre à niveau `kubectl`, veuillez consulter the section called "Configurer kubectl et eksctl".
- Un fichier existant `kubectl config` qui contient la configuration de votre cluster. Pour créer un fichier `kubectl config`, consultez the section called "Accès au cluster avec kubectl".

1. Utilisez la commande suivante pour créer un manifeste de déploiement que vous pouvez déployer sur un pod afin de confirmer la configuration. Remplacez les exemples de valeurs par vos propres valeurs.

```
cat >my-deployment.yaml <<EOF
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-app
spec:
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
    spec:
      serviceAccountName: my-service-account
      containers:
      - name: my-app
        image: public.ecr.aws/nginx/nginx:X.XX
EOF
```

2. Appliquez le manifeste à votre cluster.

```
kubectl apply -f my-deployment.yaml
```

3. Vérifiez que les variables d'environnement requises existent pour votre pod.

a. Affichez les pods qui ont été déployés avec le déploiement à l'étape précédente.

```
kubectl get pods | grep my-app
```

L'exemple qui suit illustre un résultat.

```
my-app-6f4dfff6cb-76cv9   1/1     Running   0          3m28s
```

b. Vérifiez que le pod dispose d'un fichier de jeton de compte de service monté.

```
kubectl describe pod my-app-6f4dfff6cb-76cv9 | grep
 AWS_CONTAINER_AUTHORIZATION_TOKEN_FILE:
```

L'exemple qui suit illustre un résultat.

```
AWS_CONTAINER_AUTHORIZATION_TOKEN_FILE:  /var/run/secrets/
pods.eks.amazonaws.com/serviceaccount/eks-pod-identity-token
```

4. Vérifiez que vos pods peuvent interagir avec les AWS services à l'aide des autorisations que vous avez attribuées dans la politique IAM associée à votre rôle.

> **Note**
>
> Lorsqu'un pod utilise les AWS informations d'identification d'un rôle IAM associé à un compte de service, la AWS CLI ou un autre SDKs élément des conteneurs de ce pod utilisent les informations d'identification fournies par ce rôle. Si vous ne restreignez pas l'accès aux informations d'identification fournies au rôle IAM du nœud Amazon EKS, le pod a toujours accès à ces informations d'identification. Pour plus d'informations, consultez Restreindre l'accès au profil d'instance affecté au composant master.

Si vos pods ne peuvent pas interagir avec les services comme prévu, suivez les étapes suivantes pour vérifier que tout est correctement configuré.

a. Vérifiez que vos pods utilisent une version du AWS SDK qui permet d'assumer un rôle IAM via une association EKS Pod Identity. Pour de plus amples informations, veuillez consulter the section called "Kits SDK pris en charge".

b. Vérifiez que le déploiement utilise le compte de service.

```
kubectl describe deployment my-app | grep "Service Account"
```

L'exemple qui suit illustre un résultat.

```
Service Account:  my-service-account
```

Accordez aux Pods l'accès aux AWS ressources en fonction des balises

Le contrôle d'accès par attributs (ABAC) accorde des droits aux utilisateurs grâce à des politiques qui combinent des attributs. L'identité du pod EKS associe des balises aux informations d'identification temporaires de chaque pod avec des attributs tels que le nom du cluster, l'espace de noms et le nom du compte de service. Ces balises de session de rôle permettent aux administrateurs de créer un rôle unique qui peut fonctionner avec tous les comptes de service en autorisant l'accès aux AWS ressources en fonction des balises correspondantes. En ajoutant la prise en charge des balises de session de rôle, vous pouvez renforcer la sécurité entre les clusters et les charges de travail au sein des clusters, tout en réutilisant les mêmes rôles et politiques IAM.

Exemple de politique avec balises

Vous trouverez ci-dessous un exemple de politique IAM qui accorde les autorisations `s3:GetObject` lorsque l'objet correspondant est balisé avec le nom du cluster EKS.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectTagging"
            ],
            "Resource": "*",
```

```
            "Condition": {
                "StringEquals": {
                    "s3:ExistingObjectTag/eks-cluster-name": "${aws:PrincipalTag/eks-
cluster-name}"
                }
            }
        }
    ]
}
```

Activer ou désactiver les balises de session

Lorsqu'il assume le rôle, l'identité du pod EKS ajoute un ensemble prédéfini de balises de session. Ces balises de session permettent aux administrateurs de créer un rôle unique qui peut fonctionner avec toutes les ressources en autorisant l'accès aux AWS ressources en fonction des balises correspondantes.

Activer les balises de session

Les balises de session sont automatiquement activées avec l'identité du pod EKS. Aucune action n'est requise de votre part. Par défaut, l'identité du pod EKS associe un ensemble de balises prédéfinies à votre session. Pour référencer ces balises dans les politiques, utilisez la syntaxe `${aws:PrincipalTag/` suivie de la clé de balise. Par exemple, `${aws:PrincipalTag/kubernetes-namespace}`.

- `eks-cluster-arn`
- `eks-cluster-name`
- `kubernetes-namespace`
- `kubernetes-service-account`
- `kubernetes-pod-name`
- `kubernetes-pod-uid`

Désactivation des balises de session

AWS compresse les politiques de session intégrées, les politiques ARNs gérées et les balises de session dans un format binaire compressé doté d'une limite distincte. Si vous recevez une erreur `PackedPolicyTooLarge` indiquant que le format binaire compressé a dépassé la limite de taille, vous pouvez essayer de réduire la taille en désactivant les balises de session ajoutées par l'identité du pod EKS. Pour désactiver ces balises de session, veuillez suivre les étapes suivantes :

1. Ouvrez la console Amazon EKS.
2. Dans le panneau de navigation gauche, sélectionnez Clusters, puis sélectionnez le nom du cluster que vous voulez modifier.
3. Choisissez l'onglet Access.
4. Dans Associations d'identité du pod, choisissez l'ID d'association que vous souhaitez modifier dans ID d'association, puis choisissez Modifier.
5. Sous Balises de session, choisissez Désactiver les balises de session.
6. Sélectionnez Enregistrer les modifications.

Balises inter-comptes

Toutes les balises de session ajoutées par l'identité du pod EKS sont transitives ; les clés et les valeurs des balises sont transmises à toutes les actions `AssumeRole` que vos charges de travail utilisent pour changer de rôle dans un autre compte. Vous pouvez utiliser ces balises dans les politiques d'autres comptes pour limiter l'accès dans des scénarios intercomptes. Pour plus d'informations, consultez Chaînage des rôles avec des balises de session dans le Guide de l'utilisateur IAM.

Balises personnalisées

L'identité du pod EKS ne peut pas ajouter de balises personnalisées supplémentaires à l'action `AssumeRole` qu'il effectue. Cependant, les balises que vous appliquez au rôle `${aws:PrincipalTag/` IAM sont toujours disponibles dans le même format : `${aws:PrincipalTag/MyCustomTag}` suivi de la clé, par exemple .

> **Note**
>
> Les balises ajoutées à la session via la demande `sts:AssumeRole` sont prioritaires en cas de conflit. Par exemple, supposons que :
>
> - Amazon EKS ajoute une clé `eks-cluster-name` et une valeur `my-cluster` à la session lorsque EKS assume le rôle client et
> - Vous ajoutez une balise `eks-cluster-name` au rôle IAM avec la valeur `my-own-cluster`.

Dans ce cas, la première a priorité et la valeur de la balise `eks-cluster-name` sera `my-cluster`.

Utiliser l'identité du pod avec le SDK AWS

Utilisation des informations d'identification de l'identité du pod EKS

Pour utiliser les informations d'identification d'une association d'identité du pod EKS, votre code peut utiliser n'importe quel kit SDK AWS pour créer un client pour un service AWS avec un kit SDK. Par défaut, le kit SDK recherche dans une chaîne d'emplacements les informations d'identification de la gestion des identités et des accès AWS à utiliser. Les informations d'identification d'identité du pod EKS seront utilisées si vous ne spécifiez pas de fournisseur d'informations d'identification lors de la création du client ou de l'initialisation du SDK.

Cela fonctionne parce que les identités du pod EKS ont été ajoutées au fournisseur d'informations d'identification du conteneur qui est recherché dans une étape de la chaîne d'informations d'identification par défaut. Si vos charges de travail utilisent actuellement des informations d'identification antérieures dans la chaîne de fournisseurs d'informations d'identification, ces informations continueront d'être utilisées même si vous configurez une association d'identité du pod EKS pour la même charge de travail.

Pour plus d'informations sur le fonctionnement des identités du pod EKS, consultez the section called "Fonctionnement".

Lorsque vous utilisez Découvrez comment l'identité du pod EKS accorde aux pods l'accès aux services AWS, les conteneurs de vos pods doivent utiliser une version AWS du kit SDK qui prend en charge l'assumption d'un rôle IAM à partir de l'agent d'identité du pod EKS. Veillez à utiliser les versions suivantes, ou ultérieures, pour votre kit SDK AWS :

- Java (version 2) : 2.21.30
- Java : 1.12.746
- Go v1 : v1.47.11
- Go v2 : release-2023-11-14
- Python (Boto3) : 1.34.41
- Python (botocore) : 1.34.41

- AWS CLI : 1.30.0

  AWS CLI : 2.15.0
- JavaScript v2 : 2.1550.0
- JavaScript v3 : v3.458.0
- Kotlin : v1.0.1
- Ruby : 3.188.0
- Rust : release-2024-03-13
- C++ : 1.11.263
- .NET : 3.7.734.0
- PowerShell : 4.1.502
- PHP : 3.289.0

Pour vous assurer d'utiliser un kit SDK pris en charge, suivez les instructions d'installation relatives à votre SDK préféré de la page Tools to Build on AWS lorsque vous créez vos conteneurs.

Pour obtenir la liste des modules complémentaires qui prennent en charge l'identité du pod EKS, consultez the section called "Référence relative à la prise en charge de l'identité du pod".

Désactiver **`IPv6`** dans l'agent d'identité du pod EKS

AWS Management Console

1. Pour désactiver `IPv6` dans l'agent d'identité du pod EKS, ajoutez la configuration suivante aux Paramètres de configuration facultatifs du module complémentaire EKS.
   a. Ouvrez la console Amazon EKS.
   b. Dans le panneau de navigation de gauche, sélectionnez Clusters, puis sélectionnez le nom du cluster pour lequel vous souhaitez configurer le module complémentaire.
   c. Choisissez l'onglet Modules complémentaires.
   d. Cochez la case en haut à droite de la case du module complémentaire identité du pod EKS, puis sélectionnez Modifier.
   e. Sur la page Configurer l'agent d'identité du pod EKS :
      i. Sélectionnez la version que vous souhaitez utiliser. Il est recommandé de conserver la même version que celle de l'étape précédente et d'effectuer la mise à jour de la version et de la configuration séparément.

ii. Sélectionnez Paramètres de configuration facultatifs.

iii. Saisissez la clé JSON `"agent":` et la valeur d'un objet JSON imbriqué avec une clé `"additionalArgs":` dans Valeurs de configuration. Le texte obtenu doit être un objet JSON valide. Si cette clé et cette valeur sont les seules données de la zone de texte, entourez-les d'accolades `{ }`. L'exemple suivant montre que la politique réseau est activée :

```
{
    "agent": {
        "additionalArgs": {
            "-b": "169.254.170.23"
        }
    }
}
```

Cette configuration définit l'adresse `IPv4` comme seule adresse utilisée par l'agent.

f. Pour appliquer la nouvelle configuration en remplaçant les pods de l'agent d'identité du pod EKS, sélectionnez Enregistrer les modifications.

Amazon EKS applique les modifications aux modules complémentaires Amazon EKS à l'aide d'un déploiement du `DaemonSet` Kubernetes pour l'agent d'identité du pod EKS. Vous pouvez suivre l'état du déploiement dans l'Historique des mises à jour du module complémentaire, dans AWS Management Console, ainsi qu'à l'aide de `kubectl rollout status daemonset/eks-pod-identity-agent --namespace kube-system`.

L'outil `kubectl rollout` propose les commandes suivantes :

```
$ kubectl rollout

history  -- View rollout history
pause    -- Mark the provided resource as paused
restart  -- Restart a resource
resume   -- Resume a paused resource
status   -- Show the status of the rollout
undo     -- Undo a previous rollout
```

Si le déploiement prend trop de temps, Amazon EKS annule automatiquement le déploiement, et un message indiquant un type de Mise à jour du module complémentaire avec un état Échec est ajouté à l'Historique des mises à jour du module complémentaire. Pour examiner tout

problème, commencez par l'historique du déploiement, puis journalisez `kubectl logs` sur un pod de l'agent d'identité du pod EKS pour afficher les journaux de l'agent d'identité du pod EKS.

2. Si la nouvelle entrée dans Historique des mises à jour a le statut Réussi, cela signifie que le déploiement est terminé et que le module complémentaire utilise la nouvelle configuration dans tous les pods de l'agent d'identité du pod EKS.

AWS CLI

1. Pour désactiver `IPv6` dans l'agent d'identité du pod EKS, ajoutez la configuration suivante aux valeurs de configuration du module complémentaire EKS.

   Exécutez la commande AWS CLI suivante. Remplacez `my-cluster` par le nom de votre cluster et l'ARN du rôle IAM par le rôle que vous utilisez.

   ```
   aws eks update-addon --cluster-name my-cluster --addon-name eks-pod-identity-agent \
       --resolve-conflicts PRESERVE --configuration-values '{"agent":{"additionalArgs":
    { "-b": "169.254.170.23"}}}'
   ```

   Cette configuration définit l'adresse `IPv4` comme seule adresse utilisée par l'agent.

   Amazon EKS applique les modifications aux modules complémentaires EKS à l'aide d'un déploiement du DaemonSet Kubernetes pour l'Agent d'identité du pod EKS. Vous pouvez suivre l'état du déploiement dans l'Historique des mises à jour du module complémentaire, dans AWS Management Console, ainsi qu'à l'aide de `kubectl rollout status daemonset/eks-pod-identity-agent --namespace kube-system`.

   L'outil `kubectl rollout` propose les commandes suivantes :

   ```
   kubectl rollout

   history  -- View rollout history
   pause    -- Mark the provided resource as paused
   restart  -- Restart a resource
   resume   -- Resume a paused resource
   status   -- Show the status of the rollout
   undo     -- Undo a previous rollout
   ```

   Si le déploiement prend trop de temps, Amazon EKS annule automatiquement le déploiement, et un message indiquant un type de Mise à jour du module complémentaire avec un état Échec est

ajouté à l'Historique des mises à jour du module complémentaire. Pour examiner tout problème, commencez par l'historique du déploiement, puis journalisez `kubectl logs` sur un pod de l'agent d'identité du pod EKS pour afficher les journaux de l'agent d'identité du pod EKS.

Créer un rôle IAM avec la politique d'approbation requise par l'identité du pod EKS

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

`sts:AssumeRole`

L'identité du pod EKS utilise `AssumeRole` pour assumer le rôle IAM avant de transmettre les informations d'identification temporaires à vos pods.

`sts:TagSession`

EKS Pod `TagSession` Identity inclut des balises de session dans les demandes adressées à AWS STS.

Conditions de configuration

Vous pouvez utiliser ces balises dans les clés de condition de la politique d'approbation pour restreindre les comptes de service, les espaces de noms et les clusters pouvant utiliser ce rôle. Pour obtenir la liste des balises de requête ajoutées par l'identité du pod, consultez the section called "Activer ou désactiver les balises de session".

Par exemple, vous pouvez limiter les pods pouvant assumer le rôle IAM d'identité du pod à un `ServiceAccount` et un `Namespace` spécifiques à l'aide de la politique d'approbation suivante avec la `Condition` ajoutée :

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowEksAuthToAssumeRoleForPodIdentity",
            "Effect": "Allow",
            "Principal": {
                "Service": "pods.eks.amazonaws.com"
            },
            "Action": [
                "sts:AssumeRole",
                "sts:TagSession"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/kubernetes-namespace": [
                        "Namespace"
                    ],
                    "aws:RequestTag/kubernetes-service-account": [
                        "ServiceAccount"
                    ]
                }
            }
        }
    ]
}
```

Pour obtenir la liste des clés de condition Amazon EKS, consultez la rubrique Conditions définies par Amazon Elastic Kubernetes Service de la Référence de l'autorisation de service. Pour savoir avec quelles actions et ressources vous pouvez utiliser une clé de condition, consultez la rubrique Actions définies par Amazon Elastic Kubernetes Service.

Gérer les ressources de calcul à l'aide de nœuds

Un nœud Kubernetes est une machine qui exécute des applications conteneurisées. Chaque nœud dispose des composants suivants :

- Exécution du conteneur : logiciel chargé d'exécuter les conteneurs.
- kubelet : s'assure que les conteneurs sont en bon état et fonctionnent correctement dans leur pod associé.
- kube-proxy : gère les règles réseau qui permettent la communication avec vos pods.

Pour plus d'informations, consultez Nœuds dans la documentation Kubernetes.

Votre cluster Amazon EKS peut planifier des pods sur n'importe quelle combinaison de nœuds gérés du mode automatique EKS, de nœuds autogérés, de groupes de nœuds gérés par Amazon EKS, d'AWS Fargate et de nœuds hybrides Amazon EKS. Pour en savoir plus sur les nœuds déployés dans votre cluster, consultez the section called "Accéder aux ressources du cluster".

> **Note**
>
> À l'exception des nœuds hybrides, les nœuds doivent se trouver dans le même VPC que les sous-réseaux que vous avez sélectionnés lors de la création du cluster. Cependant, les nœuds ne doivent pas nécessairement se trouver dans les mêmes sous-réseaux.

Comparaison des options de calcul

Le tableau suivant fournit plusieurs critères à évaluer pour décider quelles options répondent le mieux à vos besoins. Les nœuds autogérés constituent une autre option qui répond à tous les critères énumérés, mais ils nécessitent beaucoup plus de maintenance manuelle. Pour de plus amples informations, consultez the section called "Nœuds autogérés".

> **Note**
>
> Bottlerocket présente quelques différences spécifiques par rapport aux informations générales de ce tableau. Pour plus d'informations, consultez la documentation de Bottlerocket sur GitHub.

Critères	Groupes de nœuds gérés EKS	Mode automatique EKS	Nœuds hybrides Amazon EKS
Peut être déployé dans AWS Outposts	Non	Non	Non
Possibilité d'être déployé dans une zone locale AWS	Oui	Non	Non
Possibilité d'exécuter des conteneurs qui nécessitent Windows	Oui	Non	Non
Possibilité d'exécuter des conteneurs qui nécessitent Linux	Oui	Oui	Oui
Possibilité d'exécuter des applications nécessitant la puce Inferentia	Oui : nœuds Amazon Linux uniquement	Oui	Non
Possibilité d'exécuter des applications nécessitant un GPU	Oui : nœuds Amazon Linux uniquement	Oui	Oui
Possibilité d'exécuter des applications nécessitant des processeurs Arm	Oui	Oui	Oui
Possibilité d'exécuter AWS Bottlerocket	Oui	Oui	Non
Les pods partagent le processeur, la mémoire, le stockage et les ressources	Oui	Oui	Oui

Critères	Groupes de nœuds gérés EKS	Mode automatique EKS	Nœuds hybrides Amazon EKS
réseau avec d'autres pods.			
Doit déployer et gérer des instances Amazon EC2	Oui	Non : En savoir plus sur les instances gérées EC2	Oui : les machines physiques ou virtuelles sur site sont gérées par vos soins à l'aide des outils de votre choix.
Doit sécuriser, entretenir et corriger le système d'exploitation des instances Amazon EC2	Oui	Non	Oui : vous gérez vous-même le système d'exploitation qui fonctionne sur vos machines physiques ou virtuelles à l'aide des outils de votre choix.
Peut fournir des arguments bootstrap lors du déploiement d'un nœud, tels que des arguments kubelet supplémentaires.	Oui : utilisation de `eksctl` ou d'un modèle de lancement avec une AMI personnalisée.	Non : Utilisez un `NodeClass` pour configurer les nœuds	Oui : Vous pouvez personnaliser les arguments de démarrage avec nodeadm. Consultez the section called "Nœuds hybrides nodeadm".

Critères	Groupes de nœuds gérés EKS	Mode automatique EKS	Nœuds hybrides Amazon EKS
Possibilité d'affecter des adresses IP à des pods à partir d'un bloc d'adresse CIDR différent de l'adresse IP affectée au nœud.	Oui : utilisation d'un modèle de lancement avec une AMI personnalisée. Pour de plus amples informations, consultez the section called "Modèles de lancement".	Non	Oui : consultez the section called "Configurer CNI".
Possibilité d'accéder au nœud via SSH	Oui	Non : Découvrez comment résoudre les problèmes liés aux nœuds	Oui
Possibilité de déployer votre propre AMI personnalisée sur les nœuds	Oui : utilisation d'un modèle de lancement	Non	Oui
Possibilité de déployer votre propre CNI personnalisée sur les nœuds	Oui : utilisation d'un modèle de lancement avec une AMI personnalisée	Non	Oui

Critères	Groupes de nœuds gérés EKS	Mode automatique EKS	Nœuds hybrides Amazon EKS
Vous devez mettre à jour l'AMI du nœud par vous-même	Oui : si vous avez déployé une AMI optimisée pour Amazon EKS, vous êtes averti dans la console Amazon EKS lorsque les mises à jour sont disponibles. Vous pouvez effectuer la mise à jour en un clic dans la console. Si vous avez déployé une AMI personnal isée, vous n'êtes pas averti dans la console Amazon EKS lorsque des mises à jour sont disponibles. Vous devez effectuer la mise à jour par vous-même.	Non	Oui, le système d'exploitation qui fonctionne sur vos machines physiques ou virtuelles est géré par vous-même à l'aide des outils de votre choix. Consultez the section called "Préparer le système d'exploitation".

Critères	Groupes de nœuds gérés EKS	Mode automatique EKS	Nœuds hybrides Amazon EKS
Vous devez mettre à jour la version du nœud Kubernetes par vous-même	Oui : si vous avez déployé une AMI optimisée pour Amazon EKS, vous êtes averti dans la console Amazon EKS lorsque les mises à jour sont disponibles. Vous pouvez effectuer la mise à jour en un clic dans la console. Si vous avez déployé une AMI personnal isée, vous n'êtes pas averti dans la console Amazon EKS lorsque des mises à jour sont disponibles. Vous devez effectuer la mise à jour par vous-même.	Non	Oui : Le système d'exploitation qui fonctionne sur vos machines physiques ou virtuelles est géré par vous-même à l'aide des outils de votre choix ou de `nodeadm`. Consultez the section called "Mettre à niveau les nœuds hybrides".
Possibilité d'utilise r le stockage Amazon EBS avec des pods	Oui	Oui, en tant que fonctionnalité intégrée. Découvrez comment créer une classe de stockage.	Non
Possibilité d'utiliser le stockage Amazon EFS avec des pods	Oui	Oui	Non

Critères	Groupes de nœuds gérés EKS	Mode automatique EKS	Nœuds hybrides Amazon EKS
Possibilité d'utiliser le stockage Amazon FSx pour Lustre avec des pods	Oui	Oui	Non
Possibilité d'utiliser le Network Load Balancer pour les services	Oui	Oui	Oui : Vous devez utiliser le type de cible `ip`.
Les pods peuvent s'exécuter dans un sous-réseau public	Oui	Oui	Non : les modules s'exécutent dans un environnement sur site.
Possibilité d'affecter différents groupes de sécurité VPC à des pods individuels	Oui : nœuds Linux uniquement	Non	Non
Possibilité d'exécuter des ensembles de démons Kubernetes	Oui	Oui	Oui
Prend en charge `HostPort` et `HostNetwork` dans le manifeste du pod	Oui	Oui	Oui

Critères	Groupes de nœuds gérés EKS	Mode automatique EKS	Nœuds hybrides Amazon EKS
AWS : disponibilité dans les régions	Toutes les régions prises en charge par Amazon EKS	Toutes les régions prises en charge par Amazon EKS	Toutes les régions prises en charge par Amazon EKS, à l'exception des régions AWS GovCloud (États-Unis) et de la Chine.
Possibilité d'exécuter des conteneurs sur des hôtes dédiés Amazon EC2	Oui	Non	Non
Tarification	Coût de l'instance Amazon EC2 qui exécute plusieurs pods. Pour plus d'informations, consultez Tarification Amazon EC2.	Lorsque le mode automatique EKS est activé dans votre cluster, vous payez des frais distincts, en plus des frais standard liés aux instances EC2, pour les instances lancées à l'aide de la capacité de calcul du mode automatique. Le montant varie en fonction du type d'instance lancé et de la région AWS où se trouve votre cluster. Pour plus d'informations, consultez les tarifs Amazon EKS.	Coût horaire des nœuds hybrides vCPU. Pour plus d'informations, consultez les tarifs Amazon EKS.

Simplifiez le cycle de vie des nœuds avec des groupes de nœuds gérés

Les groupes de nœuds gérés par Amazon EKS automatisent le provisionnement et la gestion du cycle de vie des nœuds (EC2 instances Amazon) pour les clusters Amazon EKS Kubernetes.

Avec les groupes de nœuds gérés par Amazon EKS, vous n'avez pas besoin de configurer ou d'enregistrer séparément les EC2 instances Amazon qui fournissent la capacité de calcul pour exécuter vos applications Kubernetes. Vous pouvez créer, mettre à jour ou résilier les nœuds pour votre cluster en une seule opération. Les mises à jour et les résiliations de nœuds purgent automatiquement les nœuds afin de garantir la disponibilité de vos applications.

Chaque nœud géré est approvisionné dans le cadre d'un groupe Amazon EC2 Auto Scaling géré pour vous par Amazon EKS. Toutes les ressources, y compris les instances et les groupes Auto Scaling, s'exécutent au sein de votre AWS compte. Chaque groupe de nœuds s'exécute dans plusieurs zones de disponibilité que vous définissez.

Les groupes de nœuds gérés peuvent également tirer parti, en option, de la fonctionnalité de réparation automatique des nœuds, qui surveille en permanence leur état. Il réagit automatiquement aux problèmes détectés et remplace les nœuds lorsque cela est possible. Cela contribue à la disponibilité globale du cluster avec un minimum d'intervention manuelle. Pour de plus amples informations, veuillez consulter the section called "État du nœud".

Vous pouvez ajouter un groupe de nœuds gérés à des clusters nouveaux ou existants à l'aide de la console, de la AWS CLI`eksctl`, de l' AWS API ou de l'infrastructure Amazon EKS en tant qu'outils de code, notamment AWS CloudFormation. Les nœuds lancés dans le cadre d'un groupe de nœuds gérés sont automatiquement marqués pour être détectés automatiquement par l'outil Cluster Autoscaler de Kubernetes. Vous pouvez utiliser le groupe de nœuds pour appliquer des labels Kubernetes aux nœuds et les mettre à jour à tout moment.

L'utilisation des groupes de nœuds gérés par Amazon EKS n'entraîne aucun coût supplémentaire. Vous ne payez que pour les AWS ressources que vous fournissez. Il s'agit notamment EC2 des instances Amazon, des volumes Amazon EBS, des heures de cluster Amazon EKS et de toute autre AWS infrastructure. Aucun frais minimum ni aucun engagement initial ne s'appliquent.

Pour démarrer avec un nouveau cluster Amazon EKS et un groupe de nœuds gérés, consultez the section called "Création d'un cluster (console et CLI)".

Pour ajouter un groupe de nœuds gérés à un cluster existant, consultez the section called "Créer".

Concepts des groupes de nœuds gérés

- Les groupes de nœuds gérés par Amazon EKS créent et gèrent EC2 des instances Amazon pour vous.
- Chaque nœud géré est approvisionné dans le cadre d'un groupe Amazon EC2 Auto Scaling géré pour vous par Amazon EKS. De plus, toutes les ressources, y compris les EC2 instances Amazon et les groupes Auto Scaling, sont exécutées au sein de votre AWS compte.
- Le groupe Auto Scaling d'un groupe de nœuds gérés couvre chaque sous-réseau que vous spécifiez lorsque vous créez le groupe.
- Amazon EKS étiquette les ressources du groupe de nœuds gérés afin qu'elles soient configurées pour utiliser l'outil Cluster Autoscaler de Kubernetes.

> **Important**
>
> Si vous exécutez une application avec état sur plusieurs zones de disponibilité, prise en charge par des volumes Amazon EBS et utilisant l'outil Cluster Autoscaler de Kubernetes, vous devez configurer plusieurs groupes de nœuds, chacun limité à une seule zone de disponibilité. En outre, vous devez activer la fonction `--balance-similar-node-groups`.

- Vous pouvez utiliser un modèle de lancement personnalisé pour un plus grand niveau de flexibilité et de personnalisation lors du déploiement de nœuds gérés. Par exemple, vous pouvez spécifier des arguments `kubelet` supplémentaires et utiliser une AMI personnalisée. Pour de plus amples informations, veuillez consulter the section called "Modèles de lancement". Si vous n'utilisez pas de modèle de lancement personnalisé lors de la première création d'un groupe de nœuds gérés, un modèle de lancement est généré automatiquement. Ne modifiez pas manuellement ce modèle généré automatiquement, sinon des erreurs se produiront.
- Amazon EKS suit le modèle de responsabilité partagée CVEs et les correctifs de sécurité sur les groupes de nœuds gérés. Lorsque les nœuds gérés exécutent une AMI optimisée pour Amazon EKS, cette dernière est chargée de créer des versions corrigées de l'AMI lorsque des bogues ou des problèmes sont signalés. Nous pouvons publier un correctif. Cependant, vous êtes responsable du déploiement de ces versions corrigées de l'AMI sur vos groupes de nœuds gérés. Lorsque les nœuds gérés exécutent une AMI personnalisée, vous êtes responsable de la création de versions corrigées de l'AMI lorsque des bogues ou des problèmes sont signalés, puis du déploiement de l'AMI. Pour de plus amples informations, veuillez consulter the section called "Mettre à jour".

- Les groupes de nœuds gérés par Amazon EKS peuvent être lancés dans des sous-réseaux publics et privés. Si vous lancez un groupe de nœuds gérés dans un sous-réseau public à partir du 22 avril 2020, `MapPublicIpOnLaunch` du sous-réseau doit avoir la valeur true pour que les instances puissent rejoindre un cluster. Si le sous-réseau public a été créé à l'aide `eksctl` des AWS CloudFormation modèles Amazon EKS vendus le 26 mars 2020 ou après cette date, ce paramètre est déjà défini sur true. Si les sous-réseaux publics ont été créés avant le 26 mars 2020, vous devez modifier le paramètre manuellement. Pour plus d'informations, consultez la section Modification de l'attribut d' IPv4 adressage public de votre sous-réseau.
- Lorsque vous déployez un groupe de nœuds gérés dans des sous-réseaux privés, vous devez vous assurer qu'il peut accéder à Amazon ECR pour extraire des images de conteneurs. Vous pouvez procéder en connectant une passerelle NAT à la table de routage du sous-réseau ou en ajoutant les points de terminaison d'un VPC AWS PrivateLink suivants :
  - Interface de point de terminaison de l'API Amazon ECR : `com.amazonaws.region-code.ecr.api`
  - Interface de point de terminaison de l'API de registre Docker Amazon ECR : `com.amazonaws.region-code.ecr.dkr`
  - Point de terminaison de la passerelle Amazon S3 : `com.amazonaws.region-code.s3`

  Pour d'autres services et points de terminaison couramment utilisés, veuillez consulter la rubrique the section called “Clusters privés”.
- Les groupes de nœuds gérés ne peuvent pas être déployés sur AWS Outposts ou dans AWS Wavelength. Des groupes de nœuds gérés peuvent être créés sur les zones locales AWS. Pour de plus amples informations, veuillez consulter the section called “ AWS Local Zones”.
- Vous pouvez créer plusieurs groupes de nœuds gérés au sein d'un même cluster. Par exemple, vous pouvez créer un groupe de nœuds avec l'AMI Amazon Linux optimisée Amazon EKS standard pour certaines charges de travail et un autre avec la variante GPU pour les charges de travail qui nécessitent la prise en charge GPU.
- Si votre groupe de nœuds gérés rencontre un échec de vérification de l'état d'une EC2 instance Amazon, Amazon EKS renvoie un code d'erreur pour vous aider à diagnostiquer le problème. Pour de plus amples informations, veuillez consulter the section called “Codes d'erreurs liées aux groupes de nœuds gérés”.
- Amazon EKS ajoute des labels Kubernetes aux instances de groupes de nœuds gérés. Ces labels fournis par Amazon EKS ont le préfixe `eks.amazonaws.com`.
- Amazon EKS purge automatiquement les nœuds à l'aide de l'API Kubernetes lors des résiliations ou des mises à jour.

- Les budgets de perturbation des pods ne sont pas respectés lors de l'arrêt d'un nœud avec `AZRebalance` ou de la réduction du nombre de nœuds souhaité. Ces actions tentent d'expulser les pods du nœud. Mais si l'opération prend plus de 15 minutes, le nœud est arrêté, que tous les pods du nœud aient été arrêtés ou non. Pour prolonger le délai d'arrêt du nœud, ajoutez un hook de cycle de vie au groupe Auto Scaling. Pour plus d'informations, consultez la section Ajouter des hooks de cycle de vie dans le guide de l'utilisateur d'Amazon EC2 Auto Scaling.
- Pour exécuter correctement le processus de vidange après avoir reçu une notification d'interruption ponctuelle ou une notification de rééquilibrage des capacités, `CapacityRebalance` doit être réglé sur `true`.
- La mise à jour des groupes de nœuds gérés respecte les budgets de perturbation des pods que vous avez définis pour vos pods. Pour de plus amples informations, veuillez consulter the section called "Mise à jour des informations des comportements".
- L'utilisation de groupes de nœuds gérés par Amazon EKS est disponible sans coûts supplémentaires. Vous ne payez que pour les AWS ressources que vous fournissez.
- Si vous souhaitez chiffrer les volumes Amazon EBS pour vos nœuds, vous pouvez déployer les nœuds à l'aide d'un modèle de lancement. Pour déployer des nœuds gérés avec des volumes Amazon EBS chiffrés sans utiliser de modèle de lancement, chiffrez tous les nouveaux volumes Amazon EBS créés dans votre compte. Pour plus d'informations, consultez la section Chiffrement par défaut dans le guide de EC2 l'utilisateur Amazon.

Types de capacité des groupes de nœuds gérés

Lorsque vous créez un groupe de nœuds gérés, vous pouvez choisir le type de capacité à la demande ou Spot. Amazon EKS déploie un groupe de nœuds gérés avec un groupe Amazon EC2 Auto Scaling qui contient uniquement des instances On-Demand ou uniquement des instances Amazon EC2 Spot. Vous pouvez planifier des pods pour des applications tolérantes aux pannes dans des groupes de nœuds gérés Spot, et des applications intolérantes aux pannes dans des groupes de nœuds à la demande au sein d'un même cluster Kubernetes. Par défaut, un groupe de nœuds gérés déploie les EC2 instances Amazon On-Demand.

À la demande

Avec les instances à la demande, vous payez la capacité de calcul à la seconde, sans engagement à long terme.

Par défaut, si vous ne spécifiez pas de Type de capacité, le groupe de nœuds gérés est alloué avec des instances à la demande. Un groupe de nœuds géré configure un groupe Amazon EC2 Auto Scaling en votre nom avec les paramètres suivants appliqués :

- La stratégie d'allocation pour fournir la capacité à la demande est définie sur `prioritized`. Les groupes de nœuds gérés utilisent l'ordre des types d'instance transmis dans l'API pour déterminer le type d'instance à utiliser en premier lors de la fourniture de la capacité à la demande. Par exemple, vous pouvez spécifier trois types d'instances dans l'ordre suivant : `c5.large`, `c4.large` et `c3.large`. Lorsque vos instances à la demande sont lancées, le groupe de nœuds gérés fournit la capacité à la demande en commençant par `c5.large`, puis `c4.large` et enfin `c3.large`. Pour plus d'informations, consultez le groupe Amazon EC2 Auto Scaling dans le guide de l'utilisateur Amazon EC2 Auto Scaling.
- Amazon EKS ajoute le label Kubernetes suivant à tous les nœuds de votre groupe de nœuds gérés qui spécifie le type de capacité : `eks.amazonaws.com/capacityType: ON_DEMAND` Vous pouvez utiliser ce label pour programmer des applications à état ou intolérantes aux pannes sur des nœuds à la demande.

Spot

Les instances Amazon EC2 Spot sont des EC2 capacités inutilisées d'Amazon qui offrent des remises importantes par rapport aux prix à la demande. Les instances Amazon EC2 Spot peuvent être interrompues moyennant un préavis de deux minutes lorsqu'il est EC2 nécessaire de rétablir leur capacité. Pour plus d'informations, consultez la section Instances Spot dans le guide de EC2 l'utilisateur Amazon. Vous pouvez configurer un groupe de nœuds gérés avec des instances Amazon EC2 Spot afin d'optimiser les coûts des nœuds de calcul exécutés dans votre cluster Amazon EKS.

Pour utiliser des instances Spot dans un groupe de nœuds gérés, créez un groupe de nœuds gérés en définissant le type de capacité comme `spot`. Un groupe de nœuds géré configure un groupe Amazon EC2 Auto Scaling en votre nom en appliquant les meilleures pratiques Spot suivantes :

- Pour garantir que vos nœuds Spot sont alloués dans les groupes de capacité Spot optimaux, la stratégie d'allocation est définie sur l'une des options suivantes :
  - `price-capacity-optimized` (PCO) : lors de la création de nouveaux groupes de nœuds dans un cluster avec la version Kubernetes `1.28` ou supérieure, la stratégie d'allocation est définie sur `price-capacity-optimized`. Cependant, la stratégie d'allocation ne sera pas modifiée pour les groupes de nœuds déjà créés avec `capacity-optimized` avant que les groupes de nœuds gérés par Amazon EKS ne commencent à prendre en charge le PCO.

  - `capacity-optimized` (CO) : lors de la création de nouveaux groupes de nœuds dans un cluster avec la version Kubernetes `1.27` ou une version antérieure, la stratégie d'allocation est définie sur `capacity-optimized`.

  Pour augmenter le nombre de groupes de capacité Spot disponibles pour l'allocation de capacité, configurez un groupe de nœuds gérés pour utiliser plusieurs types d'instances.
- Le rééquilibrage de capacité Amazon EC2 Spot est activé afin qu'Amazon EKS puisse drainer et rééquilibrer en douceur vos nœuds Spot afin de minimiser les perturbations des applications lorsqu'un nœud Spot présente un risque élevé d'interruption. Pour plus d'informations, consultez Amazon EC2 Auto Scaling Capacity Rebalancing dans le guide de l'utilisateur d'Amazon EC2 Auto Scaling.
  - Lorsqu'un nœud Spot reçoit une recommandation de rééquilibrage, Amazon EKS tente automatiquement de lancer un nouveau nœud Spot de remplacement.
  - Si un avis d'interruption de deux minutes arrive avant que le nœud Spot de remplacement ne soit dans l'état `Ready`, Amazon EKS commence à purger le nœud Spot qui a reçu la recommandation de rééquilibrage. Amazon EKS vide le nœud dans la mesure du possible. Par conséquent, rien ne garantit qu'Amazon EKS attendra que le nœud de remplacement rejoigne le cluster avant de vider le nœud existant.
  - Lorsqu'un nœud Spot de remplacement est démarré et est dans l'état `Ready` sur Kubernetes, Amazon EKS cloisonne et purge le nœud Spot qui a reçu la recommandation de rééquilibrage. Le cloisonnement du nœud Spot garantit que le contrôleur de service n'envoie pas de nouvelles demandes à ce nœud Spot. Il le supprime également de sa liste de nœuds Spot sains et actifs. La purge du nœud Spot garantit que les pods en cours d'exécution sont expulsés proprement.
- Amazon EKS ajoute le label Kubernetes suivant à tous les nœuds de votre groupe de nœuds gérés qui spécifie le type de capacité : `eks.amazonaws.com/capacityType: SPOT` Vous pouvez utiliser ce label pour programmer des applications tolérantes aux pannes sur les nœuds Spot.

> **Important**
>
> EC2 émet un avis d'interruption Spot deux minutes avant de mettre fin à votre instance Spot. Cependant, les nœuds Pods on Spot peuvent ne pas bénéficier de la fenêtre complète de 2 minutes pour un arrêt progressif. Lorsque l'avis est EC2 émis, Amazon EKS ne commence à expulser les Pods qu'après un certain délai. Les expulsions se produisent de manière séquentielle pour protéger le serveur d'API Kubernetes. Ainsi, lors de plusieurs réclamations simultanées de Spot, certains Pods peuvent recevoir des notifications d'expulsion différées. Les pods peuvent être fermés de force sans recevoir de signaux

de terminaison, en particulier sur les nœuds à forte densité de pods, lors de réclamations simultanées ou lors de l'utilisation de longues périodes de grâce de terminaison. Pour les charges de travail Spot, nous recommandons de concevoir des applications tolérantes aux interruptions, d'utiliser des délais de résiliation de 30 secondes ou moins, d'éviter les longs hooks Prestop et de surveiller les indicateurs d'éviction des Pod pour comprendre les délais de grâce réels dans vos clusters. Pour les charges de travail nécessitant une résiliation progressive garantie, nous recommandons plutôt d'utiliser la capacité à la demande.

Lorsque vous décidez de déployer un groupe de nœuds avec une capacité à la demande ou Spot, vous devez tenir compte des conditions suivantes :

- Les instances Spot conviennent bien aux applications sans état, tolérantes aux pannes et flexibles. Il s'agit notamment des charges de travail de formation par lots et de machine learning, des mégadonnées ETLs telles qu'Apache Spark, des applications de traitement des files d'attente et des points de terminaison d'API sans état. Spot étant une EC2 capacité inutilisée d'Amazon, qui peut changer au fil du temps, nous vous recommandons d'utiliser la capacité Spot pour les charges de travail tolérantes aux interruptions. Plus précisément, la capacité Spot convient aux charges de travail qui peuvent tolérer des périodes où la capacité requise n'est pas disponible.
- Nous vous recommandons d'utiliser la capacité à la demande pour les applications qui sont intolérantes aux pannes. Cela inclut les outils de gestion de cluster tels que les outils de surveillance et d'exploitation, les déploiements qui nécessitent `StatefulSets`, et les applications avec état, telles que les bases de données.
- Pour maximiser la disponibilité de vos applications en utilisant les instances Spot, nous vous recommandons de configurer un groupe de nœuds gérés Spot pour utiliser plusieurs types d'instances. Nous vous recommandons d'appliquer les règles suivantes lorsque vous utilisez plusieurs types d'instance :
  - Dans un groupe de nœuds gérés, si vous utilisez Cluster Autoscaler, nous vous recommandons d'utiliser un ensemble flexible de types d'instances avec la même quantité de ressources vCPU et de mémoire. Ceci afin de garantir que les nœuds de votre cluster soient mis à l'échelle comme prévu. Par exemple, si vous avez besoin de quatre V CPUs et de huit GiB de mémoire, utilisez`c3.xlarge`,,`c4.xlarge`,`c5.xlarge`, `c5d.xlarge` `c5a.xlarge``c5n.xlarge`, ou d'autres types d'instance similaires.
  - Pour améliorer la disponibilité des applications, nous recommandons de déployer plusieurs groupes de nœuds gérés Spot. Pour cela, chaque groupe doit utiliser un ensemble flexible

de types d'instances qui disposent des mêmes ressources vCPU et mémoire. Par exemple, si vous avez besoin de 4 V CPUs et 8 GiB de mémoire, nous vous recommandons de créer un groupe de nœuds gérés avec`c3.xlarge`,,`c4.xlarge`,`c5.xlarge`, `c5d.xlarge` `c5a.xlarge``c5n.xlarge`, ou d'autres types d'instances similaires, et un second groupe de nœuds gérés avec`m3.xlarge`,,, `m4.xlarge` `m5.xlarge` `m5d.xlarge``m5a.xlarge`, `m5n.xlarge` ou d'autres types d'instances similaires.

- Lorsque vous déployez votre groupe de nœuds avec le type de capacité Spot qui utilise un modèle de lancement personnalisé, utilisez l'API pour transmettre plusieurs types d'instances. Ne transmettez pas un seul type d'instance via le modèle de lancement. Pour plus d'informations sur le déploiement d'un groupe de nœuds à l'aide d'un modèle de lancement, consultez the section called "Modèles de lancement".

Création d'un groupe de nœuds gérés pour votre cluster

Cette rubrique explique comment lancer des groupes de nœuds gérés Amazon EKS composés de nœuds qui s'enregistrent dans votre cluster Amazon EKS. Une fois que les nœuds ont rejoint le cluster, vous pouvez y déployer des applications Kubernetes.

Si vous créez un groupe de nœuds gérés par Amazon EKS pour la première fois, nous vous recommandons plutôt de suivre l'un des guides dans Démarrer. Ces guides fournissent des procédures détaillées pour créer un cluster Amazon EKS avec des nœuds.

⚠ Important

- Les nœuds Amazon EKS sont des instances Amazon EC2 standard. La facturation s'applique selon les tarifs normaux d'Amazon EC2. Pour plus d'informations, consultez Tarification Amazon EC2.
- Vous ne pouvez pas créer de nœuds gérés dans une AWS région où AWS Outposts ou AWS Wavelength sont activés. Vous pouvez créer des nœuds autogérés à la place. Pour plus d'informations, consultez the section called "Amazon Linux", the section called "Windows" et the section called "Flacon Rocket". Vous pouvez également créer un groupe de nœuds Amazon Linux autogéré sur un Outpost. Pour de plus amples informations, veuillez consulter the section called "Nœuds".
- Si vous ne spécifiez pas un ID d'AMI pour le fichier `bootstrap.sh` inclus dans Linux ou Bottlerocket optimisé pour Amazon EKS, les groupes de nœuds gérés imposent un nombre maximal à la valeur de `maxPods`. Pour les instances de moins de 30 VCPUs,

le nombre maximum est de`110`. Pour les instances avec une tension supérieure à 30 VCPUs, le nombre maximum passe à`250`. Cette application remplace les autres `maxPods` configurations, notamment`maxPodsExpression`. Pour plus d'informations sur le `maxPods` mode de détermination et sur la manière de le personnaliser, consultezthe section called "Comment est déterminé MaxPods".

- Un cluster Amazon EKS existant. Pour en déployer un, consultez the section called "Créer un cluster".
- Un rôle IAM existant pour les nœuds à utiliser. Pour en créer un, consultez the section called "Rôle IAM de nœud". Si ce rôle ne comporte aucune des politiques pour le VPC CNI, le rôle distinct suivant est nécessaire pour les pods VPC CNI.
- (Facultatif, mais recommandé) Le plug-in CNI Amazon VPC pour le module complémentaire Kubernetes est configuré avec son propre rôle IAM auquel est associée la politique IAM nécessaire. Pour de plus amples informations, veuillez consulter the section called "Configuration pour IRSA".
- Connaissance des considérations répertoriées dans Choisissez un type d'instance de nœud Amazon EC2 optimal. Selon le type d'instance que vous choisissez, il peut y avoir des prérequis supplémentaires pour votre cluster et votre VPC.
- Pour ajouter un groupe de nœuds gérés Windows, vous devez d'abord activer la prise en charge Windows pour votre cluster. Pour de plus amples informations, veuillez consulter the section called "Activer la prise en charge de Windows".

Vous pouvez créer un groupe de nœuds gérés avec l'une des options suivantes :

- the section called "`eksctl`"
- the section called "AWS Management Console"

`eksctl`

Création d'un groupe de nœuds gérés avec eksctl

Cette procédure nécessite `eksctl` version `0.215.0` ou ultérieure. Vous pouvez vérifier votre version avec la commande suivante :

```
eksctl version
```

Pour les instructions d'installation ou de mise à niveau de `eksctl`, consultez la rubrique Installation dans la documentation `eksctl`.

1. (Facultatif) Si la politique IAM gérée AmazonEKS_CNI_Policy est attachée à votre rôle IAM de nœud Amazon EKS, nous recommandons de l'attribuer plutôt à un rôle IAM associé au compte de service `aws-node` Kubernetes. Pour de plus amples informations, veuillez consulter the section called "Configuration pour IRSA".
2. Créez un groupe de nœuds gérés avec ou sans modèle de lancement personnalisé. La spécification manuelle d'un modèle de lancement permet une plus grande personnalisation d'un groupe de nœuds. Par exemple, cela peut permettre de déployer une AMI personnalisée ou de fournir des arguments au script `boostrap.sh` dans une AMI optimisée par Amazon EKS. Pour obtenir la liste complète de toutes les options disponibles et des valeurs par défaut, saisissez la commande suivante.

   ```
   eksctl create nodegroup --help
   ```

   Dans la commande suivante, remplacez *`my-cluster`* par le nom de votre cluster et remplacez *`my-mng`* par le nom de votre groupe de nœuds. Le nom du groupe de nœuds ne peut pas dépasser 63 caractères. Il doit commencer par une lettre ou un chiffre, mais peut également inclure des tirets et des traits de soulignement pour les autres caractères.

   > **Important**
   >
   > Si vous n'utilisez pas de modèle de lancement personnalisé lors de la création initiale d'un groupe de nœuds gérés, n'en utilisez pas plus tard pour ce groupe de nœuds. Si vous n'avez pas spécifié de modèle de lancement personnalisé, le système génère automatiquement un modèle de lancement qu'il n'est pas recommandé de modifier manuellement. La modification manuelle de ce modèle de lancement généré automatiquement peut entraîner des erreurs.

Sans modèle de lancement

`eksctl` crée un modèle de lancement Amazon EC2 par défaut dans votre compte et déploie le groupe de nœuds à l'aide d'un modèle de lancement qu'il crée en fonction des options que vous spécifiez. Avant de spécifier une valeur pour `--node-type`, consultez the section called "Types d'instances Amazon EC2".

Remplacez *ami-family* par un mot clé autorisé. Pour plus d'informations, consultez Définition de la famille AMI de nœuds dans la documentation `eksctl`. Remplacez *my-key* par le nom de votre paire de clés Amazon EC2 ou de votre clé publique. Cette clé est utilisée pour SSH dans vos nœuds après leur lancement.

> **Note**
>
> Pour Windows, cette commande n'active pas SSH. Elle associe votre key pair Amazon EC2 à l'instance et vous permet d'accéder à l'instance.

Si vous n'avez pas encore de paire de clés Amazon EC2, vous pouvez en créer une dans AWS Management Console. Pour plus d'informations Linux, consultez Paires de clés Amazon EC2 et instances Linux dans le Guide de l'utilisateur Amazon EC2. Pour plus d'informations Windows, consultez Paires de clés Amazon EC2 et instances Windows dans le Guide de l'utilisateur Amazon EC2.

Nous recommandons de bloquer l'accès des pods à IMDS si les conditions suivantes sont remplies :

- Vous prévoyez d'attribuer des rôles IAM à tous vos comptes de service Kubernetes, afin que les pods ne disposent que des permissions strictement nécessaires.
- Aucun pod du cluster n'a besoin d'accéder au service de métadonnées d'instance Amazon EC2 (IMDS) pour d'autres raisons, telles que la récupération de la région actuelle. AWS

Pour plus d'informations, consultez Restreindre l'accès au profil d'instance affecté au composant master.

Si vous souhaitez bloquer l'accès des pods à IMDS, ajoutez l'option `--disable-pod-imds` à la commande suivante.

```
eksctl create nodegroup \
  --cluster my-cluster \
  --region region-code \
  --name my-mng \
  --node-ami-family ami-family \
  --node-type m5.large \
  --nodes 3 \
  --nodes-min 2 \
  --nodes-max 4 \
```

```
  --ssh-access \
  --ssh-public-key my-key
```

Vos instances peuvent éventuellement attribuer un nombre significativement plus élevé d'adresses IP aux pods, attribuer des adresses IP aux pods à partir d'un bloc CIDR différent de celui de l'instance, et être déployées dans un cluster sans accès Internet. Pour plus d'informations, consultez the section called "Augmentation du nombre d'adresses IP", the section called "Mise en réseau personnalisée" et the section called "Clusters privés" pour connaître les options supplémentaires à ajouter à la commande précédente.

Les groupes de nœuds gérés calculent et appliquent une valeur unique pour le nombre maximum de pods qui peuvent fonctionner sur chaque nœud de votre groupe de nœuds, en fonction du type d'instance. Si vous créez un groupe de nœuds avec différents types d'instances, la plus petite valeur calculée sur tous les types d'instances est appliquée comme nombre maximal de pods pouvant fonctionner sur chaque type d'instance du groupe de nœuds. Les groupes de nœuds gérés calculent cette valeur à l'aide du script référencé dans Nombre maximal recommandé de pods par type d'instance Amazon EC2 pour Amazon EKS.

Avec un modèle de lancement

Le modèle de lancement doit déjà exister et satisfaire aux exigences précisées dans Principes de configuration du modèle de lancement. Nous recommandons de bloquer l'accès des pods à IMDS si les conditions suivantes sont remplies :

- Vous prévoyez d'attribuer des rôles IAM à tous vos comptes de service Kubernetes, afin que les pods ne disposent que des permissions strictement nécessaires.
- Aucun pod du cluster n'a besoin d'accéder au service de métadonnées d'instance Amazon EC2 (IMDS) pour d'autres raisons, telles que la récupération de la région actuelle. AWS

Pour plus d'informations, consultez Restreindre l'accès au profil d'instance affecté au composant master.

Si vous souhaitez bloquer l'accès des pods à IMDS, spécifiez les paramètres nécessaires dans le modèle de lancement.

a. Copiez les contenus suivants sur votre appareil. Remplacez les valeurs d'exemple, puis exécutez la commande modifiée pour créer le `eks-nodegroup.yaml` fichier. Plusieurs paramètres que vous spécifiez lors d'un déploiement sans modèle de lancement sont déplacés dans le modèle de lancement. Si vous ne spécifiez pas de `version`, la version par défaut du modèle est utilisée.

```
cat >eks-nodegroup.yaml <<EOF
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
metadata:
  name: my-cluster
  region: region-code
managedNodeGroups:
- name: my-mng
  launchTemplate:
    id: lt-id
    version: "1"
EOF
```

Pour obtenir la liste complète des paramètres du fichier de configuration `eksctl`, consultez Schéma de fichier de configuration dans la documentation `eksctl`. Vos instances peuvent éventuellement attribuer un nombre significativement plus élevé d'adresses IP aux pods, attribuer des adresses IP aux pods à partir d'un bloc CIDR différent de celui de l'instance, et être déployées dans un cluster sans accès Internet sortant. Pour plus d'informations, consultez the section called "Augmentation du nombre d'adresses IP", the section called "Mise en réseau personnalisée" et the section called "Clusters privés", qui présentent des options supplémentaires à intégrer dans le fichier de configuration.

Si vous n'avez pas spécifié d'ID AMI dans votre modèle de lancement, les groupes de nœuds gérés calculent et appliquent une valeur unique pour le nombre maximal de pods pouvant fonctionner sur chaque nœud du groupe, selon le type d'instance. Si vous créez un groupe de nœuds avec différents types d'instances, la plus petite valeur calculée sur tous les types d'instances est appliquée comme nombre maximal de pods pouvant fonctionner sur chaque type d'instance du groupe de nœuds. Les groupes de nœuds gérés calculent cette valeur à l'aide du script référencé dans Nombre maximal recommandé de pods par type d'instance Amazon EC2 pour Amazon EKS.

Si vous avez spécifié un ID AMI dans votre modèle de lancement, indiquez le nombre maximal de pods pouvant fonctionner sur chaque nœud du groupe si vous utilisez le réseau personnalisé ou si vous souhaitez augmenter le nombre d'adresses IP attribuées à votre instance. Pour de plus amples informations, veuillez consulter the section called "Nombre maximal de pods recommandé par Amazon EKS pour chaque type d'instance Amazon EC2".

b. Déployez le groupe de nœuds avec la commande suivante.

```
eksctl create nodegroup --config-file eks-nodegroup.yaml
```

AWS Management Console

Création d'un groupe de nœuds gérés à l'aide de la AWS Management Console

1. Attendez que le statut de votre cluster s'affiche soit `ACTIVE`. Vous ne pouvez pas créer un groupe de nœuds gérés pour un cluster qui n'est pas déjà `ACTIVE`.
2. Ouvrez la console Amazon EKS.
3. Choisissez le nom du cluster dans lequel vous souhaitez créer un groupe de nœuds gérés.
4. Sélectionner l'onglet Calcul.
5. Choisissez Ajouter un groupe de nœuds.
6. Sur la page Configurer un groupe de nœuds, définissez les paramètres en conséquence, puis choisissez Next (Suivant).
   - Nom : saisissez un nom unique pour votre groupe de nœuds gérés. Le nom du groupe de nœuds ne peut pas dépasser 63 caractères. Il doit commencer par une lettre ou un chiffre, mais peut également inclure des tirets et des traits de soulignement pour les autres caractères.
   - Rôle IAM de nœud : choisissez le rôle d'instance de nœud à utiliser avec votre groupe de nœuds. Pour de plus amples informations, veuillez consulter the section called "Rôle IAM de nœud".

   > **Important**
   > - Vous ne pouvez pas utiliser le même rôle que celui utilisé pour créer des clusters.
   > - Nous vous recommandons d'utiliser un rôle qui n'est pas actuellement utilisé par un groupe de nœuds autogérés. Sinon, vous prévoyez de l'utiliser avec un nouveau groupe de nœuds autogérés. Pour de plus amples informations, veuillez consulter the section called "Suppression".

   - Utiliser le modèle de lancement : (facultatif) choisissez si vous souhaitez utiliser un modèle de lancement existant. Sélectionnez un Nom de modèle de lancement. Sélectionnez ensuite une Version du modèle de lancement. Si vous ne sélectionnez pas de version, Amazon EKS utilise la version par défaut du modèle. Les modèles de lancement permettent de personnaliser

davantage votre groupe de nœuds, par exemple en vous permettant de déployer une AMI personnalisée, d'attribuer un nombre nettement plus élevé d'adresses IP aux pods, d'attribuer aux pods des adresses IP provenant d'un bloc CIDR différent de celui de l'instance et de déployer des nœuds dans un cluster sans accès Internet sortant. Pour plus d'informations, consultez the section called "Augmentation du nombre d'adresses IP", the section called "Mise en réseau personnalisée" et the section called "Clusters privés".

Le modèle de lancement doit satisfaire aux exigences présentées dans Personnalisation des nœuds gérés avec des modèles de lancement. Si vous n'utilisez pas votre propre modèle de lancement, l'API Amazon EKS crée un modèle de lancement Amazon EC2 par défaut dans votre compte et déploie le groupe de nœuds en utilisant ce modèle par défaut.

Si vous implémentez des rôles IAM pour les comptes de service, si vous attribuez les autorisations nécessaires directement à chaque pod nécessitant un accès aux AWS services, et si aucun pod de votre cluster n'a besoin d'accéder à IMDS pour d'autres raisons, telles que la récupération de la AWS région actuelle, vous pouvez également désactiver l'accès à IMDS pour les pods qui n'utilisent pas le réseau hôte dans un modèle de lancement. Pour plus d'informations, consultez Restreindre l'accès au profil d'instance affecté au composant master.

- Labels Kubernetes : (facultatif) vous pouvez choisir d'appliquer des labels Kubernetes aux nœuds de votre groupe de nœuds gérés.
- Rejets Kubernetes : (facultatif) vous pouvez choisir d'appliquer des rejets Kubernetes aux nœuds de votre groupe de nœuds gérés. Les options disponibles dans le menu Effet sont **`NoSchedule`** , **`NoExecute`** , et **`PreferNoSchedule`** . Pour de plus amples informations, veuillez consulter the section called "Empêcher la planification de pods sur des nœuds spécifiques".
- Identifications : (facultatif) vous pouvez choisir d'identifier votre groupe de nœuds gérés Amazon EKS. Ces balises ne se propagent pas aux autres ressources du groupe de nœuds, comme les groupes Auto Scaling ou les instances. Pour de plus amples informations, veuillez consulter the section called "Balisage de vos ressources ".

7. Sur la page Définir la configuration de calcul et de mise à l'échelle, définissez les paramètres en conséquence, puis choisissez Next (Suivant).
   - Type d'AMI : sélectionnez un type d'AMI. Si vous déployez des instances Arm, assurez-vous de prendre en compte les points à prendre en compte dans Arm Amazon Linux optimisé pour Amazon EKS AMIs avant le déploiement.

Si vous avez spécifié un modèle de lancement à l'étape précédente et que vous y avez indiqué une AMI, vous ne pouvez pas sélectionner une autre valeur. La valeur du modèle s'affiche. L'AMI spécifiée dans le modèle doit satisfaire aux exigences décrites dans Spécification d'une AMI.

- Type de capacité : sélectionnez un type de capacité. Pour plus d'informations sur le choix d'un type de capacité, consultez the section called "Types de capacité des groupes de nœuds gérés". Vous ne pouvez pas combiner différents types de capacités au sein d'un même groupe de nœuds. Si vous souhaitez utiliser les deux types de capacités, créez des groupes de nœuds distincts, chacun avec son propre type de capacité et d'instance. Consultez la section Réserver GPUs pour les groupes de nœuds gérés pour obtenir des informations sur le provisionnement et le dimensionnement des nœuds de travail accélérés par GPU.
- Types d'instance : par défaut, un ou plusieurs types d'instances sont spécifiés. Pour supprimer un type d'instance par défaut, sélectionnez la croix (X) sur le côté droit du type d'instance. Choisissez les types d'instances à utiliser dans votre groupe de nœuds gérés. Pour de plus amples informations, veuillez consulter the section called "Types d'instances Amazon EC2".

  La console affiche un ensemble de types d'instances couramment utilisés. Si vous devez créer un groupe de nœuds gérés avec un type d'instance qui n'est pas affiché`eksctl`, utilisez la AWS CLI ou un SDK pour créer le groupe de nœuds. AWS CloudFormation Si vous avez spécifié un modèle de lancement à l'étape précédente, vous ne pouvez pas sélectionner une valeur, car le type d'instance doit être spécifié dans le modèle de lancement. La valeur du modèle de lancement s'affiche. Si vous avez sélectionné Spot pour le type de capacité, nous vous recommandons de spécifier plusieurs types d'instances pour améliorer la disponibilité.
- Taille du disque : saisissez la taille du disque (en Gio) à utiliser pour le volume racine du nœud.

  Si vous avez spécifié un modèle de lancement à la page précédente, vous ne pouvez pas sélectionner une valeur, car elle doit être spécifiée dans le modèle de lancement.
- Taille souhaitée : spécifiez le nombre actuel de nœuds que le groupe de nœuds gérés doit conserver au lancement.

> **Note**
>
> Amazon EKS n'augmente ni ne réduit pas automatiquement votre groupe de nœuds. Cependant, vous pouvez configurer l'outil Cluster Autoscaler de Kubernetes pour

effectuer cette mise à l'échelle. Pour plus d'informations, consultez Cluster Autoscaler activé. AWS

- Taille minimale : spécifiez le nombre minimal de nœuds vers lequel le groupe de nœuds gérés peut être mis à l'échelle.
- Taille maximale : spécifiez le nombre maximal de nœuds vers lequel le groupe de nœuds gérés peut être mis à niveau.
- Configuration des mises à jour du groupe : (facultatif) vous pouvez sélectionner le nombre ou le pourcentage de nœuds à mettre à jour en parallèle. Ces nœuds seront indisponibles pendant la mise à jour. Pour Maximum non disponible, sélectionnez l'une des options suivantes et spécifiez une valeur :
  - Nombre : sélectionnez et spécifiez le nombre de nœuds de votre groupe de nœuds pouvant être mis à jour en parallèle.
  - Pourcentage : sélectionnez et spécifiez le pourcentage de nœuds de votre groupe de nœuds pouvant être mis à jour en parallèle. Cela est pratique si votre groupe de nœuds contient de nombreux nœuds.
- Configuration de la réparation automatique des nœuds : (facultatif) Si vous cochez la case Activer la réparation automatique des nœuds, Amazon EKS remplacera automatiquement les nœuds lorsque des problèmes seront détectés. Pour de plus amples informations, veuillez consulter the section called "État du nœud".

8. Sur la page Spécifier les détails, définissez les paramètres en conséquence, puis choisissez Next (Suivant).
   - Sous-réseaux : choisissez les sous-réseaux dans lesquels vous souhaitez lancer vos nœuds gérés.

> **Important**
>
> Si vous exécutez une application avec état sur plusieurs zones de disponibilité, prise en charge par des volumes Amazon EBS et utilisant l'outil Cluster Autoscaler de Kubernetes, vous devez configurer plusieurs groupes de nœuds, chacun limité à une seule zone de disponibilité. En outre, vous devez activer la fonction `--balance-similar-node-groups`.

> **Important**
>
> - Si vous choisissez un sous-réseau public et que seul le point de terminaison du serveur d'API publique est activé, le paramètre `MapPublicIP0nLaunch` du sous-réseau doit avoir la valeur `true` pour que les instances rejoignent un cluster. Si le sous-réseau a été créé à l'aide `eksctl` des AWS CloudFormation modèles vendus par Amazon EKS le 26 mars 2020 ou après cette date, ce paramètre est déjà défini sur. `true` Si les sous-réseaux ont été créés avec les AWS CloudFormation modèles `eksctl` ou avant le 26 mars 2020, vous devez modifier le paramètre manuellement. Pour plus d'informations, consultez la section Modification de l'attribut d' IPv4 adressage public de votre sous-réseau.
> - Si vous utilisez un modèle de lancement et spécifiez plusieurs interfaces réseau, Amazon EC2 n'attribuera pas automatiquement d'adresse `IPv4` publique, même si `MapPublicIp0nLaunch` est défini sur `true`. Pour que les nœuds puissent rejoindre le cluster dans ce scénario, vous devez soit activer le point de terminaison privé du serveur d'API du cluster, soit lancer les nœuds dans un sous-réseau privé avec un accès Internet sortant via une méthode alternative, comme une passerelle NAT. Pour plus d'informations, consultez Adressage IP des instances Amazon EC2 dans le Guide de l'utilisateur Amazon EC2.

- Configurez l'accès SSH aux nœuds (facultatif). L'activation de SSH vous permet de vous connecter à vos instances et de recueillir des informations de diagnostic en cas de problème. Nous vous recommandons vivement d'activer l'accès à distance lorsque vous créez un groupe de nœuds. Vous ne pouvez pas activer l'accès à distance après la création du groupe de nœuds.

  Si vous avez choisi d'utiliser un modèle de lancement, cette option n'est pas affichée. Pour autoriser l'accès à distance pour vos noeuds, spécifiez une paire de clés dans le modèle de lancement et assurez-vous que le port approprié est ouvert aux nœuds des groupes de sécurité que vous spécifiez dans le modèle de lancement. Pour de plus amples informations, veuillez consulter the section called "Utilisation des groupes de sécurité".

> **Note**
>
> Pour Windows, cette commande n'active pas SSH. Elle associe votre key pair Amazon EC2 à l'instance et vous permet d'accéder à l'instance.

- Pour Paire de clés SSH (facultatif), choisissez une clé SSH Amazon EC2 à utiliser. Pour plus d'informations Linux, consultez Paires de clés Amazon EC2 et instances Linux dans le Guide de l'utilisateur Amazon EC2. Pour plus d'informations Windows, consultez Paires de clés Amazon EC2 et instances Windows dans le Guide de l'utilisateur Amazon EC2. Si vous avez choisi d'utiliser un modèle de lancement, vous ne pouvez pas en sélectionner un. Lorsqu'une clé SSH Amazon EC2 est fournie à des groupes de nœuds utilisant Bottlerocket AMIs, le conteneur administratif est également activé. Pour plus d'informations, consultez Conteneur d'administration sur GitHub.
- Pour Autoriser l'accès à distance SSH depuis, si vous souhaitez limiter l'accès à des instances spécifiques, sélectionnez les groupes de sécurité associés à ces instances. Si vous ne sélectionnez pas de groupes de sécurité spécifiques, alors l'accès SSH est autorisé depuis n'importe où sur Internet (`0.0.0.0/0`).

9. Sur la page Review and create (Vérifier et créer), vérifiez la configuration de votre groupe de nœuds gérés et choisissez Create (Créer).

   Si les nœuds ne parviennent pas à rejoindre le cluster, consultez la section the section called "Les nœuds ne parviennent pas à joindre le cluster" du chapitre Dépannage.

10. Observez le statut de vos nœuds et attendez qu'ils obtiennent le statut `Ready`.

```
kubectl get nodes --watch
```

11. (Nœuds GPU uniquement) Si vous avez choisi un type d'instance GPU et une AMI accélérée optimisée pour Amazon EKS, vous devez appliquer le plug-in d'appareil NVIDIA pour Kubernetes DaemonSet sur votre cluster. *vX.X.X*Remplacez-le par la s-device-plugin version Nvidia/K8 de votre choix avant d'exécuter la commande suivante.

```
kubectl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/vX.X.X/
deployments/static/nvidia-device-plugin.yml
```

Installation des modules complémentaires Kubernetes

Maintenant que vous disposez d'un cluster Amazon EKS opérationnel avec des nœuds, vous pouvez commencer à installer des modules complémentaires Kubernetes et à déployer des applications sur votre cluster. Les rubriques suivantes de la documentation vous aideront à étendre les fonctionnalités de votre cluster.

- Le principal IAM qui a créé le cluster est le seul principal qui peut effectuer des appels vers le serveur API Kubernetes avec `kubectl` ou le AWS Management Console. Si vous souhaitez que d'autres principaux IAM aient accès à votre cluster, vous devez les ajouter. Pour plus d'informations, consultez the section called "Accès à l'API Kubernetes" et the section called "Autorisations requises".
- Nous recommandons de bloquer l'accès des pods à IMDS si les conditions suivantes sont remplies :
  - Vous prévoyez d'attribuer des rôles IAM à tous vos comptes de service Kubernetes, afin que les pods ne disposent que des permissions strictement nécessaires.
  - Aucun pod du cluster n'a besoin d'accéder au service de métadonnées d'instance Amazon EC2 (IMDS) pour d'autres raisons, telles que la récupération de la région actuelle. AWS

  Pour plus d'informations, consultez Restreindre l'accès au profil d'instance affecté au composant master.
- Configurez l'outil Cluster Autoscaler de Kubernetes pour ajuster automatiquement le nombre de nœuds dans vos groupes de nœuds.
- Déployez un exemple d'application sur votre cluster.
- Organisez et surveillez les ressources du cluster à l'aide d'outils importants pour gérer votre cluster.

Mettre à jour un groupe de nœuds gérés pour votre cluster

Lorsque vous lancez une mise à jour d'un groupe de nœuds gérés, Amazon EKS met automatiquement à jour vos nœuds pour vous, en suivant les étapes décrites dans la section Comprendre chaque phase des mises à jour des nœuds. Si vous utilisez une AMI optimisée pour Amazon EKS, Amazon EKS applique automatiquement les derniers correctifs de sécurité et les dernières mises à jour du système d'exploitation à vos nœuds dans le cadre de la dernière version publiée de l'AMI.

Il existe plusieurs scénarios dans lesquels il est utile de mettre à jour la version ou la configuration de votre groupe de nœuds gérés Amazon EKS :

- Vous avez mis à jour la version de Kubernetes de votre cluster Amazon EKS et vous souhaitez mettre à jour vos nœuds pour utiliser la même version de Kubernetes.
- Une nouvelle version d'AMI est disponible pour votre groupe de nœuds gérés. Pour plus d'informations sur les versions d'AMI, consultez ces sections :
  - the section called "Obtenir les informations sur la version"
  - the section called "Flacon Rocket"
  - the section called "Obtenir les informations sur la version"
- Vous souhaitez ajuster le nombre minimum, maximum ou souhaité d'instances dans votre groupe de nœuds gérés.
- Vous voulez ajouter ou supprimer les labels Kubernetes des instances de votre groupe de nœuds gérés.
- Vous souhaitez ajouter ou supprimer des identifications AWS dans votre groupe de nœuds gérés.
- Vous devez déployer une nouvelle version d'un modèle de lancement avec des modifications de configuration, telles qu'une AMI personnalisée mise à jour.
- Vous avez déployé la version `1.9.0` ou ultérieure du module complémentaire CNI Amazon VPC, activé le module complémentaire pour la délégation de préfixe et voulez que les nouvelles instances AWS Nitro System d'un groupe de nœuds prennent en charge un nombre considérablement accru de pods. Pour de plus amples informations, consultez the section called "Augmentation du nombre d'adresses IP".
- Vous avez activé la délégation de préfixe IP pour les nœuds Windows et voulez que les nouvelles instances AWS Nitro System d'un groupe de nœuds prennent en charge un nombre considérablement accru de pods. Pour de plus amples informations, consultez the section called "Augmentation du nombre d'adresses IP".

S'il existe une version AMI plus récente pour la version Kubernetes de votre groupe de nœuds gérés, vous pouvez mettre à jour la version de votre groupe de nœuds afin d'utiliser la version AMI plus récente. De même, si votre cluster exécute une version de Kubernetes plus récente que celle de votre groupe de nœuds, vous pouvez mettre à jour le groupe de nœuds afin d'utiliser la dernière version AMI pour correspondre à la version Kubernetes de votre cluster.

Lorsqu'un nœud d'un groupe de nœuds gérés est résilié en raison d'une opération de mise à l'échelle ou d'une mise à jour, les pods de ce nœud sont d'abord vidés. Pour de plus amples informations, consultez the section called "Mise à jour des informations des comportements".

Mise à jour d'une version de groupe de nœuds

Vous pouvez mettre à jour la version d'un groupe de nœuds à l'aide de l'une des méthodes suivantes :

- the section called "`eksctl`"
- the section called "AWS Management Console"

La version vers laquelle vous effectuez la mise à jour ne peut pas être supérieure à la version du plan de contrôle.

`eksctl`

Mettre à jour un groupe de nœuds gérés à l'aide de **`eksctl`**

Mettez à jour un groupe de nœuds gérés vers la dernière version AMI de la même version Kubernetes actuellement déployée sur les nœuds à l'aide de la commande suivante. Remplacez la *`valeur d'exemple`* par vos propres valeurs.

```
eksctl upgrade nodegroup \
  --name=node-group-name \
  --cluster=my-cluster \
  --region=region-code
```

> **Note**
>
> Si vous mettez à niveau un groupe de nœuds déployé avec un modèle de lancement vers une nouvelle version du modèle de lancement, ajoutez `--launch-template-version` *`version-number`* à la commande précédente. Le modèle de lancement doit répondre aux exigences décrites dans Personnaliser les nœuds gérés avec des modèles de lancement. Si le modèle de lancement inclut une AMI personnalisée, celle-ci doit répondre aux exigences décrites dans Spécifier une AMI. Lorsque vous mettez à niveau votre groupe de nœuds vers une version plus récente de votre modèle de lancement, chaque nœud est recyclé pour correspondre à la nouvelle configuration de la version du modèle de lancement spécifiée.

Il n'est pas possible de mettre à niveau directement un groupe de nœuds déployé sans modèle de lancement vers une nouvelle version du modèle de lancement. À la place, vous devez déployer un nouveau groupe de nœuds en utilisant le modèle de lancement pour mettre à jour le groupe de nœuds vers une nouvelle version du modèle de lancement.

Vous pouvez mettre à niveau un groupe de nœuds vers la même version que la version Kubernetes du plan de contrôle. Par exemple, si vous disposez d'un cluster exécutant Kubernetes `1.33`, vous pouvez mettre à niveau les nœuds exécutant actuellement Kubernetes `1.32` vers la version `1.33` à l'aide de la commande suivante.

```
eksctl upgrade nodegroup \
  --name=node-group-name \
  --cluster=my-cluster \
  --region=region-code \
  --kubernetes-version=1.33
```

AWS Management Console

Mettre à jour un groupe de nœuds gérés à l'aide de la AWS Management Console

1. Ouvrez la console Amazon EKS.
2. Choisissez le cluster qui contient le groupe de nœuds à mettre à jour.
3. Si une mise à jour est disponible pour au moins un groupe de nœuds, une boîte de dialogue apparaît en haut de la page pour vous en informer. Si vous sélectionnez l'onglet Calcul, vous verrez Mettre à jour maintenant dans la colonne Version de l'AMI du tableau Groupes de nœuds pour le groupe de nœuds qui dispose d'une mise à jour disponible. Pour mettre à jour le groupe de nœuds, sélectionnez Update now (Mettre à jour maintenant).

   Vous ne verrez pas de notification pour les groupes de nœuds qui ont été déployés avec une AMI personnalisée. Si vos nœuds sont déployés avec une AMI personnalisée, effectuez les étapes suivantes pour déployer une nouvelle AMI personnalisée mise à jour.

   a. Créez une version de votre AMI.

   b. Créez une version de modèle de lancement avec le nouvel ID d'AMI.

   c. Mettez à niveau les nœuds vers la nouvelle version du modèle de lancement.
4. Dans la boîte de dialogue Update node group version (Mettre à jour la version du groupe de nœuds), activez ou désactivez les options suivantes :

- Update node group version (Mettre à jour la version du groupe de nœuds) : cette option n'est pas disponible si vous avez déployé une AMI personnalisée ou que votre AMI optimisée pour Amazon EKS correspond actuellement à la dernière version de votre cluster.
- Change launch template version (Modifier la version du modèle de lancement) : cette option n'est pas disponible si le groupe de nœuds est déployé sans modèle de lancement personnalisé. Vous pouvez uniquement mettre à jour la version du modèle de lancement pour un groupe de nœuds qui a été déployé avec un modèle de lancement personnalisé. Sélectionnez la Version du modèle de lancement vers laquelle vous souhaitez mettre à jour le groupe de nœuds. Si votre groupe de nœuds est configuré avec une AMI personnalisée, la version que vous sélectionnez doit également spécifier une AMI. Lorsque vous passez à une version plus récente de votre modèle de lancement, chaque nœud est recyclé pour correspondre à la nouvelle configuration de la version du modèle de lancement spécifiée.

5. Pour Update strategy (Politique de mise à jour), sélectionnez l'une des options suivantes :
   - Mise à jour propagée : cette option respecte les budgets de perturbation des pods pour votre cluster. Les mises à jour échouent s'il existe un problème de budget de perturbation des pods qui empêche Amazon EKS de vider correctement les pods qui s'exécutent sur ce groupe de nœuds.
   - Forcer la mise à jour : cette option ne respecte pas les budgets de perturbation des pods. Les mises à jour ont lieu indépendamment des problèmes de budget de perturbation des pods en forçant le redémarrage des nœuds.
6. Choisissez Mettre à jour.

Modification de la configuration d'un groupe de nœuds

Vous pouvez modifier une partie de la configuration d'un groupe de nœuds gérés.

1. Ouvrez la console Amazon EKS.
2. Choisissez le cluster qui contient le groupe de nœuds à modifier.
3. Sélectionnez l'onglet Compute (Calcul).
4. Sélectionnez le groupe de nœuds à modifier, puis choisissez Edit (Modifier).
5. (Facultatif) Sur la page Edit node group (Modifier le groupe de nœuds), effectuez les opérations suivantes :
   a. Modifiez la Node group scaling configuration (Configuration de mise à l'échelle du groupe de nœuds).

- Taille souhaitée : spécifiez le nombre actuel de nœuds que le groupe de nœuds gérés doit conserver au lancement.
- Taille minimale : spécifiez le nombre minimal de nœuds vers lequel le groupe de nœuds gérés peut être mis à l'échelle.
- Taille maximale : spécifiez le nombre maximal de nœuds vers lequel le groupe de nœuds gérés peut être mis à niveau. Pour connaître le nombre maximal de noeuds pris en charge dans un groupe de nœuds, consultez the section called “Quotas de service”.

b. (Facultatif) Ajoutez ou supprimez des étiquettes Kubernetes aux nœuds de votre groupe de nœuds. Les labels affichés ici sont uniquement ceux que vous avez appliqués avec Amazon EKS. D'autres étiquettes peuvent exister sur vos nœuds et ne pas être affichées ici.

c. (Facultatif) Ajoutez ou supprimez des rejets Kubernetes aux nœuds de votre groupe de nœuds. Les taints ajoutés peuvent avoir l'effet de **`NoSchedule`**, **`NoExecute`** ou **`PreferNoSchedule`**. Pour de plus amples informations, consultez the section called “Empêcher la planification de pods sur des nœuds spécifiques”.

d. (Facultatif) Ajoutez ou supprimez des Tags (Balises) de la ressource de votre groupe de nœuds. Ces identifications ne sont appliquées qu'au groupe de nœuds Amazon EKS. Ils ne se propagent pas à d'autres ressources, telles que les sous-réseaux ou les instances Amazon EC2 du groupe de nœuds.

e. (Facultatif) Modifiez la Configuration des mises à jour du groupe. Sélectionnez Number (Nombre) ou Percentage (Pourcentage).

- Nombre : sélectionnez et spécifiez le nombre de nœuds de votre groupe de nœuds pouvant être mis à jour en parallèle. Ces nœuds ne seront pas disponibles pendant la mise à jour.
- Pourcentage : sélectionnez et spécifiez le pourcentage de nœuds de votre groupe de nœuds pouvant être mis à jour en parallèle. Ces nœuds ne seront pas disponibles pendant la mise à jour. Ceci est utile si vous avez beaucoup de nœuds dans votre groupe de nœuds.

f. Une fois vos modifications terminées, sélectionnez Enregistrer les modifications.

> **Important**
>
> Lors de la mise à jour de la configuration du groupe de nœuds, la modification de `NodegroupScalingConfig` ne respecte pas les budgets de perturbation des pods (PDB). Contrairement au processus de mise à jour du groupe de nœuds (qui vide les nœuds et respecte les PDB pendant la phase de mise à niveau), la mise à jour de la configuration de mise à l'échelle entraîne le résiliation immédiate des nœuds via un appel de réduction

du groupe Auto Scaling (ASG). Cela se produit sans tenir compte des PDB, quelle que soit la taille cible vers laquelle vous réduisez. Cela signifie que lorsque vous réduisez la `desiredSize` d'un groupe de nœuds géré par Amazon EKS, les pods sont évacués dès que les nœuds sont résiliés, sans respecter les PDB.

Comprendre chaque phase des mises à jour des nœuds

La stratégie de mise à jour des composants master d'Amazon EKS comporte quatre phases différentes décrites dans les sections suivantes.

Phase de configuration

La phase de configuration comporte les étapes suivantes :

1. Une nouvelle version du modèle de lancement Amazon EC2 est créée pour le groupe Auto Scaling associé à votre groupe de nœuds. La nouvelle version du modèle de lancement utilise l'AMI cible ou une version personnalisée du modèle de lancement pour la mise à jour.
2. Le groupe Auto Scaling est mis à jour pour utiliser la dernière version du modèle de lancement.
3. La quantité maximale de nœuds à mettre à niveau en parallèle est déterminée à l'aide de la propriété `updateConfig` pour le groupe de nœuds. Le maximum indisponible a un quota de 100 nœuds. La valeur par défaut est de un nœud. Pour plus d'informations, consultez la propriété updateConfig dans la référence API Amazon EKS.

Phase d'augmentation d'échelle

Lors de la mise à niveau des nœuds d'un groupe de nœuds gérés, les nœuds mis à niveau sont lancés dans la même zone de disponibilité que ceux qui sont mis à niveau. Pour garantir ce placement, nous utilisons le rééquilibrage de la zone de disponibilité d'Amazon EC2. Pour plus d'informations, consultez Rééquilibrage de la zone de disponibilité dans le Guide de l'utilisateur Amazon EC2 Auto Scaling. Pour répondre à cette exigence, il est possible que nous lancions jusqu'à deux instances par zone de disponibilité dans votre groupe de nœuds gérés.

La phase d'augmentation d'échelle comporte les étapes suivantes :

1. Il augmente la taille maximale et la taille souhaitée du groupe Auto Scaling en fonction de la plus grande des deux valeurs suivantes :

- Jusqu'à deux fois le nombre de zones de disponibilité dans lesquelles le groupe Auto Scaling est déployé.
- Le maximum indisponible de la mise à niveau.

  Par exemple, si votre groupe de nœuds a cinq zones de disponibilité et que `maxUnavailable` est égal à un, le processus de mise à niveau peut lancer un maximum de 10 nœuds. Cependant, lorsque `maxUnavailable` est égale à 20 (ou supérieure à 10), le processus lance 20 nouveaux nœuds.

2. Après la mise à l'échelle du groupe Auto Scaling, elle vérifie si les nœuds utilisant la dernière configuration sont présents dans le groupe de nœuds. Cette étape ne réussit que si elle répond à ces critères :
   - Au moins un nouveau nœud est lancé dans chaque zone de disponibilité où le nœud existe.
   - Chaque nouveau nœud doit être dans l'état `Ready`.
   - Les nouveaux nœuds doivent avoir des étiquettes Amazon EKS appliquées.

     Il s'agit des étiquettes Amazon EKS appliquées sur les composants master d'un groupe de nœuds réguliers :

     - `eks.amazonaws.com/nodegroup-image=$amiName`
     - `eks.amazonaws.com/nodegroup=$nodeGroupName`

     Il s'agit des étiquettes appliquées par Amazon EKS sur les composants master dans un modèle de lancement personnalisé ou un groupe de composants AMI :

     - `eks.amazonaws.com/nodegroup-image=$amiName`
     - `eks.amazonaws.com/nodegroup=$nodeGroupName`
     - `eks.amazonaws.com/sourceLaunchTemplateId=$launchTemplateId`
     - `eks.amazonaws.com/sourceLaunchTemplateVersion=$launchTemplateVersion`
3. Il marque les nœuds comme non planifiables afin d'éviter la planification de nouveaux pods. Il étiquette également les nœuds avec `node.kubernetes.io/exclude-from-external-load-balancers=true` afin de supprimer les anciens nœuds des équilibreurs de charge avant de les terminer.

Les raisons suivantes sont connues pour provoquer une erreur `NodeCreationFailure` dans cette phase :

Capacité insuffisante dans la zone de disponibilité

Il est possible que la zone de disponibilité n'ait pas la capacité des types d'instance demandés. Il est recommandé de configurer plusieurs types d'instances lors de la création d'un groupe de nœuds gérés.

Limites d'instance EC2 sur votre compte

Vous pouvez être amené à augmenter le nombre d'instances Amazon EC2 que votre compte peut exécuter simultanément en utilisant Service Quotas. Pour plus d'informations, consultez EC2 Service Quotas dans le Guide de l'utilisateur Amazon Elastic Compute Cloud pour les instances Linux.

Données utilisateur personnalisées

Les données utilisateur personnalisées peuvent parfois interrompre le processus d'amorçage. Ce scénario peut conduire à ce que le `kubelet` ne démarre pas sur le nœud ou que les nœuds n'obtiennent pas les étiquettes Amazon EKS attendues. Pour de plus amples informations, consultez the section called "Spécification d'une AMI".

Toute modification qui rend un nœud défectueux ou pas prêt

La pression du disque sur le nœud, la pression de la mémoire et des conditions similaires peuvent empêcher un nœud de passer à l'état `Ready`.

Chaque nœud démarre le plus rapidement en 15 minutes

Si un nœud prend plus de 15 minutes pour démarrer et rejoindre le cluster, cela entraînera l'expiration de la mise à niveau. Il s'agit de la durée totale nécessaire au démarrage d'un nouveau nœud, mesurée à partir du moment où un nouveau nœud est requis jusqu'à son intégration au cluster. Lors de la mise à niveau d'un groupe de nœuds gérés, le compteur de temps démarre dès que la taille du groupe Auto Scaling augmente.

Phase de mise à niveau

La phase de mise à niveau se déroule de deux manières différentes, selon la stratégie de mise à jour. Il existe deux stratégies de mise à jour : par défaut et minimale.

Nous vous recommandons d'utiliser la stratégie par défaut. Il crée de nouveaux nœuds avant de mettre fin aux anciens, afin que la capacité disponible soit maintenue pendant la phase de mise à niveau. Cette stratégie minimale est utile lorsque les ressources ou les coûts sont limités, par

exemple avec des accélérateurs matériels tels que des GPU. Il met fin aux anciens nœuds avant d’en créer de nouveaux, afin que la capacité totale ne dépasse jamais la quantité que vous avez configurée.

La stratégie de mise à jour par défaut comprend les étapes suivantes :

1. Il augmente le nombre de nœuds (nombre souhaité) dans le groupe Auto Scaling, ce qui entraîne la création de nœuds supplémentaires par le groupe de nœuds.
2. Un nœud est sélectionné aléatoirement pour mise à niveau, jusqu'au maximum indisponible configuré pour le groupe de nœuds.
3. Il vide les pods du nœud. Si les pods ne quittent pas le nœud dans les 15 minutes et qu’il n’y a pas d’indicateur de force, la phase de mise à niveau échoue avec une erreur `PodEvictionFailure`. Pour ce scénario, vous pouvez appliquer l’indicateur de force avec la demande `update-nodegroup-version` de suppression des pods.
4. Il isole le nœud après chaque expulsion de pod et attend 60 secondes. Cela permet au contrôleur de services de ne pas envoyer de nouvelles requêtes à ce nœud et de le supprimer de sa liste de nœuds actifs.
5. Une demande d'arrêt est envoyée au groupe Auto Scaling pour le nœud isolé.
6. Les étapes de mise à niveau précédentes sont répétées jusqu'à ce que le groupe de nœuds ne comporte plus aucun nœud déployé avec la version antérieure du modèle de lancement.

La stratégie de mise à jour minimale comprend les étapes suivantes :

1. Il isole tous les nœuds du groupe de nœuds au début, afin que le contrôleur de service n’envoie aucune nouvelle requête à ces nœuds.
2. Un nœud est sélectionné aléatoirement pour mise à niveau, jusqu'au maximum indisponible configuré pour le groupe de nœuds.
3. Il vide les pods des nœuds sélectionnés. Si les pods ne quittent pas le nœud dans les 15 minutes et qu’il n’y a pas d’indicateur de force, la phase de mise à niveau échoue avec une erreur `PodEvictionFailure`. Pour ce scénario, vous pouvez appliquer l’indicateur de force avec la demande `update-nodegroup-version` de suppression des pods.
4. Une fois que chaque pod a été évincé et a attendu 60 secondes, il envoie une demande de résiliation au groupe Auto Scaling pour les nœuds sélectionnés. Le groupe Auto Scaling crée de nouveaux nœuds (en nombre égal à celui des nœuds sélectionnés) pour remplacer la capacité manquante.

5. Les étapes de mise à niveau précédentes sont répétées jusqu'à ce que le groupe de nœuds ne comporte plus aucun nœud déployé avec la version antérieure du modèle de lancement.

Erreurs **`PodEvictionFailure`** lors de la phase de mise à niveau

Les raisons suivantes sont connues pour provoquer une erreur `PodEvictionFailure` dans cette phase :

PDB agressif

Un PDB agressif est défini sur le pod ou plusieurs PDB pointent vers le même pod.

Déploiement tolérant tous les rejets

Une fois que tous les pods ont été évacués, le nœud devrait être vide, car il a été rejeté lors des étapes précédentes. Toutefois, si le déploiement tolère toutes les rejets, il est plus probable que le nœud ne soit pas vide, ce qui entraîne l'échec de l'éviction du pod.

Phase de réduction d'échelle

La phase de réduction d'échelle diminue la taille maximale et la taille souhaitée du groupe Auto Scaling d'une unité pour revenir aux valeurs avant le début de la mise à jour.

Si le flux de mise à jour détermine que le Cluster Autoscaler augmente le groupe de nœuds pendant la phase de réduction d'échelle du flux de travail, il se termine immédiatement sans ramener le groupe de nœuds à sa taille d'origine.

Personnaliser les nœuds gérés à l'aide de modèles de lancement

Pour un niveau de personnalisation maximal, vous pouvez déployer des nœuds gérés avec votre propre modèle de lancement en suivant les étapes décrites sur cette page. L'utilisation d'un modèle de lancement permet des fonctionnalités telles que la fourniture d'arguments d'amorçage lors du déploiement d'un nœud (par exemple, des arguments kubelet supplémentaires), l'attribution d'adresses IP aux pods à partir d'un bloc CIDR différent de l'adresse IP attribuée au nœud, le déploiement de votre propre AMI personnalisée sur les nœuds ou le déploiement de votre propre CNI personnalisé sur les nœuds.

Si vous fournissez votre propre modèle de lancement lors de la création en premier lieu d'un groupe de nœuds gérés, vous bénéficierez également d'une plus grande flexibilité par la suite. Pourvu que

vous déployiez un groupe de nœuds gérés avec votre propre modèle de lancement, vous pouvez le mettre à jour de façon itérative avec une version différente du même modèle de lancement. Lorsque vous mettez à jour votre groupe de nœuds vers une version différente de votre modèle de lancement, tous les nœuds du groupe sont recyclés pour correspondre à la nouvelle configuration de la version spécifiée du modèle de lancement.

Les groupes de nœuds gérés sont toujours déployés avec un modèle de lancement à utiliser avec le groupe Amazon EC2 Auto Scaling. Si vous ne fournissez pas de modèle de lancement, l'API Amazon EKS en crée automatiquement un avec les valeurs par défaut de votre compte. Cependant, il est déconseillé de modifier les modèles de lancement générés automatiquement. De plus, les groupes de nœuds existants qui n'utilisent pas de modèle de lancement personnalisé ne peuvent pas être mis à jour directement. Vous devez plutôt créer un nouveau groupe de nœuds avec un modèle de lancement personnalisé à cet effet.

Concepts de base de la configuration d'un modèle de lancement

Vous pouvez créer un modèle de lancement Amazon EC2 Auto Scaling à l'aide de AWS Management Console la AWS CLI ou d' AWS un SDK. Pour plus d'informations, consultez Création d'un modèle de lancement pour un groupe Auto Scaling dans le Guide de l'utilisateur Amazon EC2 Auto Scaling. Certains des paramètres d'un modèle de lancement sont similaires aux paramètres utilisés pour la configuration des nœuds gérés. Lors du déploiement ou de la mise à jour d'un groupe de nœuds avec un modèle de lancement, certains paramètres doivent être spécifiés soit dans la configuration du groupe de nœuds, soit dans le modèle de lancement. Veuillez ne pas spécifier un paramètre aux deux endroits. Si un paramètre existe là où il ne devrait pas, les opérations telles que la création ou la mise à jour d'un groupe de nœuds échoueront.

Le tableau suivant répertorie les paramètres interdits dans un modèle de lancement. Il répertorie également les paramètres similaires, s'il en existe, qui sont requis dans la configuration du groupe de nœuds géré. Les paramètres répertoriés sont les paramètres qui apparaissent dans la console. Ils peuvent avoir des noms similaires mais différents dans la AWS CLI et le SDK.

Modèle de lancement – Interdit	Configuration d'un groupe de nœuds Amazon EKS
Sous-réseau sous Interfaces réseau (Ajouter une interface réseau)	Sous-réseaux sous Configuration réseau du groupe de nœuds sur la page Spécifier le réseau

Modèle de lancement – Interdit	Configuration d'un groupe de nœuds Amazon EKS
Profil d'instance IAM sous Détails avancés	Rôle IAM de nœud sous Configuration du groupe de nœuds sur la page Configurer le groupe de nœuds
Comportement d'arrêt et Arrêt – Comportement de mise en veille prolongée sous Détails avancés. Conservez le paramètre par défaut Ne pas inclure dans le modèle de lancement dans le modèle de lancement pour les deux paramètres.	Pas d'équivalent. Amazon EKS doit contrôler le cycle de vie de l'instance et non pas le groupe Auto Scaling.

Le tableau suivant répertorie les paramètres interdits dans une configuration de groupe de nœuds gérée. Il répertorie également les paramètres similaires, s'il en existe, qui sont requis dans un modèle de lancement. Les paramètres répertoriés sont les paramètres qui apparaissent dans la console. Ils peuvent porter des noms similaires dans la AWS CLI et le SDK.

Configuration du groupe de nœuds Amazon EKS : Interdite	Modèle de lancement
(Seulement si vous avez spécifié une AMI personnalisée dans un modèle de lancement) AMI type (Type d'AMI) sous Node Group compute configuration (Configuration du calcul du groupe de nœuds) sur la page Set compute and scaling configuration (Définir la configuration de calcul et de mise à l'échelle) : la console affiche Specified in launch template (Spécifié dans le modèle de lancement) et l'ID d'AMI spécifié. Si Images d'application et de système d'exploitation (Amazon Machine Image) n'a pas été spécifié dans le modèle de lancement, vous	Images d'applications et de systèmes d'exploitation (Amazon Machine Image) sous Contenu du modèle de lancement : vous devez spécifier un ID dans l'un des cas suivants : • Utilisation d'une image AMI personnalisée Si vous spécifiez une AMI qui ne répond pas aux exigences répertoriées dans Spécification d'une AMI, le déploiement du groupe de nœuds échouera. • Veut fournir des données utilisateur pour fournir des arguments au fichier `bootstrap.sh` inclus avec un AMI optimisé pour Amazon EKS. Vous pouvez activer vos

Configuration du groupe de nœuds Amazon EKS : Interdite	Modèle de lancement
pouvez sélectionner une AMI dans la configuration du groupe de nœuds.	instances pour attribuer un nombre nettement plus élevé d'adresses IP aux pods, attribuer des adresses IP aux pods à partir d'un bloc CIDR différent de celui de l'instance ou déployer un cluster privé sans accès Internet sortant. Pour plus d'informations, consultez les rubriques suivantes : • Attribuer davantage d'adresses IP aux nœuds Amazon EKS avec des préfixes • Déployer des pods dans des sous-réseaux alternatifs avec une mise en réseau personnalisée • Déployer des clusters privés avec un accès Internet limité • Spécification d'une AMI
Taille du disque sous Configuration de calcul du groupe de nœuds sur la page Définir la configuration de calcul et de mise à l'échelle : la console affiche Spécifié dans le modèle de lancement.	Taille sous Stockage (volumes) (Ajouter un nouveau volume). Vous devez le spécifier dans le modèle de lancement.
Paire de clés SSH sous Configuration du groupe de nœuds sur la page Spécifier le réseau : la console affiche la clé spécifiée dans le modèle de lancement ou Non spécifié dans le modèle de lancement.	Nom de la paire de clés sous Paire de clés (connexion).

Configuration du groupe de nœuds Amazon EKS : Interdite	Modèle de lancement
Vous ne pouvez pas spécifier de groupes de sécurité source autorisés à accéder à distance lorsque vous utilisez un modèle de lancement.	Groupes de sécurité sous Paramètres réseau pour l'instance ou Groupes de sécurité sous Interfaces réseau (Ajouter une interface réseau), mais pas les deux. Pour de plus amples informations, veuillez consulter the section called "Utilisation des groupes de sécurité".

Note

- Si vous déployez un groupe de nœuds à l'aide d'un modèle de lancement, spécifiez zéro ou un type d'instance sous Contenu du modèle de lancement dans un modèle de lancement. Vous pouvez également spécifier de 0 à 20 types d'instance sous types d'instance sur la page Définir la configuration de calcul et de mise à l'échelle dans la console. Vous pouvez également le faire à l'aide d'autres outils qui utilisent l'API Amazon EKS. Si vous spécifiez un type d'instance dans un modèle de lancement et que vous utilisez ce modèle pour déployer votre groupe de nœuds, vous ne pouvez spécifier aucun type d'instance dans la console ou à l'aide d'autres outils qui utilisent l'API Amazon EKS. Si vous ne spécifiez pas de type d'instance dans un modèle de lancement, dans la console ou à l'aide d'autres outils utilisant l'API Amazon EKS, le type d'instance `t3.medium` est utilisé. Si votre groupe de nœuds utilise le type de capacité Spot, nous vous recommandons de spécifier plusieurs types d'instance à l'aide de la console. Pour de plus amples informations, veuillez consulter the section called "Types de capacité des groupes de nœuds gérés".
- Si les conteneurs que vous déployez dans le groupe de nœuds utilisent le service de métadonnées d'instance version 2, veillez à définir 2 comme limite de saut de réponse des métadonnées dans votre modèle de lancement. Pour plus d'informations, consultez Métadonnées d'instance et données utilisateur dans le Guide de l'utilisateur Amazon EC2.
- Les modèles de lancement ne prennent pas en charge la fonctionnalité `InstanceRequirements` qui permet une sélection flexible du type d'instance.

Étiquetage des instances Amazon EC2

Vous pouvez utiliser le paramètre `TagSpecification` d'un modèle de lancement pour spécifier les identifications à appliquer aux instances Amazon EC2 dans votre groupe de nœuds. L'entité IAM qui appelle le `CreateNodegroup` ou `UpdateNodegroupVersion` APIs doit disposer des autorisations pour `ec2:RunInstances` et`ec2:CreateTags`, et les balises doivent être ajoutées au modèle de lancement.

Utilisation des groupes de sécurité

Vous pouvez utiliser un modèle de lancement pour spécifier des groupes de sécurité Amazon EC2 à appliquer aux instances de votre groupe de nœuds. Vous pouvez le faire soit dans le paramètre des groupes de sécurité au niveau de l'instance, soit dans le cadre des paramètres de configuration de l'interface réseau. Cependant, vous ne pouvez pas créer un modèle de lancement qui spécifie à la fois le niveau d'instance et les groupes de sécurité de l'interface réseau. Prenez en compte les conditions suivantes qui s'appliquent à l'utilisation de groupes de sécurité personnalisés avec des groupes de nœuds gérés :

- Lorsque vous utilisez le AWS Management Console, Amazon EKS n'autorise que les modèles de lancement dotés d'une seule spécification d'interface réseau.
- Par défaut, Amazon EKS applique le groupe de sécurité du cluster aux instances de votre groupe de nœuds pour faciliter la communication entre les nœuds et le plan de contrôle. Si vous spécifiez des groupes de sécurité personnalisés dans le modèle de lancement à l'aide de l'une des options mentionnées précédemment, Amazon EKS n'ajoute pas le groupe de sécurité du cluster. Vous devez donc vous assurer que les règles d'entrée et de sortie de vos groupes de sécurité permettent la communication avec le point de terminaison de votre cluster. Si vos règles de groupe de sécurité sont incorrectes, les composants master ne peuvent pas rejoindre le cluster. Pour plus d'informations sur les règles de groupe de sécurité, consultez the section called “Exigences de groupe de sécurité”.
- Si vous avez besoin d'un accès SSH aux instances de votre groupe de nœuds, incluez un groupe de sécurité qui autorise cet accès.

Données utilisateur Amazon EC2

Le modèle de lancement comprend une section pour les données utilisateur personnalisées. Vous pouvez définir les paramètres de configuration de votre groupe de nœuds dans cette section sans créer manuellement une personnalisation individuelle AMIs. Pour plus d'informations sur les

paramètres disponibles pour Bottlerocket, consultez la section Utilisation des données utilisateur sur. GitHub

Vous pouvez fournir des données d'utilisateur Amazon EC2 dans votre modèle de lancement en utilisant `cloud-init` lors du lancement de vos instances. Pour plus d'informations, consultez la documentation cloud-init. Vos données utilisateur peuvent être utilisées pour effectuer des opérations de configuration courantes. Elles comprennent :

- Inclusion d'utilisateurs ou de groupes
- Installations de packages

Les données utilisateur Amazon EC2 figurant dans les modèles de lancement utilisés avec des groupes de nœuds gérés doivent être au format d'archive en plusieurs parties MIME pour Amazon Linux AMIs et au format TOML pour Bottlerocket. AMIs En effet, vos données utilisateur sont fusionnées avec les données utilisateur d'Amazon EKS requises pour que les nœuds puissent rejoindre le cluster. Ne spécifiez aucune commande dans vos données utilisateur qui démarre ou modifie `kubelet`. Ceci est effectué dans le cadre des données utilisateur fusionnées par Amazon EKS. Certains paramètres `kubelet`, tels que la définition des étiquettes sur les nœuds, peuvent être configurés directement via l'API des groupes de nœuds gérés.

> **Note**
>
> Pour plus d'informations sur la personnalisation avancée de `kubelet`, y compris son démarrage manuel ou le passage de paramètres de configuration personnalisés, consultez the section called "Spécification d'une AMI". Si un ID d'AMI personnalisé est spécifié dans un modèle de lancement, Amazon EKS ne fusionne pas les données utilisateur.

Les détails suivants fournissent plus d'informations sur la section des données utilisateur.

Données utilisateur Amazon Linux 2

Vous pouvez combiner plusieurs blocs de données utilisateur dans un seul fichier MIME multi-part. Par exemple, vous pouvez combiner un boothook cloud qui configure le démon Docker avec un script shell de données utilisateur qui installe un package personnalisé. Un fichier MIME multi-part est constitué des composants suivants :

- La déclaration de type de contenu et de limite de partie : `Content-Type: multipart/mixed; boundary="==MYBOUNDARY=="`

- La déclaration de version MIME : `MIME-Version: 1.0`
- Un ou plusieurs blocs de données qui contiennent les composants suivants :
  - La limite de début qui signale le début d'un bloc de données utilisateur : `--==MYBOUNDARY==`
  - La déclaration de type de contenu du bloc : `Content-Type: text/cloud-config; charset="us-ascii"`. Pour plus d'informations sur les types de contenus, consultez la documentation sur Cloud-Init.
  - Le contenu des données utilisateur (par exemple, une liste de commandes shell ou de directives `cloud-init`).
  - La limite de fin qui signale la fin du fichier MIME multi-part : `--==MYBOUNDARY==--`

  Voici un exemple de fichier MIME multi-part que vous pouvez utiliser pour créer le vôtre.

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="==MYBOUNDARY=="

--==MYBOUNDARY==
Content-Type: text/x-shellscript; charset="us-ascii"

#!/bin/bash
echo "Running custom user data script"

--==MYBOUNDARY==--
```

Données utilisateur Amazon Linux 2023

Amazon Linux 2023 (AL2023) introduit un nouveau processus d'initialisation des nœuds `nodeadm` qui utilise un schéma de configuration YAML. Si vous utilisez des groupes de nœuds autogérés ou une AMI avec un modèle de lancement, vous devez désormais fournir explicitement des métadonnées de cluster supplémentaires lors de la création d'un nouveau groupe de nœuds. Un exemple des paramètres minimum requis est présenté ci-dessous, où `apiServerEndpoint`, `certificateAuthority` et le service `cidr` sont désormais obligatoires :

```
---
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
    name: my-cluster
```

```
    apiServerEndpoint: https://example.com
    certificateAuthority: Y2VydGlmaWNhdGVBdXRob3JpdHk=
    cidr: 10.100.0.0/16
```

Vous définirez généralement cette configuration dans vos données utilisateur, soit telle quelle, soit intégrée dans un document MIME multipart :

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="BOUNDARY"

--BOUNDARY
Content-Type: application/node.eks.aws

---
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig spec: [...]

--BOUNDARY--
```

Dans AL2, les métadonnées de ces paramètres ont été découvertes à partir de l'appel d'`DescribeCluster`API Amazon EKS. Avec AL2023, ce comportement a changé car l'appel d'API supplémentaire risque d'être limité lors de la mise à l'échelle de nœuds à grande échelle. Cette modification ne vous affecte pas si vous utilisez des groupes de nœuds gérés sans modèle de lancement, ou si vous utilisez Karpenter. Pour plus d'informations sur `certificateAuthority` et le service `cidr`, consultez `DescribeCluster` dans la Référence de l'API Amazon EKS.

Voici un exemple complet de données AL2023 utilisateur qui combine un script shell pour personnaliser le nœud (comme l'installation de packages ou la pré-mise en cache des images de conteneur) avec la configuration requise. `nodeadm` Cet exemple présente des personnalisations courantes, notamment : * L'installation de paquets système supplémentaires * La mise en cache préalable des images de conteneur pour améliorer le temps de démarrage des pods * La configuration du proxy HTTP * La configuration des métriques `kubelet` pour l'étiquetage des nœuds

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="BOUNDARY"

--BOUNDARY
Content-Type: text/x-shellscript; charset="us-ascii"
```

```
#!/bin/bash
set -o errexit
set -o pipefail
set -o nounset

# Install additional packages
yum install -y htop jq iptables-services

# Pre-cache commonly used container images
nohup docker pull public.ecr.aws/eks-distro/kubernetes/pause:3.2 &

# Configure HTTP proxy if needed
cat > /etc/profile.d/http-proxy.sh << 'EOF'
export HTTP_PROXY="http://proxy.example.com:3128"
export HTTPS_PROXY="http://proxy.example.com:3128"
export NO_PROXY="localhost,127.0.0.1,169.254.169.254,.internal"
EOF

--BOUNDARY
Content-Type: application/node.eks.aws

apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
    name: my-cluster
    apiServerEndpoint: https://example.com
    certificateAuthority: Y2VydGlmaWNhdGVBdXRob3JpdHk=
    cidr: 10.100.0.0/16
  kubelet:
    config:
      clusterDNS:
      - 10.100.0.10
    flags:
    - --node-labels=app=my-app,environment=production

--BOUNDARY--
```

Données utilisateur Bottlerocket

Bottlerocket structure les données utilisateur au format TOML. Vous pouvez fournir des données utilisateur qui seront fusionnées avec les données utilisateur fournies par Amazon EKS. Par exemple, vous pouvez fournir des paramètres `kubelet` supplémentaires.

```
[settings.kubernetes.system-reserved]
cpu = "10m"
memory = "100Mi"
ephemeral-storage= "1Gi"
```

Pour plus d'informations sur les paramètres pris en charge, consultez la documentation Bottlerocket. Vous pouvez configurer des étiquettes de nœuds et des rejets dans vos données utilisateur. Cependant, nous vous recommandons de les configurer plutôt dans votre groupe de nœuds. Amazon EKS applique ces configurations lorsque vous le faites.

Lorsque les données utilisateur sont fusionnées, la mise en forme n'est pas conservée, mais le contenu reste le même. La configuration que vous fournissez dans vos données utilisateur remplace tous les paramètres qui sont configurés par Amazon EKS. Ainsi, si vous définissez `settings.kubernetes.max-pods` ou `settings.kubernetes.cluster-dns-ip`, ces valeurs dans vos données utilisateur sont appliquées aux nœuds.

Amazon EKS ne prend pas en charge tous les TOML valides. Voici une liste des formats connus non pris en charge :

- Guillemets dans les clés entre guillemets : `'quoted "value"' = "value"`
- Guillemets échappés dans les valeurs : `str = "I'm a string. \"You can quote me \""`
- Flottants et entiers mélangés : `numbers = [ 0.1, 0.2, 0.5, 1, 2, 5 ]`
- Types mixtes dans les tableaux : `contributors = ["foo@example.com", { name = "Baz", email = "baz@example.com" }]`
- En-têtes entre parenthèses avec clés entre guillemets : `[foo."bar.baz"]`

Données utilisateur Windows

Les données utilisateur Windows utilisent des PowerShell commandes. Lorsque vous créez un groupe de nœuds gérés, vos données utilisateur personnalisées sont combinées avec les données utilisateur gérées par Amazon EKS. Vos PowerShell commandes viennent en

premier, suivies des commandes de gestion des données utilisateur, le tout dans une seule `<powershell></powershell>` balise.

> **Important**
>
> Lors de la création de groupes de nœuds Windows, Amazon EKS met à jour le `aws-auth ConfigMap` pour permettre aux nœuds Linux de rejoindre le cluster. Le service ne configure pas automatiquement les autorisations pour Windows AMIs. Si vous utilisez des nœuds Windows, vous devrez gérer l'accès via l'API d'entrée d'accès ou en mettant à jour directement le `aws-auth ConfigMap`. Pour de plus amples informations, veuillez consulter the section called "Activer la prise en charge de Windows".

> **Note**
>
> Lorsque aucun ID d'AMI n'est spécifié dans le modèle de lancement, n'utilisez pas le script Windows Amazon EKS Bootstrap dans les données utilisateur pour configurer Amazon EKS.

Voici un exemple de données utilisateur.

```
<powershell>
Write-Host "Running custom user data script"
</powershell>
```

Spécification d'une AMI

Si vous avez l'une des conditions suivantes, spécifiez un ID d'AMI dans le champ `ImageId` de votre modèle de lancement. Sélectionnez votre besoin d'informations supplémentaires.

Fournissez des données utilisateur pour transmettre des arguments au **`bootstrap.sh`** fichier inclus dans une Linux/Bottlerocket AMI optimisée pour Amazon EKS

L'amorçage est un terme utilisé pour décrire l'ajout de commandes pouvant être exécutées au démarrage d'une instance. Par exemple, la mise en place initiale permet d'utiliser des arguments kubelet supplémentaires. Vous pouvez transmettre les arguments au script `bootstrap.sh` en

utilisant `eksctl` sans spécifier de modèle de lancement. Vous pouvez également le faire en spécifiant les informations dans la section des données utilisateur d'un modèle de lancement.

eksctl sans spécifier de modèle de lancement

Créez un fichier nommé *`my-nodegroup.yaml`* avec les contenus suivants. Remplacez chaque *`example value`* par vos propres valeurs. Les arguments `--apiserver-endpoint`, `--b64-cluster-ca` et `--dns-cluster-ip` sont facultatifs. Cependant, leur définition permet au script `bootstrap.sh` d'éviter de passer un appel `describeCluster`. Ceci est utile dans les configurations de clusters privés ou les clusters où vous effectuez fréquemment des opérations de mise à l'échelle et de réduction des nœuds. Pour plus d'informations sur le `bootstrap.sh` script, consultez le fichier bootstrap.sh sur GitHub.

- Le seul argument requis est le nom du cluster (*`my-cluster`*).
- Pour récupérer un ID d'AMI optimisé pour `ami-`*`1234567890abcdef0`* , consultez les sections suivantes :
  - Récupérez l'AMI Amazon Linux recommandée IDs
  - Récupérez l'AMI Bottlerocket recommandée IDs
  - Récupérez l'AMI Microsoft Windows recommandée IDs
- Pour récupérer le *`certificate-authority`* de votre cluster, exécutez la commande suivante.

```
aws eks describe-cluster --query "cluster.certificateAuthority.data" --output text
 --name my-cluster --region region-code
```

- Pour récupérer le *`api-server-endpoint`* de votre cluster, exécutez la commande suivante.

```
aws eks describe-cluster --query "cluster.endpoint" --output text --name my-
cluster --region region-code
```

- La valeur de `--dns-cluster-ip` est votre CIDR de service avec `.10` à la fin. Pour récupérer le *`service-cidr`* de votre cluster, exécutez la commande suivante. Par exemple, si la valeur renvoyée est `ipv4 10.100.0.0/16`, votre valeur est *`10.100.0.10`*.

```
aws eks describe-cluster --query "cluster.kubernetesNetworkConfig.serviceIpv4Cidr"
 --output text --name my-cluster --region region-code
```

- Cet exemple fournit un argument `kubelet` supplémentaire pour définir une valeur `max-pods` personnalisée à l'aide du script `bootstrap.sh` inclus dans l'AMI optimisée pour Amazon EKS.

Le nom du groupe de nœuds ne peut pas dépasser 63 caractères. Il doit commencer par une lettre ou un chiffre, mais peut également inclure des tirets et des traits de soulignement pour les autres caractères. Pour obtenir de l'aide sur la sélection de *my-max-pods-value*, consultez the section called "Nombre maximal de pods recommandé par Amazon EKS pour chaque type d'instance Amazon EC2". Pour plus d'informations sur le mode `maxPods` de détermination lors de l'utilisation de groupes de nœuds gérés, consultezthe section called "Comment est déterminé MaxPods".

```
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-cluster
  region: region-code

managedNodeGroups:
  - name: my-nodegroup
    ami: ami-1234567890abcdef0
    instanceType: m5.large
    privateNetworking: true
    disableIMDSv1: true
    labels: { x86-al2-specified-mng }
    overrideBootstrapCommand: |
      #!/bin/bash
      /etc/eks/bootstrap.sh my-cluster \
        --b64-cluster-ca certificate-authority \
        --apiserver-endpoint api-server-endpoint \
        --dns-cluster-ip service-cidr.10 \
        --kubelet-extra-args '--max-pods=my-max-pods-value' \
        --use-max-pods false
```

Pour chaque option de fichier `eksctl config` disponible, consultez Schéma de fichier de configuration dans la documentation `eksctl`. L'utilitaire `eksctl` crée toujours un modèle de lancement pour vous et remplit ses données utilisateur avec les données que vous fournissez dans le fichier `config`.

Créez un groupe de nœuds avec la commande suivante.

```
eksctl create nodegroup --config-file=my-nodegroup.yaml
```

Données utilisateur dans un modèle de lancement

Spécifiez les informations suivantes dans la section des données utilisateur de votre modèle de lancement. Remplacez chaque *example value* par vos propres valeurs. Les arguments `--apiserver-endpoint`, `--b64-cluster-ca` et `--dns-cluster-ip` sont facultatifs. Cependant, leur définition permet au script `bootstrap.sh` d'éviter de passer un appel `describeCluster`. Ceci est utile dans les configurations de clusters privés ou les clusters où vous effectuez fréquemment des opérations de mise à l'échelle et de réduction des nœuds. Pour plus d'informations sur le `bootstrap.sh` script, consultez le fichier bootstrap.sh sur GitHub.

- Le seul argument requis est le nom du cluster (*my-cluster*).
- Pour récupérer le *certificate-authority* de votre cluster, exécutez la commande suivante.

```
aws eks describe-cluster --query "cluster.certificateAuthority.data" --output text
 --name my-cluster --region region-code
```

- Pour récupérer le *api-server-endpoint* de votre cluster, exécutez la commande suivante.

```
aws eks describe-cluster --query "cluster.endpoint" --output text --name my-
cluster --region region-code
```

- La valeur de `--dns-cluster-ip` est votre CIDR de service avec `.10` à la fin. Pour récupérer le *service-cidr* de votre cluster, exécutez la commande suivante. Par exemple, si la valeur renvoyée est `ipv4 10.100.0.0/16`, votre valeur est *10.100.0.10*.

```
aws eks describe-cluster --query "cluster.kubernetesNetworkConfig.serviceIpv4Cidr"
 --output text --name my-cluster --region region-code
```

- Cet exemple fournit un argument `kubelet` supplémentaire pour définir une valeur `max-pods` personnalisée à l'aide du script `bootstrap.sh` inclus dans l'AMI optimisée pour Amazon EKS. Pour obtenir de l'aide sur la sélection de *my-max-pods-value*, consultez the section called "Nombre maximal de pods recommandé par Amazon EKS pour chaque type d'instance Amazon EC2". Pour plus d'informations sur le mode `maxPods` de détermination lors de l'utilisation de groupes de nœuds gérés, consultezthe section called "Comment est déterminé MaxPods".

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="==MYBOUNDARY=="

--==MYBOUNDARY==
```

```
Content-Type: text/x-shellscript; charset="us-ascii"

#!/bin/bash
set -ex
/etc/eks/bootstrap.sh my-cluster \
  --b64-cluster-ca certificate-authority \
  --apiserver-endpoint api-server-endpoint \
  --dns-cluster-ip service-cidr.10 \
  --kubelet-extra-args '--max-pods=my-max-pods-value' \
  --use-max-pods false

--==MYBOUNDARY==--
```

Fournissez des données utilisateur pour transmettre des arguments au fichier **`Start-EKSBootstrap.ps1`** inclus dans une AMI Windows optimisée pour Amazon EKS

L'amorçage est un terme utilisé pour décrire l'ajout de commandes pouvant être exécutées au démarrage d'une instance. Vous pouvez transmettre les arguments au script `Start-EKSBootstrap.ps1` en utilisant `eksctl` sans spécifier de modèle de lancement. Vous pouvez également le faire en spécifiant les informations dans la section des données utilisateur d'un modèle de lancement.

Si vous voulez spécifier un ID d'AMI Windows personnalisé, tenez compte des considérations suivantes :

- Vous devez utiliser un modèle de lancement et fournir les commandes d'amorçage requises dans la section des données utilisateur. Pour récupérer l'ID Windows souhaité, vous pouvez utiliser le tableau de la section Créer des nœuds avec Windows optimisé AMIs.
- Il existe plusieurs limites et conditions. Par exemple, vous devez ajouter des éléments `eks:kube-proxy-windows` à votre carte de configuration AWS IAM Authenticator. Pour de plus amples informations, veuillez consulter the section called "Limites et conditions lors de la spécification d'un ID d'AMI".

Spécifiez les informations suivantes dans la section des données utilisateur de votre modèle de lancement. Remplacez chaque *`example value`* par vos propres valeurs. Les arguments `-APIServerEndpoint`, `-Base64ClusterCA` et `-DNSClusterIP` sont facultatifs. Cependant, leur définition permet au script `Start-EKSBootstrap.ps1` d'éviter de passer un appel `describeCluster`.

- Le seul argument requis est le nom du cluster (*my-cluster*).
- Pour récupérer le *certificate-authority* de votre cluster, exécutez la commande suivante.

```
aws eks describe-cluster --query "cluster.certificateAuthority.data" --output text --
name my-cluster --region region-code
```

- Pour récupérer le *api-server-endpoint* de votre cluster, exécutez la commande suivante.

```
aws eks describe-cluster --query "cluster.endpoint" --output text --name my-cluster
 --region region-code
```

- La valeur de `--dns-cluster-ip` est votre CIDR de service avec `.10` à la fin. Pour récupérer le *service-cidr* de votre cluster, exécutez la commande suivante. Par exemple, si la valeur renvoyée est `ipv4 10.100.0.0/16`, votre valeur est *10.100.0.10*.

```
aws eks describe-cluster --query "cluster.kubernetesNetworkConfig.serviceIpv4Cidr" --
output text --name my-cluster --region region-code
```

- Pour des arguments supplémentaires, consultez the section called "Paramètres de configuration du script d'amorçage".

> **Note**
>
> Si vous utilisez un CIDR de service personnalisé, vous devez le spécifier à l'aide du paramètre `-ServiceCIDR`. Sinon, la résolution DNS pour les pods dans le cluster échouera.

```
<powershell>
[string]$EKSBootstrapScriptFile = "$env:ProgramFiles\Amazon\EKS\Start-EKSBootstrap.ps1"
& $EKSBootstrapScriptFile -EKSClusterName my-cluster `
  -Base64ClusterCA certificate-authority `
  -APIServerEndpoint api-server-endpoint `
  -DNSClusterIP service-cidr.10
</powershell>
```

Exécution d'une AMI personnalisée en raison d'exigences spécifiques en matière de sécurité, de conformité ou de politique interne.

Pour plus d'informations, consultez Amazon Machine Images (AMI) dans le Guide de l'utilisateur Amazon EC2. La spécification de création d'AMI Amazon EKS contient des ressources et des scripts de configuration pour créer une AMI Amazon EKS personnalisée basée sur Amazon Linux. Pour plus d'informations, consultez Spécifications de création d'AMI Amazon EKS sur GitHub. Pour créer une AMIs installation personnalisée avec d'autres systèmes d'exploitation, consultez Amazon EKS Sample Custom AMIs on GitHub.

Vous ne pouvez pas utiliser de références de paramètres dynamiques pour les AMI IDs dans les modèles de lancement utilisés avec les groupes de nœuds gérés.

⚠ Important

- Lorsque vous spécifiez une AMI, Amazon EKS ne valide pas la version de Kubernetes intégrée à votre AMI par rapport à la version du plan de contrôle de votre cluster. Il vous incombe de vous assurer que la version Kubernetes de votre AMI personnalisée est conforme à la politique de distorsion des versions de Kubernetes :
  - La `kubelet` version de vos nœuds ne doit pas être plus récente que la version de votre cluster
  - La `kubelet` version sur vos nœuds doit être égale ou supérieure à 3 versions mineures par rapport à la version de votre cluster (pour la version de Kubernetes `1.28` ou supérieure), ou jusqu'à 2 versions mineures par rapport à la version de votre cluster (pour la version de Kubernetes ou inférieure) `1.27`

    La création de groupes de nœuds gérés présentant des violations de distorsion de version peut entraîner :
  - Les nœuds ne parviennent pas à rejoindre le cluster
  - Comportement non défini ou incompatibilités d'API
  - Instabilité du cluster ou défaillances de charge de travail
- Lorsque vous spécifiez une AMI, Amazon EKS ne fusionne aucune donnée utilisateur. C'est à vous qu'il incombe de fournir les commandes `bootstrap` requises pour que les nœuds rejoignent le cluster. Si vos nœuds ne parviennent pas à joindre le cluster, les actions `CreateNodegroup` et `UpdateNodegroupVersion` Amazon EKS échouent également.

Limites et conditions lors de la spécification d'un ID d'AMI

Voici les limites et les conditions impliquées dans la spécification d'un ID d'AMI avec des groupes de nœuds gérés :

- Vous devez créer un groupe de nœuds pour passer de la spécification d'un ID d'AMI dans un modèle de lancement à la non-spécification d'un ID d'AMI.
- Vous n'êtes pas notifié dans la console lorsqu'une version plus récente de l'AMI est disponible. Pour mettre à jour votre groupe de nœuds vers une version d'AMI plus récente, vous devez créer une nouvelle version de votre modèle de lancement avec un ID d'AMI mis à jour. Vous devez ensuite mettre à jour le groupe de nœuds avec la nouvelle version du modèle de lancement.
- Les champs suivants ne peuvent pas être définis dans l'API si vous spécifiez un ID d'AMI :
  - `amiType`
  - `releaseVersion`
  - `version`
- Tous les `taints` définis dans l'API sont appliqués de manière asynchrone si vous spécifiez un identifiant d'AMI. Pour appliquer des rejets avant l'ajout d'un nœud au cluster, vous devez les transférer à `kubelet` dans vos données utilisateur à l'aide de l'indicateur de ligne de commande `--register-with-taints`. Pour plus d'informations, consultez kubelet dans la documentation Kubernetes.
- Lorsque vous spécifiez un ID AMI personnalisé pour les groupes de nœuds gérés par Windows, ajoutez-le `eks:kube-proxy-windows` à votre carte de configuration AWS IAM Authenticator. Cela est nécessaire pour le bon fonctionnement du DNS.
  a. Ouvrez la carte de configuration de l'authentificateur AWS IAM pour la modifier.

```
kubectl edit -n kube-system cm aws-auth
```

  b. Ajoutez cette entrée à la liste `groups` sous chaque `rolearn` associé aux nœuds Windows. Votre carte de configuration doit ressembler à aws-auth-cm-windows.yaml.

```
- eks:kube-proxy-windows
```

  c. Enregistrez le fichier et quittez votre éditeur de texte.
- Pour toute AMI utilisant un modèle de lancement personnalisé, la valeur par défaut `HttpPutResponseHopLimit` pour les groupes de nœuds gérés est définie sur 2.

Supprimer un groupe de nœuds gérés de votre cluster

Cette rubrique explique comment supprimer un groupe de nœuds gérés Amazon EKS. Lorsque vous supprimez un groupe de nœuds gérés, Amazon EKS définit d'abord 0 pour les tailles minimale, maximale et souhaitée de votre groupe Auto Scaling. Cela entraîne ensuite une réduction d'échelle de votre groupe de nœuds.

Avant de mettre fin à chaque instance, Amazon EKS envoie un signal pour vider ce nœud. Pendant le processus de vidange, Kubernetes effectue les opérations suivantes pour chaque pod du nœud : exécute tous les hooks de `preStop` cycle de vie configurés, envoie `SIGTERM` des signaux aux conteneurs, puis attend l'`terminationGracePeriodSeconds`arrêt progressif. Si le nœud n'a pas été vidé au bout de 5 minutes, Amazon EKS permet à Auto Scaling de poursuivre la mise hors service forcée de l'instance. Une fois que toutes les instances ont été mises hors service, le groupe Auto Scaling est supprimé.

> **Important**
>
> Si vous supprimez un groupe de nœuds gérés qui utilise un rôle IAM de nœud qui n'est utilisé par aucun autre groupe de nœuds gérés dans le cluster, le rôle est supprimé de la `ConfigMap` `aws-auth`. Si l'un des groupes de nœuds autogérés dans le cluster utilisent le même rôle IAM de nœud, le statut des nœuds autogérés devient `NotReady`. De plus, le fonctionnement du cluster est également perturbé. Pour ajouter un mappage pour le rôle que vous utilisez uniquement pour les groupes de nœuds autogérés, consultez the section called "Créer des entrées d'accès", si la version de plateforme de votre cluster est au moins égale à la version minimale indiquée dans la section Conditions préalables de l'article Accorder aux utilisateurs IAM l'accès à Kubernetes avec des entrées d'accès EKS. Si la version de votre plateforme est antérieure à la version minimale requise pour les entrées d'accès, vous pouvez rajouter l'entrée dans le fichier `aws-auth` `ConfigMap`. Pour plus d'informations, entrez `eksctl create iamidentitymapping --help` dans votre terminal.

Vous pouvez supprimer un groupe de nœuds gérés à l'aide de la commande suivante :

- the section called "`eksctl`"
- the section called "AWS Management Console"
- the section called "AWS CLI"

`eksctl`

Supprimer un groupe de nœuds gérés avec **`eksctl`**

Entrez la commande suivante. Remplacez chaque `<example value>` par vos propres valeurs.

```
eksctl delete nodegroup \
  --cluster <my-cluster> \
  --name <my-mng> \
  --region <region-code>
```

Pour plus d'options, consultez la section Supprimer et vider des groupes de nœuds dans la documentation `eksctl`.

AWS Management Console

Supprimer un groupe de nœuds gérés avec AWS Management Console

1. Ouvrez la console Amazon EKS.
2. Sur la page Clusters, choisissez le cluster qui contient le groupe de nœuds à supprimer.
3. Sur la page du cluster sélectionné, choisissez l'onglet Compute.
4. Dans la section Node Groups (Groupes de nœuds), choisissez le groupe de nœuds à supprimer. Ensuite, choisissez Supprimer.
5. Dans la boîte de dialogue de confirmation de la suppression du groupe de nœuds, entrez le nom du groupe de nœuds. Ensuite, choisissez Supprimer.

AWS CLI

Supprimer un groupe de nœuds gérés à l'aide de la AWS CLI

1. Entrez la commande suivante. Remplacez chaque `<example value>` par vos propres valeurs.

   ```
   aws eks delete-nodegroup \
     --cluster-name <my-cluster> \
     --nodegroup-name <my-mng> \
     --region <region-code>
   ```

2. S'il `cli_pager=` est défini dans la configuration de la CLI, utilisez les touches fléchées de votre clavier pour faire défiler la sortie de réponse. Appuyez sur la touche q lorsque vous avez terminé.

Pour plus d'options, consultez la `delete-nodegroup` commande dans le manuel de référence des commandes de la AWS CLI.

Gérer vous-même les nœuds grâce aux nœuds autogérés

Un cluster contient un ou plusieurs nœuds Amazon EC2 sur lesquels les pods sont planifiés. Les nœuds Amazon EKS s'exécutent dans votre compte AWS et se connectent au plan de contrôle de votre cluster via le point de terminaison de serveur de l'API du cluster. Vous êtes facturé pour ceux-ci en fonction des tarifs Amazon EC2. Pour plus d'informations, consultez Tarification Amazon EC2.

Un cluster peut contenir plusieurs groupes de nœuds. Chaque groupe de nœuds contient un ou plusieurs nœuds déployés dans un groupe Amazon EC2 Auto Scaling. Le type d'instance des nœuds au sein du groupe peut varier, par exemple lorsque vous utilisez la sélection de type d'instance basée sur les attributs avec Karpenter. Toutes les instances d'un groupe de nœuds doivent utiliser le rôle IAM du nœud Amazon EKS.

Amazon EKS fournit des images Amazon Machine Image (AMI) spécialisées appelées « AMI optimisées pour Amazon EKS ». Les AMI sont configurées pour fonctionner avec Amazon EKS. Leurs composants incluent `containerd`, `kubelet` et l'authentificateur IAM AWS. Les AMI contiennent également un script d'amorçage spécialisé qui leur permet de détecter et de se connecter automatiquement au plan de contrôle de votre cluster.

Si vous limitez l'accès au point de terminaison public de votre cluster à l'aide de blocs CIDR, nous vous recommandons d'activer également l'accès au point de terminaison privé. Cela permet aux nœuds de communiquer avec le cluster. Sans l'activation du point de terminaison privé, les blocs CIDR que vous spécifiez pour l'accès public doivent inclure les sources de sortie de votre VPC. Pour de plus amples informations, consultez the section called "Accès au point de terminaison de cluster".

Pour ajouter des nœuds autogérés à votre cluster Amazon EKS, consultez les rubriques qui suivent. Si vous lancez manuellement des nœuds autogérés, balisez chaque nœud avec la balise suivante en vous assurant que `<cluster-name>` correspond à votre cluster. Pour de plus amples informations, veuillez consulter Ajout et suppression de balises sur une ressource individuelle. Si vous suivez les étapes des guides qui suivent, l'identification requise est ajoutée au nœud pour vous.

Clé	Valeur
`kubernetes.io/cluster/<cluster-name>`	`owned`

> **Important**
>
> Les balises du service de métadonnées d'instance (IMDS) Amazon EC2 ne sont pas compatibles avec les nœuds Amazon EKS. Lorsque les balises de métadonnées d'instance sont activées, l'utilisation de barres obliques (« / ») dans les valeurs des balises est interdite. Cette limitation peut entraîner des échecs de lancement d'instances, en particulier lorsque vous utilisez des outils de gestion de nœuds tels que Karpenter ou Cluster Autoscaler, car ces services dépendent de balises contenant des barres obliques pour fonctionner correctement.

Pour plus d'informations sur les nœuds à partir d'une perspective Kubernetes générale, consultez Nœuds dans la documentation Kubernetes.

Rubriques

- Créer des nœuds Amazon Linux autogérés
- Créer des nœuds Bottlerocket autogérés
- Créer des nœuds Microsoft Windows autogérés
- Créer des nœuds Ubuntu Linux autogérés
- Mettez à jour les nœuds autogérés de votre cluster

Créer des nœuds Amazon Linux autogérés

Cette rubrique décrit comment lancer des groupes Auto Scaling de nœuds Linux qui s'enregistrent auprès de votre cluster Amazon EKS. Une fois que les nœuds ont rejoint le cluster, vous pouvez y déployer des applications Kubernetes. Vous pouvez également lancer des nœuds Amazon Linux autogérés avec `eksctl` ou le AWS Management Console. Si vous devez lancer des nœuds sur AWS Outposts, consultez. the section called "Nœuds"

- Un cluster Amazon EKS existant. Pour en déployer un, consultez the section called "Créer un cluster". Si vous avez des sous-réseaux dans la AWS région où AWS Outposts, AWS Wavelength ou Local AWS Zones est activé, ces sous-réseaux ne doivent pas avoir été transmis lorsque vous avez créé votre cluster.
- Un rôle IAM existant pour les nœuds à utiliser. Pour en créer un, consultez the section called "Rôle IAM de nœud". Si ce rôle ne comporte aucune des politiques pour le VPC CNI, le rôle distinct suivant est nécessaire pour les pods VPC CNI.
- (Facultatif, mais recommandé) Le plug-in CNI Amazon VPC pour le module complémentaire Kubernetes est configuré avec son propre rôle IAM auquel est associée la politique IAM nécessaire. Pour de plus amples informations, veuillez consulter the section called "Configuration pour IRSA".
- Connaissance des considérations répertoriées dans la section Choisissez un type d'instance de EC2 nœud Amazon optimal. Selon le type d'instance que vous choisissez, il peut y avoir des prérequis supplémentaires pour votre cluster et votre VPC.

Vous pouvez lancer des nœuds Linux autogérés à l'aide de l'une des méthodes suivantes :

- the section called "`eksctl`"
- the section called "AWS Management Console"

`eksctl`

Lancer des nœuds Linux autogérés à l'aide de **`eksctl`**

1. Installez la version `0.215.0` ou une version ultérieure de l'outil de ligne de `eksctl` commande installé sur votre appareil ou AWS CloudShell. Pour installer ou mettre à jour `eksctl`, veuillez consulter Installation dans la documentation de `eksctl`.
2. (Facultatif) Si la politique d'IAM gérée AmazonEKS_CNI_Policy est associée à votre rôle IAM de nœud Amazon EKS, nous vous recommandons de l'attribuer à un rôle IAM que vous associez au compte de service Kubernetes `aws-node` à la place. Pour de plus amples informations, veuillez consulter the section called "Configuration pour IRSA".
3. La commande suivante crée un groupe de nœuds dans un cluster existant. Remplacer *`al-nodes`* avec un nom pour votre groupe de nœuds. Le nom du groupe de nœuds ne peut pas dépasser 63 caractères. Il doit commencer par une lettre ou un chiffre, mais peut également inclure des tirets et des traits de soulignement pour les autres caractères. Remplacez *`my-cluster`* par le

nom de votre cluster. Un nom ne peut contenir que des caractères alphanumériques (sensibles à la casse) et des traits d'union. Il doit commencer par un caractère alphanumérique et ne peut pas dépasser 100 caractères. Le nom doit être unique dans la AWS région et le AWS compte dans lesquels vous créez le cluster. Remplacez les valeurs de *example value* restantes par vos propres valeurs. Par défaut, les nœuds sont créés avec la même version de Kubernetes que le plan de contrôle.

Avant de choisir une valeur pour`--node-type`, consultez l'article Choisissez un type d'instance de EC2 nœud Amazon optimal.

*my-key*Remplacez-le par le nom de votre paire de EC2 clés Amazon ou de votre clé publique. Cette clé est utilisée pour SSH dans vos nœuds après leur lancement. Si vous ne possédez pas encore de paire de EC2 clés Amazon, vous pouvez en créer une dans le AWS Management Console. Pour plus d'informations, consultez les paires de EC2 clés Amazon dans le guide de EC2 l'utilisateur Amazon.

Créez votre groupe de nœuds avec la commande suivante.

> **Important**
>
> Si vous souhaitez déployer un groupe de nœuds sur des sous-réseaux AWS Outposts, Wavelength ou Local Zone, vous devez prendre en compte d'autres considérations :
>
> - Les sous-réseaux ne doivent pas avoir été renseignés lors de la création du cluster.
> - Vous devez créer le groupe de nœuds avec un fichier config qui spécifie les sous-réseaux et `volumeType: gp2`. Pour plus d'informations, consultez Créer un nodegroup à partir d'un fichier config et Schéma du fichier Config dans la documentation `eksctl`.

```
eksctl create nodegroup \
  --cluster my-cluster \
  --name al-nodes \
  --node-type t3.medium \
  --nodes 3 \
  --nodes-min 1 \
  --nodes-max 4 \
  --ssh-access \
  --managed=false \
```

```
  --ssh-public-key my-key
```

Pour déployer un groupe de nœuds qui :

- peut attribuer un nombre d'adresses IP aux pods nettement supérieur à celui de la configuration par défaut, consultez the section called "Augmentation du nombre d'adresses IP".
- peut attribuer des adresses `IPv4` aux pods à partir d'un bloc CIDR différent de celui de l'instance, consultez the section called "Mise en réseau personnalisée".
- peut attribuer des adresses `IPv6` aux pods et aux services, consultez the section called "IPv6".
- ne dispose pas d'un accès Internet sortant, consultez the section called "Clusters privés".

Pour obtenir la liste complète de toutes les options disponibles et des valeurs par défaut, saisissez la commande suivante.

```
eksctl create nodegroup --help
```

Si les nœuds ne parviennent pas à rejoindre le cluster, consultez la section the section called "Les nœuds ne parviennent pas à joindre le cluster" du chapitre Dépannage.

L'exemple qui suit illustre un résultat. Plusieurs lignes sont affichées pendant la création des nœuds. L'une des dernières lignes de sortie est similaire à la ligne d'exemple suivante.

```
[#]  created 1 nodegroup(s) in cluster "my-cluster"
```

4. (Facultatif) Déployez un exemple d'application pour tester votre cluster et les nœuds Linux.
5. Nous recommandons de bloquer l'accès des pods à IMDS si les conditions suivantes sont remplies :
   - Vous prévoyez d'attribuer des rôles IAM à tous vos comptes de service Kubernetes, afin que les pods ne disposent que des permissions strictement nécessaires.
   - Aucun pod du cluster n'a besoin d'accéder au service de métadonnées d' EC2 instance Amazon (IMDS) pour d'autres raisons, telles que la récupération de la région actuelle AWS .

   Pour plus d'informations, consultez Restreindre l'accès au profil d'instance affecté au composant master.

AWS Management Console

Étape 1 : lancer les nœuds Linux autogérés à l'aide de la AWS Management Console

1. Téléchargez la dernière version du AWS CloudFormation modèle.

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2025-11-26/
amazon-eks-nodegroup.yaml
```

2. Attendez que le statut de votre cluster s'affiche soit `ACTIVE`. Si vous lancez vos nœuds avant que le cluster soit actif, les nœuds ne s'enregistrent pas avec le cluster et vous devrez les relancer.
3. Ouvrez la AWS CloudFormation console.
4. Choisissez Create stack (Créer une pile), puis sélectionnez Avec de nouvelles ressources (standard).
5. Pour Spécifier un modèle, sélectionnez Upload a template file (Télécharger un fichier de modèle), puis sélectionnez Choose file (Choisir un fichier).
6. Sélectionnez le fichier `amazon-eks-nodegroup.yaml` que vous avez téléchargé.
7. Sélectionnez Suivant.
8. Dans la page Specify stack details (Spécifier les détails de la pile), saisissez les paramètres suivants, puis choisissez Next (Suivant) :
   - Nom de pile : Choisissez un nom de pile pour votre AWS CloudFormation pile. Par exemple, vous pouvez l'appeler *`my-cluster-nodes`*. Un nom ne peut contenir que des caractères alphanumériques (sensibles à la casse) et des traits d'union. Il doit commencer par un caractère alphanumérique et ne peut pas dépasser 100 caractères. Le nom doit être unique dans la AWS région et le AWS compte dans lesquels vous créez le cluster.
   - ClusterName: Entrez le nom que vous avez utilisé lors de la création de votre cluster Amazon EKS. Ce nom doit être identique au nom du cluster, sinon vos nœuds ne pourront pas rejoindre le cluster.
   - ClusterControlPlaneSecurityGroup: Choisissez la SecurityGroupsvaleur à partir de la AWS CloudFormation sortie que vous avez générée lors de la création de votre VPC.

     Les étapes suivantes montrent une opération permettant de récupérer le groupe applicable.

     a. Ouvrez la console Amazon EKS.

     b. Choisissez le nom du cluster.

     c. Choisissez l'onglet Networking (Mise en réseau).

     d. Utilisez la valeur Groupes de sécurité supplémentaires comme référence lorsque vous effectuez une sélection ClusterControlPlaneSecurityGroupdans la liste déroulante.
   - ApiServerEndpoint: Entrez le point de terminaison du serveur API pour votre cluster EKS. Cela se trouve dans la section Détails de la console EKS Cluster

- CertificateAuthorityData: Entrez les données de l'autorité de certification codées en base64, qui se trouvent également dans la section Détails de la console EKS Cluster.
- ServiceCidr: Entrez la plage CIDR utilisée pour attribuer des adresses IP aux services Kubernetes au sein du cluster. Cela se trouve dans l'onglet réseau de la console EKS Cluster.
- AuthenticationMode: Sélectionnez le mode d'authentification utilisé dans le cluster EKS en consultant l'onglet d'accès dans la console du cluster EKS.
- NodeGroupName: Entrez le nom de votre groupe de nœuds. Ce nom peut être utilisé ultérieurement pour identifier le groupe de nœuds autoscaling qui est créé pour vos nœuds. Le nom du groupe de nœuds ne peut pas dépasser 63 caractères. Il doit commencer par une lettre ou un chiffre, mais peut également inclure des tirets et des traits de soulignement pour les autres caractères.
- NodeAutoScalingGroupMinSize: Entrez le nombre minimum de nœuds que votre groupe Auto Scaling de nœuds peut atteindre.
- NodeAutoScalingGroupDesiredCapacity: Entrez le nombre de nœuds que vous souhaitez atteindre lors de la création de votre pile.
- NodeAutoScalingGroupMaxSize: Entrez le nombre maximum de nœuds que votre groupe Auto Scaling de nœuds peut atteindre.
- NodeInstanceType: Choisissez un type d'instance pour vos nœuds. Pour de plus amples informations, veuillez consulter the section called “Types d'instances Amazon EC2”.
- NodeImageIdSSMParam: prérempli avec le paramètre Amazon EC2 Systems Manager d'une récente AMI Amazon Linux 2023 optimisée pour Amazon EKS pour une version variable de Kubernetes. Pour utiliser une autre version mineure de Kubernetes prise en charge avec Amazon EKS, remplacez *1.XX* par une autre version prise en charge. Nous vous recommandons de spécifier la même version de Kubernetes que celle de votre cluster.

  Vous pouvez également le remplacer *amazon-linux-2023* par un autre type d'AMI. Pour de plus amples informations, veuillez consulter the section called “Recevez les dernières nouvelles IDs”.

  > **Note**
  >
  > Le nœud Amazon EKS AMIs est basé sur Amazon Linux. Vous pouvez suivre les événements liés à la sécurité ou à la confidentialité d'Amazon Linux 2023 dans le centre de sécurité Amazon Linux ou vous abonner au flux RSS associé. Les événements

de sécurité et de confidentialité incluent une présentation du problème, les packages concernés et la manière de mettre à jour vos instances pour résoudre le problème.

- NodeImageId: (Facultatif) Si vous utilisez votre propre AMI personnalisée (au lieu d'une AMI optimisée pour Amazon EKS), entrez un ID d'AMI de nœud pour votre AWS région. Si vous spécifiez une valeur ici, elle remplace toutes les valeurs du NodeImageIdSSMParamchamp.
- NodeVolumeSize: Spécifiez une taille de volume racine pour vos nœuds, en GiB.
- NodeVolumeType: Spécifiez un type de volume racine pour vos nœuds.
- KeyName: Entrez le nom d'une paire de clés Amazon EC2 SSH que vous pourrez utiliser pour vous connecter via SSH à vos nœuds après leur lancement. Si vous ne possédez pas encore de paire de EC2 clés Amazon, vous pouvez en créer une dans le AWS Management Console. Pour plus d'informations, consultez les paires de EC2 clés Amazon dans le guide de EC2 l'utilisateur Amazon.
- VpcId: Entrez l'ID du VPC que vous avez créé.
- Sous-réseaux : choisissez les sous-réseaux que vous avez créés pour votre VPC. Si vous avez créé votre Amazon VPC en suivant les étapes décrites dans Créer un Amazon VPC pour votre cluster Amazon EKS, spécifiez uniquement les sous-réseaux privés au sein de l'Amazon VPC dans lesquels vos nœuds doivent être lancés. Vous pouvez consulter les sous-réseaux privés en ouvrant le lien de chaque sous-réseau depuis l'onglet Networking (Mise en réseau) de votre cluster.

⚠ Important

- Si certains sous-réseaux sont des sous-réseaux publics, leur paramètre d'attribution automatique d'adresse IP publique doit être activé. Si le paramètre n'est pas activé pour le sous-réseau public, aucun nœud que vous déployez sur ce sous-réseau public ne se verra attribuer d'adresse IP publique et ne pourra pas communiquer avec le cluster ou d'autres AWS services. Si le sous-réseau a été déployé avant le 26 mars 2020 en utilisant l'un des modèles de AWS CloudFormation VPC Amazon EKS ou `eksctl` en utilisant, l'attribution automatique d'adresses IP publiques est désactivée pour les sous-réseaux publics. Pour plus d'informations sur la façon d'activer l'attribution d'adresses IP publiques pour un sous-réseau, consultez la section Modification de l'attribut d' IPv4 adressage public de votre sous-réseau. Si le nœud est déployé sur un sous-réseau privé, il est capable de communiquer avec le cluster et d'autres AWS services via une passerelle NAT.

- Si les sous-réseaux n'ont pas d'accès Internet, assurez-vous de prendre connaissance des considérations et des étapes supplémentaires décrites dans Déployer des clusters privés avec un accès Internet limité.
- Si vous sélectionnez AWS les sous-réseaux Outposts, Wavelength ou Local Zone, les sous-réseaux ne doivent pas avoir été transmis lors de la création du cluster.

9. Sélectionnez les choix que vous souhaitez sur la page Configure stack options (Configurer les options de la pile), puis choisissez Next (Suivant).
10. Cochez la case à gauche de J'accuse réception AWS CloudFormation susceptible de créer des ressources IAM. , puis choisissez Create stack.
11. Lorsque la création de votre pile est terminée, sélectionnez la pile dans la console et choisissez Outputs (Sorties). Si vous utilisez le mode `EKS API and ConfigMap` d'authentification `EKS API` ou, il s'agit de la dernière étape.
12. Si vous utilisez le mode `ConfigMap` d'authentification, enregistrez le NodeInstanceRolepour le groupe de nœuds créé.

Étape 2 : autoriser les nœuds à rejoindre votre cluster

> **Note**
>
> Les deux étapes suivantes ne sont nécessaires que si le mode d'authentification Configmap est utilisé dans le cluster EKS. En outre, si vous avez lancé des nœuds au sein d'un VPC privé sans accès Internet sortant, veillez à autoriser les nœuds à rejoindre votre cluster depuis le VPC.

1. Vérifiez si vous avez déjà appliqué le `ConfigMap aws-auth`.

```
kubectl describe configmap -n kube-system aws-auth
```

2. Si vous voyez un `ConfigMap aws-auth`, mettez-le à jour si nécessaire.
   a. Ouvrez le `ConfigMap` pour le modifier.

```
kubectl edit -n kube-system configmap/aws-auth
```

   b. Ajoutez une nouvelle entrée `mapRoles` si nécessaire. Définissez la `rolearn` valeur sur la NodeInstanceRolevaleur que vous avez enregistrée lors de la procédure précédente.

```
[...]
data:
  mapRoles: |
    - rolearn: <ARN of instance role (not instance profile)>
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
[...]
```

c. Enregistrez le fichier et quittez votre éditeur de texte.

3. Si vous avez reçu un message d'erreur indiquant « `Error from server (NotFound): configmaps "aws-auth" not found` », appliquez le stock `ConfigMap`.

   a. Téléchargez la mappe de configuration.

   ```
   curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/aws-auth-cm.yaml
   ```

   b. Dans le `aws-auth-cm.yaml` fichier, définissez la `rolearn` valeur sur celle NodeInstanceRoleque vous avez enregistrée lors de la procédure précédente. Pour ce faire, utilisez un éditeur de texte ou remplacez *`my-node-instance-role`* et exécutez la commande suivante :

   ```
   sed -i.bak -e 's|<ARN of instance role (not instance profile)>|my-node-instance-role|' aws-auth-cm.yaml
   ```

   c. Appliquez la configuration. L'exécution de cette commande peut prendre quelques minutes.

   ```
   kubectl apply -f aws-auth-cm.yaml
   ```

4. Observez le statut de vos nœuds et attendez qu'ils obtiennent le statut `Ready`.

   ```
   kubectl get nodes --watch
   ```

   Saisissez `Ctrl+C` pour revenir à une invite de shell.

> **Note**
>
> Si vous recevez d'autres erreurs concernant les types d'autorisations ou de ressources, consultez the section called "Accès non autorisé ou refusé (`kubectl`)" dans la rubrique relative à la résolution des problèmes.

Si les nœuds ne parviennent pas à rejoindre le cluster, consultez la section the section called "Les nœuds ne parviennent pas à joindre le cluster" du chapitre Dépannage.

5. (Nœuds GPU uniquement) Si vous avez choisi un type d'instance GPU et l'AMI accélérée optimisée pour Amazon EKS, vous devez appliquer le plug-in d'appareil NVIDIA pour Kubernetes en tant que tel DaemonSet sur votre cluster. *`vX.X.X`*Remplacez-le par la s-device-plugin version Nvidia/K8 de votre choix avant d'exécuter la commande suivante.

```
kubectl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/vX.X.X/
deployments/static/nvidia-device-plugin.yml
```

Étape 3 : actions supplémentaires

1. (Facultatif) Déployez un exemple d'application pour tester votre cluster et les nœuds Linux.
2. (Facultatif) Si la politique IAM gérée par Amazoneks_CNI_Policy (si vous avez `IPv4` un cluster) ou celle (que vous avez créée vous-même si vous avez un `IPv6` cluster) est attachée *`AmazonEKS_CNI_IPv6_Policy`* au rôle IAM de votre nœud Amazon EKS, nous vous recommandons de l'attribuer à un rôle IAM que vous associez plutôt au compte de service Kubernetes. `aws-node` Pour de plus amples informations, veuillez consulter the section called "Configuration pour IRSA".
3. Nous recommandons de bloquer l'accès des pods à IMDS si les conditions suivantes sont remplies :
   - Vous prévoyez d'attribuer des rôles IAM à tous vos comptes de service Kubernetes, afin que les pods ne disposent que des permissions strictement nécessaires.
   - Aucun pod du cluster n'a besoin d'accéder au service de métadonnées d' EC2 instance Amazon (IMDS) pour d'autres raisons, telles que la récupération de la région actuelle AWS .

   Pour plus d'informations, consultez Restreindre l'accès au profil d'instance affecté au composant master.

Créer des nœuds Bottlerocket autogérés

> **Note**
>
> Les groupes de nœuds gérés peuvent offrir certains avantages pour votre cas d'utilisation. Pour de plus amples informations, veuillez consulter the section called "Groupes de nœuds gérés".

Cette rubrique décrit comment lancer des groupes Auto Scaling de nœuds Bottlerocket qui s'enregistrent auprès de votre cluster Amazon EKS. Bottlerocket est un système d'exploitation open source basé sur Linux AWS que vous pouvez utiliser pour exécuter des conteneurs sur des machines virtuelles ou des hôtes bare metal. Une fois que les nœuds ont rejoint le cluster, vous pouvez y déployer des applications Kubernetes. Pour plus d'informations sur Bottlerocket, consultez la section Utilisation d'une AMI Bottlerocket avec Amazon EKS activé GitHub et la prise en charge des AMI personnalisées dans la documentation. `eksctl`

Pour plus d'informations sur les mises à niveau sur place, consultez Bottlerocket Update Operator on. GitHub

> **Important**
>
> - Les nœuds Amazon EKS sont EC2 des instances Amazon standard, et ils vous sont facturés sur la base des prix habituels des EC2 instances Amazon. Pour plus d'informations, consultez les EC2 tarifs Amazon.
> - Vous pouvez lancer des nœuds Bottlerocket dans des clusters étendus Amazon EKS sur AWS Outposts, mais vous ne pouvez pas les lancer dans des clusters locaux sur Outposts. AWS Pour de plus amples informations, veuillez consulter Amazon EKS sur AWS Outposts.
> - Vous pouvez effectuer un déploiement sur EC2 des instances Amazon avec x86 ou des processeurs Arm. Cependant, vous ne pouvez pas déployer sur des instances équipées de puces Inferentia.
> - Bottlerocket est compatible avec. AWS CloudFormation Cependant, aucun CloudFormation modèle officiel ne peut être copié pour déployer des nœuds Bottlerocket pour Amazon EKS.

- Les images Bottlerocket ne sont pas fournies avec un serveur SSH ou un shell. Vous pouvez utiliser des méthodes out-of-band d'accès pour autoriser SSH à activer le conteneur d'administration et pour passer certaines étapes de configuration d'amorçage avec les données utilisateur. Pour plus d'informations, consultez ces sections dans le README.md de bottlerocket sur GitHub :
  - Exploration
  - Conteneur d'administration
  - Paramètres Kubernetes

Cette procédure nécessite `eksctl` version `0.215.0` ou ultérieure. Vous pouvez vérifier votre version avec la commande suivante :

```
eksctl version
```

Pour obtenir des instructions sur l'installation ou la mise à niveau d'`eksctl`, consultez Installation dans la documentation `eksctl`. REMARQUE : cette procédure ne fonctionne que pour les clusters créés avec `eksctl`.

1. Copiez les contenus suivants sur votre appareil. Remplacez *`my-cluster`* par le nom de votre cluster. Un nom ne peut contenir que des caractères alphanumériques (sensibles à la casse) et des traits d'union. Il doit commencer par un caractère alphanumérique et ne peut pas dépasser 100 caractères. Le nom doit être unique au sein de la AWS région et du AWS compte dans lesquels vous créez le cluster. Remplacer *`ng-bottlerocket`* avec un nom pour votre groupe de nœuds. Le nom du groupe de nœuds ne peut pas dépasser 63 caractères. Il doit commencer par une lettre ou un chiffre, mais peut également inclure des tirets et des traits de soulignement pour les autres caractères. Pour effectuer un déploiement sur des instances Arm, remplacez *`m5.large`* par un type d'instance Arm. Remplacez-le *`my-ec2-keypair-name`* par le nom d'une paire de clés Amazon EC2 SSH que vous pourrez utiliser pour vous connecter via SSH à vos nœuds après leur lancement. Si vous ne possédez pas encore de paire de EC2 clés Amazon, vous pouvez en créer une dans le AWS Management Console. Pour plus d'informations, consultez les paires de EC2 clés Amazon dans le guide de EC2 l'utilisateur Amazon. Remplacez toutes les valeurs d'exemple restantes par vos propres valeurs. Une fois les remplacements effectués, exécutez la commande modifiée pour créer le fichier `bottlerocket.yaml`.

   Si vous spécifiez un type d' EC2 instance Arm Amazon, consultez les points à prendre en compte dans Arm Amazon Linux optimisé pour Amazon EKS AMIs avant le déploiement. Pour obtenir des

instructions sur le déploiement à l'aide d'une AMI personnalisée, consultez les sections Building Bottlerocket on et GitHub Custom AMI support dans la documentation. `eksctl` Pour déployer un groupe de nœuds gérés, déployez une AMI personnalisée à l'aide d'un modèle de lancement. Pour de plus amples informations, veuillez consulter the section called "Modèles de lancement".

> **Important**
>
> Pour déployer un groupe de nœuds sur des sous-réseaux AWS Outposts, AWS Wavelength ou AWS Local Zone, ne transmettez pas les sous-réseaux Outposts AWS , AWS Wavelength ou AWS Local Zone lorsque vous créez le cluster. Vous devez spécifier les sous-réseaux dans l'exemple suivant. Pour plus d'informations, consultez Créer un nodegroup à partir d'un fichier de configuration et Schéma du fichier de configuration dans la documentation `eksctl`. Remplacez *`region-code`* par la AWS région dans laquelle se trouve votre cluster.

```
cat >bottlerocket.yaml <<EOF
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-cluster
  region: region-code
  version: '1.33'

iam:
  withOIDC: true

nodeGroups:
  - name: ng-bottlerocket
    instanceType: m5.large
    desiredCapacity: 3
    amiFamily: Bottlerocket
    ami: auto-ssm
    iam:
       attachPolicyARNs:
          - arn:aws: iam::aws:policy/AmazonEKSWorkerNodePolicy
          - arn:aws: iam::aws:policy/AmazonEC2ContainerRegistryReadOnly
          - arn:aws: iam::aws:policy/AmazonSSMManagedInstanceCore
```

```
          - arn:aws: iam::aws:policy/AmazonEKS_CNI_Policy
    ssh:
        allow: true
        publicKeyName: my-ec2-keypair-name
EOF
```

2. Deployez vos nœuds avec la commande suivante :

```
eksctl create nodegroup --config-file=bottlerocket.yaml
```

   L'exemple qui suit illustre un résultat.

   Plusieurs lignes sont affichées pendant la création des nœuds. L'une des dernières lignes de sortie est similaire à la ligne d'exemple suivante.

```
[#]  created 1 nodegroup(s) in cluster "my-cluster"
```

3. (Facultatif) Créez un volume persistant Kubernetes sur un nœud Bottlerocket à l'aide du Plugin CSI Amazon EBS. Le pilote Amazon EBS par défaut s'appuie sur des outils de système de fichiers qui ne sont pas inclus dans Bottlerocket. Pour obtenir plus d'informations sur la création d'une classe de stockage à l'aide du pilote, consultez the section called "Amazon EBS".
4. (Facultatif) Par défaut, `kube-proxy` définit le paramètre de noyau `nf_conntrack_max` sur une valeur par défaut qui peut différer de ce que Bottlerocket a initialement défini au démarrage. Pour conserver le paramètre par défaut de Bottlerocket, modifiez la configuration `kube-proxy` à l'aide de la commande suivante.

```
kubectl edit -n kube-system daemonset kube-proxy
```

   Ajoutez `--conntrack-max-per-core` et `--conntrack-min` aux arguments `kube-proxy` présentés dans l'exemple suivant. Un paramètre de `0` signifie aucun changement.

```
      containers:
      - command:
        - kube-proxy
        - --v=2
        - --config=/var/lib/kube-proxy-config/config
        - --conntrack-max-per-core=0
        - --conntrack-min=0
```

5. (Facultatif) Déployer un exemple d'application pour tester vos nœuds Bottlerocket.

6. Nous recommandons de bloquer l'accès des pods à IMDS si les conditions suivantes sont remplies :
   - Vous prévoyez d'attribuer des rôles IAM à tous vos comptes de service Kubernetes, afin que les pods ne disposent que des permissions strictement nécessaires.
   - Aucun pod du cluster n'a besoin d'accéder au service de métadonnées d' EC2 instance Amazon (IMDS) pour d'autres raisons, telles que la récupération de la région actuelle AWS .

   Pour plus d'informations, consultez Restreindre l'accès au profil d'instance affecté au composant master.

Créer des nœuds Microsoft Windows autogérés

Cette rubrique décrit comment lancer des groupes Auto Scaling de nœuds Windows qui s'enregistrent auprès de votre cluster Amazon EKS. Une fois que les nœuds ont rejoint le cluster, vous pouvez y déployer des applications Kubernetes.

> **Important**
>
> - Les nœuds Amazon EKS sont des instances Amazon EC2 standard pour lesquelles vous êtes facturé en fonction du tarif normal pour les instances Amazon EC2. Pour plus d'informations, consultez Tarification Amazon EC2.
> - Vous pouvez lancer des nœuds Windows dans des clusters étendus Amazon EKS sur AWS Outposts, mais vous ne pouvez pas les lancer dans des clusters locaux sur AWS Outposts. Pour de plus amples informations, veuillez consulter Amazon EKS sur AWS Outposts.

Activer la prise en charge de Windows pour votre cluster avec. Nous vous recommandons de passer en revue les considérations importantes avant de lancer un groupe de nœuds Windows. Pour de plus amples informations, veuillez consulter the section called "Activer la prise en charge de Windows".

Vous pouvez lancer des nœuds Windows autogérés à l'aide de l'une des méthodes suivantes :

- the section called "`eksctl`"
- the section called "AWS Management Console"

`eksctl`

Lancer des nœuds Windows autogérés à l'aide de **`eksctl`**

Cette procédure exige que vous installiez `eksctl` et que votre version `eksctl` soit au moins la version `0.215.0`. Vous pouvez vérifier votre version à l'aide de la commande suivante.

```
eksctl version
```

Pour les instructions d'installation ou de mise à niveau de `eksctl`, consultez la rubrique Installation dans la documentation `eksctl`.

> **Note**
>
> Cette procédure fonctionne uniquement pour les clusters créés avec `eksctl`.

1. (Facultatif) Si la politique IAM gérée par Amazoneks_CNI_Policy (si vous avez `IPv4` un cluster) ou celle (que vous avez créée vous-même si vous avez un `IPv6` cluster) est attachée *`AmazonEKS_CNI_IPv6_Policy`* au rôle IAM de votre nœud Amazon EKS, nous vous recommandons de l'attribuer à un rôle IAM que vous associez plutôt au compte de service Kubernetes. `aws-node` Pour de plus amples informations, veuillez consulter the section called "Configuration pour IRSA".
2. Cette procédure part du principe que vous disposez d'un cluster existant. Si vous ne disposez pas encore d'un cluster Amazon EKS et d'un groupe de nœuds Amazon Linux auquel ajouter un groupe de nœuds Windows, nous vous recommandons de suivre the section called "Créer un cluster (eksctl)". Ce guide fournit une démonstration complète pour créer un cluster Amazon EKS avec des nœuds Amazon Linux.

   Créez votre groupe de nœuds avec la commande suivante. Remplacez *`region-code`* par la AWS région dans laquelle se trouve votre cluster. Remplacez *`my-cluster`* par le nom de votre cluster. Un nom ne peut contenir que des caractères alphanumériques (sensibles à la casse) et des traits d'union. Il doit commencer par un caractère alphanumérique et ne peut pas dépasser 100 caractères. Le nom doit être unique dans la AWS région et le AWS compte dans lesquels vous créez le cluster. Remplacer *`ng-windows`* avec un nom pour votre groupe de nœuds. Le nom du groupe de nœuds ne peut pas dépasser 63 caractères. Il doit commencer par une lettre ou un chiffre, mais peut également inclure des tirets et des traits de soulignement pour les autres

caractères. Vous pouvez *2019* remplacer par 2022 pour utiliser Windows Server 2022 ou 2025 Windows Server 2025. Remplacez le reste des valeurs d'exemple par vos propres valeurs.

> **Important**
>
> Pour déployer un groupe de nœuds sur des sous-réseaux AWS Outposts, AWS Wavelength ou AWS Local Zone, ne transmettez pas les sous-réseaux Outposts AWS , Wavelength ou Local Zone lorsque vous créez le cluster. Créez le groupe de nœuds avec un fichier de configuration, en spécifiant les AWS sous-réseaux Outposts, Wavelength ou Local Zone. Pour plus d'informations, consultez Créer un nodegroup à partir d'un fichier config et Schéma du fichier Config dans la documentation `eksctl`.

```
eksctl create nodegroup \
    --region region-code \
    --cluster my-cluster \
    --name ng-windows \
    --node-type t2.large \
    --nodes 3 \
    --nodes-min 1 \
    --nodes-max 4 \
    --managed=false \
    --node-ami-family WindowsServer2019FullContainer
```

> **Note**
>
> - Si les nœuds ne parviennent pas à rejoindre le cluster, reportez-vous à the section called "Les nœuds ne parviennent pas à joindre le cluster" dans le guide de dépannage.
> - Pour voir les options disponibles pour les commandes `eksctl`, saisissez la commande suivante.
>
>   ```
>   eksctl command -help
>   ```

L'exemple qui suit illustre un résultat. Plusieurs lignes sont affichées pendant la création des nœuds. L'une des dernières lignes de sortie est similaire à la ligne d'exemple suivante.

```
[#]  created 1 nodegroup(s) in cluster "my-cluster"
```

3. (Facultatif) Déployez un exemple d'application pour tester votre cluster et les nœuds Windows.
4. Nous recommandons de bloquer l'accès des pods à IMDS si les conditions suivantes sont remplies :
   - Vous prévoyez d'attribuer des rôles IAM à tous vos comptes de service Kubernetes, afin que les pods ne disposent que des permissions strictement nécessaires.
   - Aucun pod du cluster n'a besoin d'accéder au service de métadonnées d'instance Amazon EC2 (IMDS) pour d'autres raisons, telles que la récupération de la région actuelle. AWS

   Pour plus d'informations, consultez Restreindre l'accès au profil d'instance affecté au composant master.

AWS Management Console

Conditions préalables

- Un cluster Amazon EKS existant et un groupe de nœuds Linux. Si vous ne disposez pas de ces ressources, nous vous recommandons de les créer à l'aide de l'un de nos guides dans Démarrer. Ces guides décrivent comment créer un cluster Amazon EKS avec des nœuds Linux.
- Un VPC et un groupe de sécurité existants qui remplissent les conditions requises pour un cluster Amazon EKS. Pour plus d'informations, consultez the section called "Exigences requises pour le VPC et les sous-réseaux" et the section called "Exigences de groupe de sécurité". Les guides disponibles dans Démarrer créent un VPC qui répond aux exigences. Vous pouvez également suivre les instructions de Créer un Amazon VPC pour votre cluster Amazon EKS pour en créer un manuellement.
- Un cluster Amazon EKS existant qui utilise un VPC et un groupe de sécurité qui remplit les conditions requises pour un cluster Amazon EKS. Pour de plus amples informations, veuillez consulter the section called "Créer un cluster". Si vous avez des sous-réseaux dans la AWS région où AWS Outposts, AWS Wavelength ou Local AWS Zones est activé, ces sous-réseaux ne doivent pas avoir été transmis lorsque vous avez créé le cluster.

Étape 1 : lancer des nœuds Windows autogérés à l'aide de la AWS Management Console

1. Attendez que le statut de votre cluster s'affiche soit `ACTIVE`. Si vous lancez vos nœuds avant que le cluster soit actif, les nœuds ne s'enregistrent pas avec le cluster et vous devez les relancer.

2. Ouvrez la console AWS CloudFormation .
3. Sélectionnez Créer la pile.
4. Dans Spécifier le modèle, sélectionnez URL Amazon S3.
5. Copiez l'URL suivante et collez-la dans l'URL Amazon S3.

```
https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2023-02-09/amazon-eks-windows-nodegroup.yaml
```

6. Sélectionnez Next (Suivant) deux fois.
7. Sur la page Création rapide d'une pile, saisissez les paramètres suivants en conséquence :
   - Nom de pile : Choisissez un nom de pile pour votre AWS CloudFormation pile. Par exemple, vous pouvez l'appeler `my-cluster-nodes`.
   - ClusterName: Entrez le nom que vous avez utilisé lors de la création de votre cluster Amazon EKS.

     > **Important**
     >
     > Ce nom doit correspondre exactement au nom que vous avez utilisé dans Étape 1 : créer votre cluster Amazon EKS. Sinon, vos nœuds ne pourront pas rejoindre le cluster.

   - ClusterControlPlaneSecurityGroup: Choisissez le groupe de sécurité dans la AWS CloudFormation sortie que vous avez générée lors de la création de votre VPC. Les étapes suivantes montrent une méthode permettant de récupérer le groupe applicable.
     a. Ouvrez la console Amazon EKS.
     b. Choisissez le nom du cluster.
     c. Choisissez l'onglet Networking (Mise en réseau).
     d. Utilisez la valeur Groupes de sécurité supplémentaires comme référence lorsque vous effectuez une sélection ClusterControlPlaneSecurityGroupdans la liste déroulante.
   - NodeGroupName: Entrez le nom de votre groupe de nœuds. Ce nom peut être utilisé ultérieurement pour identifier le groupe de nœuds autoscaling qui est créé pour vos nœuds. Le nom du groupe de nœuds ne peut pas dépasser 63 caractères. Il doit commencer par une lettre ou un chiffre, mais peut également inclure des tirets et des traits de soulignement pour les autres caractères.
   - NodeAutoScalingGroupMinSize: Entrez le nombre minimum de nœuds que votre groupe Auto Scaling de nœuds peut atteindre.

- NodeAutoScalingGroupDesiredCapacity: Entrez le nombre de nœuds que vous souhaitez atteindre lors de la création de votre pile.
- NodeAutoScalingGroupMaxSize: Entrez le nombre maximum de nœuds que votre groupe Auto Scaling de nœuds peut atteindre.
- NodeInstanceType: Choisissez un type d'instance pour vos nœuds. Pour de plus amples informations, veuillez consulter the section called “Types d'instances Amazon EC2”.

> **Note**
>
> Les types d'instances pris en charge pour la dernière version du plugin Amazon VPC CNI pour Kubernetes sont répertoriés dans vpc_ip_resource_limit.go on. GitHub Vous devrez peut-être mettre à jour votre version de CNI pour utiliser les derniers types d'instances pris en charge. Pour de plus amples informations, veuillez consulter the section called “CNI Amazon VPC”.

- NodeImageIdSSMParam: prérempli avec le paramètre Amazon EC2 Systems Manager de l'ID AMI Windows Core optimisé pour Amazon EKS actuellement recommandé. Pour utiliser la version complète de Windows, remplacez `Core` par `Full`.
- NodeImageId: (Facultatif) Si vous utilisez votre propre AMI personnalisée (au lieu d'une AMI optimisée pour Amazon EKS), entrez un ID d'AMI de nœud pour votre AWS région. Si vous spécifiez une valeur pour ce champ, elle remplace toutes les valeurs du NodeImageIdSSMParamchamp.
- NodeVolumeSize: Spécifiez une taille de volume racine pour vos nœuds, en GiB.
- KeyName: Entrez le nom d'une paire de clés SSH Amazon EC2 que vous pourrez utiliser pour vous connecter via SSH à vos nœuds après leur lancement. Si vous ne possédez pas déjà une paire de clés Amazon EC2, vous pouvez en créer une dans l' AWS Management Console. Pour plus d'informations, veuillez consulter la rubrique Paires de clés Amazon EC2 dans le Guide de l'utilisateur Amazon EC2.

> **Note**
>
> Si vous ne fournissez pas de paire de clés ici, la AWS CloudFormation pile ne sera pas créée.

- BootstrapArguments: Spécifiez tous les arguments facultatifs à transmettre au script bootstrap du nœud, tels que des `kubelet` arguments supplémentaires en utilisant-`KubeletExtraArgs`.

- Désactiver IMDSv1 : par défaut, chaque nœud prend en charge les versions 1 du service de métadonnées d'instance (IMDSv1) et IMDSv2. Vous pouvez désactiver IMDSv1. Pour empêcher l'utilisation des futurs nœuds et pods du groupe de nœuds MDSv1, définissez Disable IMDSv1 sur true. Pour de plus amples informations au sujet d'IMDS, consultez Configuration du service des métadonnées d'instance.
- VpcId: Sélectionnez l'ID du VPC que vous avez créé.
- NodeSecurityGroups: Sélectionnez le groupe de sécurité créé pour votre groupe de nœuds Linux lorsque vous avez créé votre VPC. Si vos nœuds Linux sont associés à plusieurs groupes de sécurité, spécifiez-les tous. C'est le cas, par exemple, si le groupe de nœuds Linux a été créé avec `eksctl`.
- Sous-réseaux : choisissez les sous-réseaux que vous avez créés. Si vous avez créé votre VPC en suivant les étapes décrites dans Créer un Amazon VPC pour votre cluster Amazon EKS, spécifiez uniquement les sous-réseaux privés au sein du VPC dans lesquels vos nœuds doivent être lancés.

> ⚠ Important
>
> - Si certains sous-réseaux sont des sous-réseaux publics, leur paramètre d'attribution automatique d'adresse IP publique doit être activé. Si le paramètre n'est pas activé pour le sous-réseau public, aucun nœud que vous déployez sur ce sous-réseau public ne se verra attribuer d'adresse IP publique et ne pourra pas communiquer avec le cluster ou d'autres AWS services. Si le sous-réseau a été déployé avant le 26 mars 2020 en utilisant l'un des modèles de AWS CloudFormation VPC Amazon EKS ou `eksctl` en utilisant, l'attribution automatique d'adresses IP publiques est désactivée pour les sous-réseaux publics. Pour plus d'informations sur la façon d'activer l'attribution d'adresses IP publiques pour un sous-réseau, consultez la section Modification de l'attribut d' IPv4 adressage public de votre sous-réseau. Si le nœud est déployé sur un sous-réseau privé, il est capable de communiquer avec le cluster et d'autres AWS services via une passerelle NAT.
> - Si les sous-réseaux n'ont pas accès à Internet, veuillez vous assurer de prendre connaissance des considérations et des étapes supplémentaires décrites dans Déployer des clusters privés avec un accès Internet limité.
> - Si vous sélectionnez AWS les sous-réseaux Outposts, Wavelength ou Local Zone, les sous-réseaux ne doivent pas avoir été transmis lors de la création du cluster.

8. Confirmez que la pile peut créer des ressources IAM, puis choisissez Create stack (Créer une pile).
9. Lorsque la création de votre pile est terminée, sélectionnez la pile dans la console et choisissez Outputs (Sorties).
10. Enregistrez le NodeInstanceRolepour le groupe de nœuds créé. Vous en aurez besoin lors de la configuration de vos nœuds Windows Amazon EKS.

Étape 2 : autoriser les nœuds à rejoindre votre cluster

1. Vérifiez si vous avez déjà appliqué le `ConfigMap` `aws-auth`.

```
kubectl describe configmap -n kube-system aws-auth
```

2. Si vous voyez un `ConfigMap` `aws-auth`, mettez-le à jour si nécessaire.
   a. Ouvrez le `ConfigMap` pour le modifier.

```
kubectl edit -n kube-system configmap/aws-auth
```

   b. Ajoutez de nouvelles entrées `mapRoles` selon vos besoins. Définissez les `rolearn` valeurs selon les NodeInstanceRolevaleurs que vous avez enregistrées dans les procédures précédentes.

```
[...]
data:
  mapRoles: |
- rolearn: <ARN of linux instance role (not instance profile)>
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
    - rolearn: <ARN of windows instance role (not instance profile)>
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
        - eks:kube-proxy-windows
[...]
```

   c. Enregistrez le fichier et quittez votre éditeur de texte.

3. Si vous avez reçu un message d'erreur indiquant « `Error from server (NotFound): configmaps "aws-auth" not found` », appliquez le stock `ConfigMap`.

   a. Téléchargez la mappe de configuration.

   ```
   curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/aws-auth-cm-windows.yaml
   ```

   b. Dans le `aws-auth-cm-windows.yaml` fichier, définissez les `rolearn` valeurs en fonction des NodeInstanceRolevaleurs applicables que vous avez enregistrées dans les procédures précédentes. Vous pouvez effectuer cette opération à l'aide d'un éditeur de texte ou en remplaçant les valeurs d'exemple et en exécutant la commande suivante :

   ```
   sed -i.bak -e 's|<ARN of linux instance role (not instance profile)>|my-node-linux-instance-role|' \
       -e 's|<ARN of windows instance role (not instance profile)>|my-node-windows-instance-role|' aws-auth-cm-windows.yaml
   ```

   > **Important**
   >
   > - Ne modifiez aucune autre ligne de ce fichier.
   > - Veuillez ne pas utiliser le même rôle IAM pour les nœuds Windows et Linux.

   c. Appliquez la configuration. L'exécution de cette commande peut prendre quelques minutes.

   ```
   kubectl apply -f aws-auth-cm-windows.yaml
   ```

4. Observez le statut de vos nœuds et attendez qu'ils obtiennent le statut `Ready`.

   ```
   kubectl get nodes --watch
   ```

   Saisissez `Ctrl+C` pour revenir à une invite de shell.

   > **Note**
   >
   > Si vous recevez d'autres erreurs concernant les types d'autorisations ou de ressources, consultez the section called "Accès non autorisé ou refusé (`kubectl`)" dans la rubrique relative à la résolution des problèmes.

Si les nœuds ne parviennent pas à rejoindre le cluster, consultez la section the section called "Les nœuds ne parviennent pas à joindre le cluster" du chapitre Dépannage.

Étape 3 : actions supplémentaires

1. (Facultatif) Déployez un exemple d'application pour tester votre cluster et les nœuds Windows.
2. (Facultatif) Si la politique IAM gérée par Amazoneks_CNI_Policy (si vous avez `IPv4` un cluster) ou celle (que vous avez créée vous-même si vous avez un `IPv6` cluster) est attachée *`AmazonEKS_CNI_IPv6_Policy`* au rôle IAM de votre nœud Amazon EKS, nous vous recommandons de l'attribuer à un rôle IAM que vous associez plutôt au compte de service Kubernetes. `aws-node` Pour de plus amples informations, veuillez consulter the section called "Configuration pour IRSA".
3. Nous recommandons de bloquer l'accès des pods à IMDS si les conditions suivantes sont remplies :
   - Vous prévoyez d'attribuer des rôles IAM à tous vos comptes de service Kubernetes, afin que les pods ne disposent que des permissions strictement nécessaires.
   - Aucun pod du cluster n'a besoin d'accéder au service de métadonnées d'instance Amazon EC2 (IMDS) pour d'autres raisons, telles que la récupération de la région actuelle. AWS

   Pour plus d'informations, consultez Restreindre l'accès au profil d'instance affecté au composant master.

Créer des nœuds Ubuntu Linux autogérés

> **Note**
>
> Les groupes de nœuds gérés peuvent offrir certains avantages pour votre cas d'utilisation. Pour de plus amples informations, veuillez consulter the section called "Groupes de nœuds gérés".

Cette rubrique décrit comment lancer des groupes Auto Scaling de nœuds Ubuntu sur Amazon Elastic Kubernetes Service (EKS) ou Ubuntu Pro sur Amazon Elastic Kubernetes Service (EKS) qui s'enregistrent auprès de votre cluster Amazon EKS. Ubuntu et Ubuntu Pro pour EKS sont basés sur le LTS officiel Ubuntu Minimal, incluent le AWS noyau personnalisé développé conjointement

avec AWS EKS et spécialement conçu pour EKS. Ubuntu Pro renforce la sécurité en prenant en charge les périodes de support étendu EKS, le livepatch du noyau, la conformité FIPS et la possibilité d'exécuter un nombre illimité de conteneurs Pro.

Une fois les nœuds intégrés au cluster, vous pouvez y déployer des applications conteneurisées. Pour plus d'informations, consultez la documentation relative à Ubuntu sur AWS et Prise en charge des AMI personnalisées dans la documentation `eksctl`.

> **Important**
>
> - Les nœuds Amazon EKS sont EC2 des instances Amazon standard, et ils vous sont facturés sur la base des prix habituels des EC2 instances Amazon. Pour plus d'informations, consultez les EC2 tarifs Amazon.
> - Vous pouvez lancer des nœuds Ubuntu dans des clusters étendus Amazon EKS sur AWS Outposts, mais vous ne pouvez pas les lancer dans des clusters locaux sur AWS Outposts. Pour de plus amples informations, veuillez consulter Amazon EKS sur AWS Outposts.
> - Vous pouvez déployer sur des EC2 instances Amazon avec x86 ou des processeurs Arm. Cependant, les instances équipées de puces Inferentia peuvent nécessiter l'installation préalable du Kit SDK de Neuron.

Cette procédure nécessite `eksctl` version `0.215.0` ou ultérieure. Vous pouvez vérifier votre version avec la commande suivante :

```
eksctl version
```

Pour obtenir des instructions sur l'installation ou la mise à niveau d'`eksctl`, consultez Installation dans la documentation `eksctl`. REMARQUE : cette procédure ne fonctionne que pour les clusters créés avec `eksctl`.

1. Copiez les contenus suivants sur votre appareil. Remplacez `my-cluster` par le nom de votre cluster. Un nom ne peut contenir que des caractères alphanumériques (sensibles à la casse) et des traits d'union. Il doit commencer par une lettre et ne peut pas dépasser 100 caractères. Remplacer `ng-ubuntu` avec un nom pour votre groupe de nœuds. Le nom du groupe de nœuds ne peut pas dépasser 63 caractères. Il doit commencer par une lettre ou un chiffre, mais peut également inclure des tirets et des traits de soulignement pour les autres caractères. Pour effectuer un déploiement sur des instances Arm, remplacez `m5.large` par un type d'instance

Arm. Remplacez-le `my-ec2-keypair-name` par le nom d'une paire de clés Amazon EC2 SSH que vous pourrez utiliser pour vous connecter via SSH à vos nœuds après leur lancement. Si vous ne possédez pas encore de paire de EC2 clés Amazon, vous pouvez en créer une dans le AWS Management Console. Pour plus d'informations, consultez les paires de EC2 clés Amazon dans le guide de EC2 l'utilisateur Amazon. Remplacez toutes les valeurs d'exemple restantes par vos propres valeurs. Une fois les remplacements effectués, exécutez la commande modifiée pour créer le fichier `ubuntu.yaml`.

> **Important**
>
> Pour déployer un groupe de nœuds sur des sous-réseaux AWS Outposts, AWS Wavelength ou AWS Local Zone, ne transmettez pas les sous-réseaux Outposts AWS , AWS Wavelength ou AWS Local Zone lorsque vous créez le cluster. Vous devez spécifier les sous-réseaux dans l'exemple suivant. Pour plus d'informations, consultez Créer un nodegroup à partir d'un fichier de configuration et Schéma du fichier de configuration dans la documentation `eksctl`. Remplacez *`region-code`* par la AWS région dans laquelle se trouve votre cluster.

```
cat >ubuntu.yaml <<EOF
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-cluster
  region: region-code
  version: '1.33'

iam:
  withOIDC: true

nodeGroups:
  - name: ng-ubuntu
    instanceType: m5.large
    desiredCapacity: 3
    amiFamily: Ubuntu2204
    iam:
       attachPolicyARNs:
          - arn:aws: iam::aws:policy/AmazonEKSWorkerNodePolicy
```

```
          - arn:aws: iam::aws:policy/AmazonEC2ContainerRegistryReadOnly
          - arn:aws: iam::aws:policy/AmazonSSMManagedInstanceCore
          - arn:aws: iam::aws:policy/AmazonEKS_CNI_Policy
    ssh:
        allow: true
        publicKeyName: my-ec2-keypair-name
EOF
```

Pour créer un groupe de nœuds Ubuntu Pro, il suffit de remplacer la valeur `amiFamily` par `UbuntuPro2204`.

2. Deployez vos nœuds avec la commande suivante :

```
eksctl create nodegroup --config-file=ubuntu.yaml
```

L'exemple qui suit illustre un résultat.

Plusieurs lignes sont affichées pendant la création des nœuds. L'une des dernières lignes de sortie est similaire à la ligne d'exemple suivante.

```
[#]  created 1 nodegroup(s) in cluster "my-cluster"
```

3. (Facultatif) Déployez un exemple d'application pour tester vos nœuds Ubuntu.
4. Nous recommandons de bloquer l'accès des pods à IMDS si les conditions suivantes sont remplies :
   - Vous prévoyez d'attribuer des rôles IAM à tous vos comptes de service Kubernetes, afin que les pods ne disposent que des permissions strictement nécessaires.
   - Aucun pod du cluster n'a besoin d'accéder au service de métadonnées d' EC2 instance Amazon (IMDS) pour d'autres raisons, telles que la récupération de la région actuelle AWS .

   Pour plus d'informations, consultez Restreindre l'accès au profil d'instance affecté au composant master.

Mettez à jour les nœuds autogérés de votre cluster

Quand une nouvelle AMI optimisée pour Amazon EKS est publiée, pensez à remplacer les nœuds autogérés au sein de votre groupe par cette nouvelle AMI. De même, si vous avez mis à jour la version Kubernetes pour votre cluster Amazon EKS, mettez à jour les nœuds afin qu'ils utilisent la même version de Kubernetes.

> **Important**
>
> Cette rubrique couvre les mises à jour des nœuds pour les groupes de nœuds autogérés. Si vous utilisez des groupes de nœuds gérés, consultez the section called "Mettre à jour".

Deux méthodes de base permettent de mettre à jour les groupes de nœuds autogérés dans vos clusters afin d'utiliser une nouvelle AMI :

Migrer des applications vers un nouveau groupe de nœuds

Créez un nouveau groupe de nœuds et migrez vos pods vers ce groupe. La migration vers un nouveau groupe de nœuds est plus élégante que la simple mise à jour de l'ID AMI dans une pile AWS CloudFormation existante. En effet, le processus de migration rejette l'ancien groupe de nœuds en tant que `NoSchedule` et vide les nœuds une fois qu'une nouvelle pile est prête à accepter la charge de travail Pod existante.

Mettre à jour une pile de nœuds AWS CloudFormation

Mettez à jour la pile AWS CloudFormation d'un groupe de nœuds existant afin d'utiliser la nouvelle AMI. Cette méthode n'est pas prise en charge pour les groupes de nœuds créés avec `eksctl`.

Migrer des applications vers un nouveau groupe de nœuds

Cette rubrique vous explique comment créer un groupe de nœuds, migrer avec élégance vos applications existantes vers ce nouveau groupe, puis supprimer l'ancien groupe de nœuds de votre cluster. Vous pouvez migrer vers un nouveau groupe de nœuds à l'aide de`eksctl` ou de AWS Management Console.

- the section called "`eksctl`"
- the section called "AWS Management Console et AWS CLI"

`eksctl`

Pour migrer vos applications vers un nouveau groupe de nœuds avec **`eksctl`**

Pour plus d'informations sur l'utilisation d'eksctl pour la migration, consultez la section Nœuds non gérés dans la documentation `eksctl`.

Cette procédure nécessite `eksctl` version `0.215.0` ou ultérieure. Vous pouvez vérifier votre version avec la commande suivante :

```
eksctl version
```

Pour les instructions d'installation ou de mise à niveau de `eksctl`, consultez la rubrique Installation dans la documentation `eksctl`.

> **Note**
>
> Cette procédure fonctionne uniquement pour les clusters et les groupes de nœuds créés avec `eksctl`.

1. Récupérez le nom de vos groupes de nœuds existants, en remplaçant *`my-cluster`* par le nom de votre cluster.

   ```
   eksctl get nodegroups --cluster=my-cluster
   ```

   L'exemple qui suit illustre un résultat.

   ```
   CLUSTER      NODEGROUP          CREATED               MIN SIZE      MAX SIZE
    DESIRED CAPACITY     INSTANCE TYPE     IMAGE ID
   default      standard-nodes   2019-05-01T22:26:58Z  1             4             3
                  t3.medium         ami-05a71d034119ffc12
   ```

2. Lancez un nouveau groupe de nœuds avec `eksctl` à l'aide de la commande suivante. Dans la commande, remplacez chaque *`example value`* par vos propres valeurs. Le numéro de version ne peut pas être postérieur à la version Kubernetes pour votre plan de contrôle. De même, il ne peut pas être antérieur de plus de deux versions mineures à la version Kubernetes de votre plan de contrôle. Nous vous recommandons d'utiliser la même version que votre plan de contrôle.

   Nous recommandons de bloquer l'accès des pods à IMDS si les conditions suivantes sont remplies :

   - Vous prévoyez d'attribuer des rôles IAM à tous vos comptes de service Kubernetes, afin que les pods ne disposent que des permissions strictement nécessaires.
   - Aucun pod du cluster n'a besoin d'accéder au service de métadonnées d' EC2 instance Amazon (IMDS) pour d'autres raisons, telles que la récupération de la région actuelle AWS .

Pour plus d'informations, consultez Restreindre l'accès au profil d'instance affecté au composant master.

Pour bloquer l'accès des pods à IMDS, ajoutez l'option `--disable-pod-imds` à la commande suivante.

> **Note**
>
> Pour plus d'indicateurs disponibles et leurs descriptions, consultez https://eksctl.io/.

```
eksctl create nodegroup \
  --cluster my-cluster \
  --version 1.33 \
  --name standard-nodes-new \
  --node-type t3.medium \
  --nodes 3 \
  --nodes-min 1 \
  --nodes-max 4 \
  --managed=false
```

3. Lorsque la commande précédente se termine, vérifiez que tous vos nœuds ont atteint l'état `Ready` à l'aide de la commande suivante :

```
kubectl get nodes
```

4. Supprimez le groupe de nœuds d'origine avec la commande suivante. Dans la commande, remplacez chaque *example value* par les noms de votre cluster et de votre groupe de nœuds :

```
eksctl delete nodegroup --cluster my-cluster --name standard-nodes-old
```

AWS Management Console et AWS CLI

Migrez vos applications vers un nouveau groupe de nœuds à l'aide de la AWS CLI AWS Management Console et

1. Lancez un nouveau groupe de nœuds en suivant les étapes décrites dans Créer des nœuds Amazon Linux autogérés.

2. Lorsque la création de votre pile est terminée, sélectionnez la pile dans la console et choisissez Outputs (Sorties).
3. Enregistrez le NodeInstanceRolepour le groupe de nœuds créé. Vous en aurez besoin pour ajouter les nouveaux nœuds Amazon EKS à votre cluster.

> **Note**
>
> Si vous avez attachés des stratégies IAM supplémentaires au rôle IAM de votre ancien groupe de nœuds, attachez ces mêmes stratégies au rôle IAM de votre nouveau groupe de nœuds pour maintenir cette fonctionnalité sur le nouveau groupe. Cela s'applique si vous avez ajouté des autorisations pour le Kubernetes Cluster Autoscaler, par exemple.

4. Mettez à jour les groupes de sécurité pour les deux groupes de nœuds afin qu'ils puissent communiquer entre eux. Pour de plus amples informations, veuillez consulter the section called "Exigences de groupe de sécurité".
   a. Enregistrez le groupe de sécurité IDs pour les deux groupes de nœuds. Ceci est affiché sous forme de NodeSecurityGroupvaleur dans les sorties de la AWS CloudFormation pile.

      Vous pouvez utiliser les commandes AWS CLI suivantes pour obtenir le groupe IDs de sécurité à partir des noms de pile. Dans ces commandes, `oldNodes` il s'agit du nom de la AWS CloudFormation pile de votre ancienne pile de nœuds et `newNodes` du nom de la pile vers laquelle vous effectuez la migration. Remplacez chaque *`example value`* par vos propres valeurs.

      ```
      oldNodes="old_node_CFN_stack_name"
      newNodes="new_node_CFN_stack_name"

      oldSecGroup=$(aws cloudformation describe-stack-resources --stack-name $oldNodes \
      --query 'StackResources[?
      ResourceType==`AWS::EC2::SecurityGroup`].PhysicalResourceId' \
      --output text)
      newSecGroup=$(aws cloudformation describe-stack-resources --stack-name $newNodes \
      --query 'StackResources[?
      ResourceType==`AWS::EC2::SecurityGroup`].PhysicalResourceId' \
      --output text)
      ```

   b. Ajoutez des règles d'entrée pour chaque groupe de sécurité des nœuds afin d'accepter le trafic de l'autre.

Les commandes AWS CLI suivantes ajoutent des règles entrantes à chaque groupe de sécurité qui autorisent tout le trafic sur tous les protocoles en provenance de l'autre groupe de sécurité. Cette configuration permet aux pods au sein de chaque groupe de nœuds peuvent ainsi communiquer entre eux pendant que vous migrez la charge de travail vers le nouveau groupe.

```
aws ec2 authorize-security-group-ingress --group-id $oldSecGroup \
--source-group $newSecGroup --protocol -1
aws ec2 authorize-security-group-ingress --group-id $newSecGroup \
--source-group $oldSecGroup --protocol -1
```

5. Modifiez la configmap `aws-auth` afin de mapper le nouveau rôle d'instance des nœuds dans RBAC.

```
kubectl edit configmap -n kube-system aws-auth
```

Ajoutez une nouvelle entrée `mapRoles` pour le nouveau groupe de nœuds.

```
apiVersion: v1
data:
  mapRoles: |
    - rolearn: ARN of instance role (not instance profile)
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes>
    - rolearn: arn:aws: iam::111122223333:role/nodes-1-16-NodeInstanceRole-
U11V27W93CX5
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
```

Remplacez l'*ARN of instance role (not instance profile)*extrait par la NodeInstanceRolevaleur que vous avez enregistrée lors d'une étape précédente. Ensuite, enregistrez et fermez le fichier pour appliquer le configmap mise à jour.

6. Vérifiez le statut de vos nœuds et attendez que vos nouveaux nœuds rejoignent votre cluster et affichent le statut `Ready`.

```
kubectl get nodes --watch
```

7. (Facultatif) Si vous utilisez l'outil Cluster Autoscaler de Kubernetes, réduisez verticalement le déploiement à zéro (0) réplicas afin d'éviter tout conflit entre les actions de mise à l'échelle.

```
kubectl scale deployments/cluster-autoscaler --replicas=0 -n kube-system
```

8. Utilisez la commande suivante pour rejeter chacun des nœuds que vous souhaitez supprimer avec `NoSchedule`. Ceci afin que les nouveaux pods ne soient pas planifiés ou replanifiés sur les nœuds que vous remplacez. Pour plus d'informations, consultez Rejets et tolérances dans la documentation de Kubernetes.

```
kubectl taint nodes node_name key=value:NoSchedule
```

   Si vous mettez à niveau vos nœuds vers une nouvelle version de Kubernetes, vous pouvez identifier et rejeter tous les nœuds d'une version Kubernetes donnée (dans ce cas, `1.31`) avec le snippet de code suivant. Le numéro de version ne peut pas être postérieur à la version Kubernetes de votre plan de contrôle. Il ne peut pas non plus être antérieur de plus de deux versions mineures à la version Kubernetes de votre plan de contrôle. Nous vous recommandons d'utiliser la même version que votre plan de contrôle.

```
K8S_VERSION=1.31
nodes=$(kubectl get nodes -o jsonpath="{.items[?(@.status.nodeInfo.kubeletVersion==
\"v$K8S_VERSION\")].metadata.name}")
for node in ${nodes[@]}
do
    echo "Tainting $node"
    kubectl taint nodes $node key=value:NoSchedule
done
```

9. Déterminez le fournisseur DNS de votre cluster.

```
kubectl get deployments -l k8s-app=kube-dns -n kube-system
```

   L'exemple qui suit illustre un résultat. Ce cluster utilise CoreDNS pour la résolution DNS, mais votre cluster peut renvoyer la valeur `kube-dns` à la place) :

```
NAME      DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
```

```
coredns   1         1         1            1           31m
```

10. Si votre déploiement actuel exécute moins de deux réplicas, dimensionnez le déploiement à deux réplicas. Remplacez *coredns* par `kubedns` si la sortie de votre commande a plutôt renvoyé cette valeur.

```
kubectl scale deployments/coredns --replicas=2 -n kube-system
```

11. Drainez chacun des nœuds que vous souhaitez supprimer de votre cluster à l'aide de la commande suivante :

```
kubectl drain node_name --ignore-daemonsets --delete-local-data
```

Si vous mettez à niveau vos nœuds vers une nouvelle version de Kubernetes, identifiez et videz tous les nœuds d'une version particulière de Kubernetes (dans ce cas,*1.31*) à l'aide de l'extrait de code suivant.

```
K8S_VERSION=1.31
nodes=$(kubectl get nodes -o jsonpath="{.items[?(@.status.nodeInfo.kubeletVersion==
\"v$K8S_VERSION\")].metadata.name}")
for node in ${nodes[@]}
do
    echo "Draining $node"
    kubectl drain $node --ignore-daemonsets --delete-local-data
done
```

12. Une fois que vos anciens nœuds ont été purgés, révoquez les règles d'entrée du groupe de sécurité que vous avez autorisées précédemment. Supprimez ensuite la AWS CloudFormation pile pour mettre fin aux instances.

> **Note**
>
> Si vous avez associé des politiques IAM supplémentaires à votre ancien rôle IAM de groupe de nœuds, par exemple en ajoutant des autorisations pour le Kubernetes Cluster Autoscaler, détachez-les du rôle avant de supprimer votre stack. AWS CloudFormation

a. Révoquez les règles entrantes que vous aviez créées précédemment pour les groupes de sécurité de vos nœuds. Dans ces commandes, `oldNodes` il s'agit du nom de la AWS

CloudFormation pile de votre ancienne pile de nœuds et `newNodes` du nom de la pile vers laquelle vous effectuez la migration.

```
oldNodes="old_node_CFN_stack_name"
newNodes="new_node_CFN_stack_name"

oldSecGroup=$(aws cloudformation describe-stack-resources --stack-name $oldNodes \
--query 'StackResources[?
ResourceType==`AWS::EC2::SecurityGroup`].PhysicalResourceId' \
--output text)
newSecGroup=$(aws cloudformation describe-stack-resources --stack-name $newNodes \
--query 'StackResources[?
ResourceType==`AWS::EC2::SecurityGroup`].PhysicalResourceId' \
--output text)
aws ec2 revoke-security-group-ingress --group-id $oldSecGroup \
--source-group $newSecGroup --protocol -1
aws ec2 revoke-security-group-ingress --group-id $newSecGroup \
--source-group $oldSecGroup --protocol -1
```

b. Ouvrez la AWS CloudFormation console.

c. Sélectionnez votre ancienne pile de nœuds.

d. Sélectionnez Delete (Supprimer).

e. Dans la boîte de dialogue de confirmation Delete stack (Supprimer la pile), choisissez Delete stack (Supprimer la pile).

13. Modifiez la configmap `aws-auth` afin de supprimer l'ancien rôle d'instance des nœuds dans RBAC.

```
kubectl edit configmap -n kube-system aws-auth
```

Supprimez l'entrée `mapRoles` de l'ancien groupe de nœuds.

```
apiVersion: v1
data:
  mapRoles: |
    - rolearn: arn:aws: iam::111122223333:role/nodes-1-16-NodeInstanceRole-
W70725MZQFF8
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
```

```
    - rolearn: arn:aws: iam::111122223333:role/nodes-1-15-NodeInstanceRole-
U11V27W93CX5
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes>
```

Enregistrez et fermez le fichier afin d'appliquer la configmap mise à jour.

14. (Facultatif) Si vous utilisez le composant Cluster Autoscaler de Kubernetes, dimensionnez le déploiement à un réplica.

> **Note**
>
> Vous devez également labeliser votre nouveau groupe Auto Scaling de façon appropriée (par exemple `k8s.io/cluster-autoscaler/enabled,k8s.io/cluster-autoscaler/my-cluster`) et mettre à jour la commande de votre déploiement Cluster Autoscaler afin qu'elle pointe vers le groupe Auto Scaling nouvellement labelisé. Pour plus d'informations, consultez Cluster Autoscaler activé. AWS

```
kubectl scale deployments/cluster-autoscaler --replicas=1 -n kube-system
```

15. (Facultatif) Vérifiez que vous utilisez la dernière version du plug-in CNI Amazon VPC pour Kubernetes. Vous devrez peut-être mettre à jour votre version de CNI pour utiliser les derniers types d'instances pris en charge. Pour de plus amples informations, veuillez consulter the section called “CNI Amazon VPC”.

16. Si votre cluster utilise `kube-dns` pour la résolution DNS (voir [migrate-determine-dns-step]), réduisez horizontalement le déploiement `kube-dns` à un réplica.

```
kubectl scale deployments/kube-dns --replicas=1 -n kube-system
```

Mettre à jour une pile de AWS CloudFormation nœuds

Cette rubrique décrit comment mettre à jour une pile de nœuds AWS CloudFormation autogérés existante avec une nouvelle AMI. Vous pouvez utiliser cette procédure pour mettre à jour vos nœuds vers une nouvelle version de Kubernetes suite à une mise à jour du cluster. Sinon, vous

pouvez effectuer une mise à jour vers la dernière AMI optimisée pour Amazon EKS pour une version Kubernetes existante.

> **Important**
>
> Cette rubrique couvre les mises à jour des nœuds pour les groupes de nœuds autogérés. Pour plus d'informations sur l'utilisation de Simplifier le cycle de vie des nœuds avec des groupes de nœuds gérés, consultez the section called "Mettre à jour".

Le dernier AWS CloudFormation modèle de nœud Amazon EKS par défaut est configuré pour lancer une instance avec la nouvelle AMI dans votre cluster avant de supprimer l'ancienne, une par une. Cette configuration garantit que vous disposez toujours du nombre souhaité d'instances actives de votre groupe Auto Scaling dans votre cluster pendant la mise à jour propagée.

> **Note**
>
> Cette méthode n'est pas prise en charge pour les groupes de nœuds créés avec `eksctl`. Si vous avez créé votre cluster ou votre groupe de nœuds avec `eksctl`, consultez the section called "Migration".

1. Déterminez le fournisseur DNS de votre cluster.

   ```
   kubectl get deployments -l k8s-app=kube-dns -n kube-system
   ```

   L'exemple qui suit illustre un résultat. Ce cluster utilise CoreDNS pour la résolution DNS, mais votre cluster peut renvoyer `kube-dns` à la place. Votre sortie peut être différente selon la version de `kubectl` que vous utilisez.

   ```
   NAME      DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
   coredns   1         1         1            1           31m
   ```

2. Si votre déploiement actuel exécute moins de deux réplicas, dimensionnez le déploiement à deux réplicas. Remplacez *`coredns`* par `kube-dns` si la sortie de votre commande a plutôt renvoyé cette valeur.

   ```
   kubectl scale deployments/coredns --replicas=2 -n kube-system
   ```

3. (Facultatif) Si vous utilisez l'outil Cluster Autoscaler de Kubernetes, réduisez verticalement le déploiement à zéro (0) réplicas afin d'éviter tout conflit entre les actions de mise à l'échelle.

```
kubectl scale deployments/cluster-autoscaler --replicas=0 -n kube-system
```

4. Déterminez le type d'instance et le nombre d'instances souhaité pour votre groupe de nœuds actuel. Vous saisirez ces valeurs ultérieurement lorsque vous mettrez à jour le AWS CloudFormation modèle du groupe.
   a. Ouvrez la EC2 console Amazon à l'adresse https://console.aws.amazon.com/ec2/.
   b. Dans le panneau de navigation de gauche, sélectionnez Launch Configurations (Configurations du lancement) et notez le type d'instance pour votre configuration actuelle de lancement des nœuds.
   c. Dans le panneau de navigation de gauche, sélectionnez Auto Scaling Groups (Groupes Auto Scaling) et notez le nombre d'instances souhaité pour le groupe Auto Scaling actuel des nœuds.
5. Ouvrez la AWS CloudFormation console.
6. Sélectionnez votre pile de groupe de nœuds, puis choisissez Update (Mettre à jour).
7. Sélectionnez Replace current template (Remplacer le modèle actuel) et sélectionnez Amazon S3 URL (URL Amazon S3).
8. Pour l'URL Amazon S3, collez l'URL suivante dans la zone de texte pour vous assurer que vous utilisez la dernière version du AWS CloudFormation modèle de nœud. Ensuite, sélectionnez Next (Suivant) :

```
https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2022-12-23/amazon-eks-nodegroup.yaml
```

9. Sur la page Specify stack details (Spécifier les détails de la pile), renseignez les paramètres suivants, puis choisissez Next (Suivant) :
   - NodeAutoScalingGroupDesiredCapacity— Entrez le nombre d'instances souhaité que vous avez enregistré lors d'une étape précédente. Ou saisissez le nouveau nombre de nœuds souhaité pour la mise à l'échelle lorsque votre pile sera mise à jour.
   - NodeAutoScalingGroupMaxSize— Entrez le nombre maximum de nœuds auxquels votre groupe Auto Scaling de nœuds peut s'étendre. Cette valeur doit être supérieure d'au moins un nœud à votre capacité souhaitée. Ceci afin de pouvoir effectuer une mise à jour continue de vos nœuds sans réduire votre nombre de nœuds pendant la mise à jour.
   - NodeInstanceType— Choisissez le type d'instance que vous avez enregistré à l'étape précédente. Vous pouvez également choisir un type d'instance différent pour vos nœuds.

Avant de choisir un autre type d'instance, consultez Choisir un type d'instance de EC2 nœud Amazon optimal. Chaque type d' EC2 instance Amazon prend en charge un nombre maximum d'interfaces réseau élastiques (interface réseau) et chaque interface réseau prend en charge un nombre maximum d'adresses IP. Étant donné que chaque nœud de travail et chaque pod se voient attribuer sa propre adresse IP, il est important de choisir un type d'instance qui prendra en charge le nombre maximum de pods que vous souhaitez exécuter sur chaque EC2 nœud Amazon. Pour une liste du nombre d'interfaces réseau et d'adresses IP pris en charge par les types d'instances, consultez adresses IP par interface réseau et par type d'instance. Par exemple, le type d'instance `m5.large` prend en charge un maximum de 30 adresses IP pour le composant master et les pods.

> **Note**
>
> Les types d'instances pris en charge pour la dernière version du plugin CNI d'Amazon VPC pour Kubernetes sont indiqués dans vpc_ip_resource_limit.go sur GitHub. Vous devrez peut-être mettre à jour votre plug-in CNI Amazon VPC pour la version Kubernetes afin d'utiliser les derniers types d'instance pris en charge. Pour de plus amples informations, veuillez consulter the section called "CNI Amazon VPC".

> **Important**
>
> Certains types d'instances peuvent ne pas être disponibles dans toutes les AWS régions.

- NodeImageIdSSMParam— Le paramètre Amazon EC2 Systems Manager de l'ID AMI vers lequel vous souhaitez effectuer la mise à jour. La valeur suivante utilise la dernière AMI optimisée Amazon EKS pour la version `1.33` de Kubernetes.

```
/aws/service/eks/optimized-ami/1.33/amazon-linux-2/recommended/image_id
```

Vous pouvez le *`1.33`* remplacer par une version de plateforme identique. Ou bien, elle doit être antérieure d'une version au maximum à la version Kubernetes exécutée sur votre plan de contrôle. Nous vous recommandons de conserver vos nœuds dans la même version que votre plan de contrôle. Vous pouvez également le remplacer *`amazon-linux-2`* par un autre type d'AMI. Pour de plus amples informations, veuillez consulter the section called "Recevez les dernières nouvelles IDs".

> **Note**
>
> L'utilisation du paramètre Amazon EC2 Systems Manager vous permet de mettre à jour vos nœuds à l'avenir sans avoir à rechercher et à spécifier un ID AMI. Si votre AWS CloudFormation stack utilise cette valeur, toute mise à jour de stack lance toujours la dernière AMI optimisée Amazon EKS recommandée pour la version de Kubernetes que vous avez spécifiée. C'est le cas même si vous ne modifiez aucune valeur dans le modèle.

- NodeImageId— Pour utiliser votre propre AMI personnalisée, entrez l'ID de l'AMI à utiliser.

> **Important**
>
> Cette valeur remplace toute valeur spécifiée pour NodeImageIdSSMParam. Si vous souhaitez utiliser la NodeImageIdSSMParamvaleur, assurez-vous que la valeur pour NodeImageIdest vide.

- Désactiver IMDSv1 : par défaut, chaque nœud prend en charge les versions 1 du service de métadonnées d'instance (IMDSv1) et IMDSv2. Cependant, vous pouvez le désactiver IMDSv1. Sélectionnez true si vous ne souhaitez pas que les nœuds ou les pods planifiés dans le groupe de nœuds soient utilisés IMDSv1. Pour de plus amples informations au sujet d'IMDS, consultez Configuration du service des métadonnées d'instance. Si vous avez implémenté des rôles IAM pour les comptes de service, attribuez les autorisations nécessaires directement à tous les pods nécessitant un accès aux AWS services. Ainsi, aucun pod de votre cluster n'a besoin d'accéder à l'IMDS pour d'autres raisons, telles que la récupération de la région actuelle AWS . Ensuite, vous pouvez également désactiver l'accès IMDSv2 aux pods qui n'utilisent pas le réseau hôte. Pour plus d'informations, consultez Restreindre l'accès au profil d'instance affecté au composant master.

10. (Facultatif) Sur la page Options (Options), étiquetez les ressources de votre pile. Choisissez Suivant.
11. Sur la page Review (Vérification), vérifiez vos informations, acceptez que la pile puisse créer des ressources IAM, puis choisissez Update stack (Mettre à jour la pile).

> **Note**
>
> La mise à jour de chaque nœud du cluster prend plusieurs minutes. Attendez que la mise à jour de tous les nœuds soit terminée avant d'effectuer les étapes suivantes.

12. Si le fournisseur DNS de votre cluster est `kube-dns`, réduisez horizontalement le déploiement `kube-dns` à une réplica.

```
kubectl scale deployments/kube-dns --replicas=1 -n kube-system
```

13. (Facultatif) Si vous utilisez le composant Cluster Autoscaler de Kubernetes, redimensionnez le déploiement à la quantité souhaitée de réplicas.

```
kubectl scale deployments/cluster-autoscaler --replicas=1 -n kube-system
```

14. (Facultatif) Vérifiez que vous utilisez la dernière version du plug-in CNI Amazon VPC pour Kubernetes. Vous devrez peut-être mettre à jour votre plug-in CNI Amazon VPC pour la version Kubernetes afin d'utiliser les derniers types d'instance pris en charge. Pour de plus amples informations, veuillez consulter the section called "CNI Amazon VPC".

Simplifiez la gestion des calculs avec AWS Fargate

Cette rubrique porte sur l'utilisation d'Amazon EKS pour exécuter les pods Kubernetes sur AWS Fargate. Fargate est une technologie qui fournit une capacité de calcul à la demande et de taille appropriée pour les conteneurs. Avec Fargate, il n'est pas nécessaire d'allouer, de configurer ou de mettre à l'échelle des groupes de machines virtuelles pour exécuter les conteneurs. Vous n'avez plus à choisir les types de serveurs, à décider quand vos clusters de nœuds doivent être mis à l'échelle, ni à optimiser les packs de clusters.

Vous pouvez contrôler les pods qui démarrent sur Fargate et la manière dont ils s'exécutent avec des profils Fargate. Les profils Fargate sont définis dans le cadre de votre cluster Amazon EKS. Amazon EKS intègre Kubernetes à Fargate à l'aide des contrôleurs créés par AWS en utilisant le modèle extensible en amont fourni par Kubernetes. Ces contrôleurs font partie du plan de contrôle Kubernetes géré par Amazon EKS et sont responsables de la planification des pods Kubernetes natifs sur Fargate. Les contrôleurs Fargate comprennent un nouveau planificateur qui s'exécute parallèlement au planificateur Kubernetes par défaut, en plus de plusieurs contrôleurs d'admission mutants et validants. Lorsque vous démarrez un pod qui répond aux critères d'exécution sur Fargate,

les contrôleurs Fargate qui fonctionnent dans le cluster reconnaissent, mettent à jour et planifient le pod sur Fargate.

Cette rubrique décrit les différents composants des pods qui s'exécutent sur Fargate, et appelle à des considérations particulières pour l'utilisation de Fargate avec Amazon EKS.

Considérations relatives à AWS Fargate

Voici quelques éléments à prendre en compte pour l'utilisation de Fargate sur Amazon EKS.

- Chaque pod fonctionnant sur Fargate a sa propre limite de calcul. Les nœuds ne partagent pas le noyau sous-jacent, les ressources CPU, les ressources mémoire ou l'interface réseau Elastic avec un autre pod.
- Les Network Load Balancers et les Application Load Balancers (ALB) peuvent être utilisés avec Fargate avec des cibles IP uniquement. Pour plus d'informations, consultez the section called "Créer un équilibreur de charge de réseau" et the section called "Répartition de la charge des applications".
- Les services exposés de Fargate s'exécutent uniquement en mode IP de type cible, et non en mode IP de nœud. La manière recommandée de vérifier la connectivité entre un service s'exécutant sur un nœud géré et un service s'exécutant sur Fargate est de se connecter via le nom du service.
- Les pods doivent correspondre à un profil Fargate au moment où ils sont programmés pour fonctionner sur Fargate. Les pods qui ne correspondent pas à un profil Fargate peuvent être bloqués comme `Pending`. Si un profil Fargate correspondant existe, vous pouvez supprimer les pods en attente que vous avez créés pour les reprogrammer sur Fargate.
- Les DaemonSets ne sont pas pris en charge sur Fargate. Si votre application nécessite un démon, reconfigurez ce démon pour qu'il s'exécute en tant que conteneur secondaire dans vos pods.
- Les conteneurs privilégiés ne sont pas pris en charge sur Fargate.
- Les pods fonctionnant sur Fargate ne peuvent pas spécifier `HostPort` ou `HostNetwork` dans le manifeste du pod.
- La limite logicielle par défaut `nofile` et `nproc` est de 1 024 et la limite matérielle est de 65 535 pour les pods Fargate.
- Les GPU ne sont actuellement pas disponibles sur Fargate.
- Les pods qui fonctionnent sur Fargate ne sont pris en charge que sur des sous-réseaux privés (avec un accès aux services AWS par passerelle NAT, mais sans route directe vers une passerelle

Internet). Le VPC de votre cluster doit donc disposer de sous-réseaux privés. Pour les clusters sans accès Internet sortant, veuillez consulter the section called "Clusters privés".

- Vous pouvez utiliser la fonctionnalité Ajuster les ressources des pods avec Vertical Pod Autoscaler pour définir la taille initiale correcte du processeur et de la mémoire pour vos pods Fargate, puis utiliser la fonctionnalité Mettre à l'échelle les déploiements de pods avec Horizontal Pod Autoscaler pour mettre à l'échelle ces pods. Si vous souhaitez que le Vertical Pod Autoscaler redéploie automatiquement les pods vers Fargate avec des combinaisons de CPU et de mémoire plus importantes, définissez le mode de Vertical Pod Autoscale sur `Auto` ou `Recreate` pour garantir une fonctionnalité correcte. Pour plus d'informations, consultez la documentation Vertical Pod Autoscaler (VPA) sur GitHub.
- La résolution DNS et les noms d'hôte DNS doivent être activés pour votre VPC. Pour plus d'informations, consultez Affichage et mise à jour de la prise en charge de DNS pour votre VPC.
- Amazon EKS Fargate ajoute une défense approfondie aux applications Kubernetes en isolant chaque pod au sein d'une machine virtuelle (VM). Cette frontière VM empêche l'accès aux ressources basées sur l'hôte utilisées par d'autres pods en cas de fuite du conteneur, qui est une méthode courante d'attaque des applications conteneurisées et d'accès aux ressources en dehors du conteneur.

  L'utilisation d'Amazon EKS ne modifie pas vos responsabilités dans le cadre du modèle de responsabilité partagée. Vous devez examiner attentivement la configuration des contrôles de sécurité et de gouvernance du cluster. Le moyen le plus sûr d'isoler une application est toujours de l'exécuter dans un cluster séparé.
- Les profils Fargate prennent en charge la spécification de sous-réseaux à partir de blocs CIDR secondaires de VPC. Vous pouvez spécifier un bloc CIDR secondaire. En effet, le nombre d'adresses IP disponibles dans un sous-réseau est limité. Par conséquent, il existe également un nombre limité de pods qui peuvent être créés dans le cluster. En utilisant différents sous-réseaux pour les pods, vous pouvez augmenter le nombre d'adresses IP disponibles. Pour plus d'informations, consultez Ajout de blocs d'adresse CIDR IPv4 à un VPC.
- Le service de métadonnées d'instance Amazon EC2 (IMDS) n'est pas disponible pour les pods qui sont déployés sur des nœuds Fargate. Si vous avez des pods déployés sur Fargate qui nécessitent des informations d'identification IAM, attribuez-les à vos pods à l'aide des rôles IAM pour les comptes de service. Si vos pods ont besoin d'accéder à d'autres informations disponibles via IMDS, vous devez coder ces informations en dur dans la spécification de votre pod. Cela inclut la région AWS ou la zone de disponibilité dans laquelle un pod est déployé.
- Vous ne pouvez pas déployer de pods Fargate sur AWS Outposts, AWS Wavelength ou AWS Local Zones.

- Amazon EKS doit régulièrement appliquer des correctifs aux pods Fargate pour assurer leur sécurité. Les tentatives de mises à jour sont réalisées de manière à réduire les potentielles répercussions ; toutefois, il arrive que des pods doivent être supprimés si leur expulsion a échoué. Certaines actions peuvent être prises pour minimiser les perturbations. Pour de plus amples informations, consultez the section called "Événements d'application de correctifs au système d'exploitation".
- Le pluginAmazon VPC CNI pour Amazon EKS est installé sur des nœuds Fargate. Vous ne pouvez pas utiliser les plugins Alternate CNI pour les clusters Amazon EKS avec des nœuds Fargate.
- Un Pod fonctionnant sur Fargate monte automatiquement un système de fichiers Amazon EFS, sans nécessiter d'étapes manuelles d'installation de pilotes. Vous ne pouvez pas utiliser l'approvisionnement dynamique des volumes persistants avec les nœuds Fargate, mais vous pouvez utiliser l'approvisionnement statique.
- Amazon EKS ne prend pas en charge Fargate Spot.
- Vous ne pouvez pas monter des volumes Amazon EBS sur des pods Fargate.
- Vous pouvez exécuter le contrôleur Amazon EBS CSI sur les nœuds Fargate, mais le DaemonSet du nœud Amazon EBS CSI ne peut fonctionner que sur les instances Amazon EC2.
- Une fois qu'une tâche Kubernetes est marquée `Completed` ou `Failed`, les pods créés par la tâche continuent généralement d'exister. Ce comportement vous permet de consulter vos journaux et vos résultats, mais avec Fargate, vous devrez payer des frais si vous ne nettoyez pas le travail après coup.

  Pour supprimer automatiquement les pods associés après l'exécution ou l'échec d'une tâche, vous pouvez spécifier une durée à l'aide du contrôleur TTL (time-to-live). L'exemple suivant montre comment spécifier `.spec.ttlSecondsAfterFinished` dans votre manifeste de tâche.

```
apiVersion: batch/v1
kind: Job
metadata:
  name: busybox
spec:
  template:
    spec:
      containers:
      - name: busybox
        image: busybox
        command: ["/bin/sh", "-c", "sleep 10"]
      restartPolicy: Never
```

```
  ttlSecondsAfterFinished: 60 # <-- TTL controller
```

Tableau de comparaison de Fargate

Critères	AWS Fargate
Peut être déployé dans AWS Outposts	Non
Possibilité d'être déployé dans une zone locale AWS	Non
Possibilité d'exécuter des conteneurs qui nécessitent Windows	Non
Possibilité d'exécuter des conteneurs qui nécessitent Linux	Oui
Possibilité d'exécuter des applications nécessitant la puce Inferentia	Non
Possibilité d'exécuter des applications nécessitant un GPU	Non
Possibilité d'exécuter des applications nécessitant des processeurs Arm	Non
Possibilité d'exécuter AWS Bottlerocket	Non
Les pods partagent un environnement d'exécution du noyau avec d'autres pods	Non : chaque pod a un noyau dédié
Les pods partagent le processeur, la mémoire, le stockage et les ressources réseau avec d'autres pods.	Non : chaque pod dispose de ressources dédiées et peut être dimensionné indépendamment pour optimiser l'utilisation des ressources.

Critères	AWS Fargate
Les pods peuvent utiliser plus de matériel et de mémoire que demandé dans les spécifications des pods	Non : le pod peut toutefois être redéployé à l'aide d'une plus grande configuration de vCPU et de mémoire.
Doit déployer et gérer des instances Amazon EC2	Non
Doit sécuriser, entretenir et corriger le système d'exploitation des instances Amazon EC2	Non
Peut fournir des arguments bootstrap lors du déploiement d'un nœud, tels que des arguments kubelet supplémentaires.	Non
Possibilité d'affecter des adresses IP à des pods à partir d'un bloc d'adresse CIDR différent de l'adresse IP affectée au nœud.	Non
Possibilité d'accéder au nœud via SSH	Non : il n'y a pas de système d'exploitation hôte de nœud auquel se connecter par SSH.
Possibilité de déployer votre propre AMI personnalisée sur les nœuds	Non
Possibilité de déployer votre propre CNI personnalisée sur les nœuds	Non
Vous devez mettre à jour l'AMI du nœud par vous-même	Non
Vous devez mettre à jour la version du nœud Kubernetes par vous-même	Non : vous ne gérez pas les nœuds.
Possibilité d'utiliser le stockage Amazon EBS avec des pods	Non

Critères	AWS Fargate
Possibilité d'utiliser le stockage Amazon EFS avec des pods	Oui
Possibilité d'utiliser le stockage Amazon FSx pour Lustre avec des pods	Non
Possibilité d'utiliser le Network Load Balancer pour les services	Oui, lors de l'utilisation de la fonction Créer un Network Load Balancer
Les pods peuvent s'exécuter dans un sous-rése au public	Non
Possibilité d'affecter différents groupes de sécurité VPC à des pods individuels	Oui
Possibilité d'exécuter des ensembles de démons Kubernetes	Non
Prend en charge `HostPort` et `HostNetwork` dans le manifeste du pod	Non
AWS : disponibilité dans les régions	Quelques régions prises en charge par Amazon EKS
Possibilité d'exécuter des conteneurs sur des hôtes dédiés Amazon EC2	Non
Tarification	Coût d'une mémoire Fargate individuelle et d'une configuration CPU. Chaque pod a son propre coût. Pour plus d'informations, consultez Tarification de AWS Fargate.

Commencez à utiliser AWS Fargate pour votre cluster

Cette rubrique explique comment commencer à exécuter des pods sur AWS Fargate avec votre cluster Amazon EKS.

Si vous limitez l'accès au point de terminaison public de votre cluster à l'aide de blocs CIDR, nous vous recommandons d'activer également l'accès au point de terminaison privé. De cette manière, les pods Fargate peuvent communiquer avec le cluster. Sans l'activation du point de terminaison privé, les blocs CIDR que vous spécifiez pour l'accès public doivent inclure les sources de sortie de votre VPC. Pour de plus amples informations, veuillez consulter the section called "Accès au point de terminaison de cluster".

Prérequis

Un cluster existant. Si vous ne disposez pas déjà d'un cluster Amazon EKS, consultez Démarrer.

Étape 1 : s'assurer que les nœuds existants peuvent communiquer avec les pods Fargate

Si vous travaillez avec un nouveau cluster sans nœuds ou un cluster comportant uniquement des groupes de nœuds gérés (voir the section called "Groupes de nœuds gérés"), vous pouvez passer à the section called "Étape 2 : créer un rôle d'exécution Fargate Pod".

Supposons que vous travaillez avec un cluster existant ayant déjà des nœuds associés. Assurez-vous que les pods sur ces nœuds peuvent communiquer librement avec les pods qui s'exécutent sur Fargate. Les pods qui s'exécutent sur Fargate sont automatiquement configurés pour utiliser le groupe de sécurité du cluster auquel ils sont associés. Assurez-vous que tous les nœuds existants dans votre cluster peuvent envoyer et recevoir du trafic vers et depuis le groupe de sécurité du cluster. Les groupes de nœuds gérés sont automatiquement configurés pour utiliser également le groupe de sécurité du cluster. Vous n'avez donc pas besoin de les modifier ou de vérifier leur compatibilité (voir the section called "Groupes de nœuds gérés").

Pour les groupes de nœuds existants créés avec `eksctl` ou avec les AWS CloudFormation modèles gérés par Amazon EKS, vous pouvez ajouter le groupe de sécurité du cluster aux nœuds manuellement. Vous pouvez également modifier le modèle de lancement du groupe Auto Scaling pour le groupe de nœuds, afin d'attacher le groupe de sécurité du cluster aux instances. Pour plus d'informations, consultez Modification des groupes de sécurité d'une instance dans le Guide de l'utilisateur VPC Amazon.

Vous pouvez rechercher un groupe de sécurité pour votre cluster dans la AWS Management Console section Mise en réseau du cluster. Vous pouvez également le faire à l'aide de la commande AWS CLI suivante. Lorsque vous utilisez cette commande, remplacez `<my-cluster>` par le nom de votre cluster.

```
aws eks describe-cluster --name <my-cluster> --query
 cluster.resourcesVpcConfig.clusterSecurityGroupId
```

Étape 2 : créer un rôle d'exécution Fargate Pod

Lorsque votre cluster crée des pods sur AWS Fargate, les composants qui s'exécutent sur l'infrastructure Fargate doivent effectuer des appels en votre nom. AWS APIs Pour ce faire, le rôle d'exécution de pod Amazon EKS fournit les autorisations IAM. Pour créer un rôle d'exécution AWS Fargate Pod, consultez. the section called "Rôle IAM d'exécution de pod"

> **Note**
>
> Si vous avez créé votre cluster avec `eksctl` à l'aide de l'option `--fargate`, votre cluster comporte déjà un rôle d'exécution de pod que vous pouvez trouver dans la console IAM en utilisant `eksctl-my-cluster-FargatePodExecutionRole-ABCDEFGHIJKL`. De même, si vous utilisez `eksctl` pour créer vos profils Fargate, `eksctl` créera votre rôle d'exécution de pod si celui-ci n'existe pas déjà.

Étape 3 : créer un profil Fargate pour votre cluster

Avant de pouvoir planifier des pods s'exécutant sur Fargate dans votre cluster, vous devez définir un profil Fargate qui spécifie quels pods utilisent Fargate lorsqu'ils sont lancés. Pour de plus amples informations, veuillez consulter the section called "Définir des profils".

> **Note**
>
> Si vous avez créé votre cluster avec `eksctl` à l'aide de l'option `--fargate`, un profil Fargate est déjà créé pour votre cluster avec des sélecteurs pour tous les pods dans les espaces de noms `kube-system` et `default`. Utilisez la procédure suivante pour créer des profils Fargate pour tout autre espace de noms que vous souhaitez utiliser avec Fargate.

Vous pouvez créer un profil Fargate en utilisant l'un des outils suivants :

- the section called "`eksctl`"
- the section called "AWS Management Console"

eksctl

Cette procédure nécessite `eksctl` version `0.215.0` ou ultérieure. Vous pouvez vérifier votre version avec la commande suivante :

```
eksctl version
```

Pour les instructions d'installation ou de mise à niveau de `eksctl`, consultez la rubrique Installation dans la documentation `eksctl`.

Pour créer un profil Fargate avec **eksctl**

Créez votre profil Fargate avec la commande `eksctl` suivante, en remplaçant chaque `<example value>` par vos propres valeurs. Vous devez spécifier un espace de noms. Cependant, l'option `--labels` n'est pas obligatoire.

```
eksctl create fargateprofile \
    --cluster <my-cluster> \
    --name <my-fargate-profile> \
    --namespace <my-kubernetes-namespace> \
    --labels <key=value>
```

Vous pouvez utiliser certains caractères génériques pour les étiquettes `<my-kubernetes-namespace>` et `<key=value>`. Pour de plus amples informations, veuillez consulter the section called "Caractères génériques du profil Fargate".

AWS Management Console

Pour créer un profil Fargate avec AWS Management Console

1. Ouvrez la console Amazon EKS.
2. Choisissez le cluster pour lequel vous voulez créer un profil Fargate.
3. Choisissez l'onglet Calcul.
4. Sous Fargate profiles (Profils Fargate), choisissez Add Fargate profile (Ajouter un profil Fargate).
5. Sur la page Configure Fargate profile (Configurer le profil Fargate), procédez comme suit :
   a. Dans Nom, saisissez un nom pour votre profil Fargate. Le nom doit être unique.
   b. Pour le rôle d'exécution du pod, choisissez le rôle d'exécution du pod à utiliser avec votre profil Fargate. Seuls les rôles IAM avec le principal de service `eks-fargate-`

`pods.amazonaws.com` sont affichés. Si vous ne voyez aucun rôle répertorié ici, vous devez en créer un. Pour de plus amples informations, veuillez consulter the section called "Rôle IAM d'exécution de pod".

c. Modifiez les sous-réseaux sélectionnés selon vos besoins.

> **Note**
>
> Seuls les sous-réseaux privés sont pris en charge pour les pods qui s'exécutent sur Fargate.

d. Dans Identifications, vous pouvez éventuellement étiqueter votre profil Fargate. Ces balises ne se propagent pas aux autres ressources qui sont associées au profil, comme les pods.

e. Choisissez Suivant.

6. Sur la page Configurer la sélection de pod, procédez comme suit :

a. Pour l'espace de noms, saisissez un espace de noms correspondant aux pods.

- Vous pouvez utiliser des espaces de noms spécifiques pour les faire correspondre, tels que `kube-system` ou `default`.
- Vous pouvez utiliser certains caractères génériques (par exemple, `prod-*`) pour faire correspondre plusieurs espaces de noms (par exemple, `prod-deployment` et `prod-test`). Pour de plus amples informations, veuillez consulter the section called "Caractères génériques du profil Fargate".

b. (Facultatif) Ajoutez des labels Kubernetes au sélecteur. Ajoutez-les spécifiquement à celui auquel les pods de l'espace de noms spécifié doivent correspondre.

- Vous pouvez ajouter la balise `infrastructure: fargate` au sélecteur afin que seuls les pods de l'espace de noms spécifié qui possèdent également la balise Kubernetes `infrastructure: fargate` correspondent au sélecteur.
- Vous pouvez utiliser certains caractères génériques (par exemple, `key?: value?`) pour faire correspondre plusieurs espaces de noms (par exemple, `keya: valuea` et `keyb: valueb`). Pour de plus amples informations, veuillez consulter the section called "Caractères génériques du profil Fargate".

c. Choisissez Suivant.

7. Sur la page Vérifier et créer, vérifiez les informations de votre profil Fargate et choisissez Créer.

Étape 4 : mettre à jour CoreDNS

Par défaut, CoreDNS est configuré pour s'exécuter sur l' EC2 infrastructure Amazon sur des clusters Amazon EKS. Si vous souhaitez uniquement exécuter vos pods sur Fargate dans votre cluster, effectuez les étapes suivantes.

> **Note**
>
> Si vous avez créé votre cluster à l'aide de `eksctl` en utilisant l'option `--fargate`, vous pouvez passer directement à the section called “Étapes suivantes”.

1. Créez un profil Fargate pour CoreDNS à l'aide de la commande suivante. Remplacez-le `<my-cluster>` par le nom de votre cluster, `<111122223333>` par votre identifiant de compte, `<AmazonEKSFargatePodExecutionRole>` par le nom de votre rôle d'exécution de Pod et `<000000000000000c>` par le nom IDs de vos sous-réseaux privés. `<000000000000000a>` `<000000000000000b>` Si vous ne disposez pas d'un rôle d'exécution Pod, vous devez d'abord en créer un (voir the section called “Étape 2 : créer un rôle d'exécution Fargate Pod”).

   > **Important**
   >
   > L'ARN de rôle ne peut pas inclure un chemin d'accès autre que `/`. Par exemple, si le nom de votre rôle est `development/apps/AmazonEKSFargatePodExecutionRole`, vous devez le remplacer par `AmazonEKSFargatePodExecutionRole` lorsque vous spécifiez l'ARN du rôle. Le format de l'ARN de rôle doit être `arn:aws: iam::<111122223333>:role/<AmazonEKSFargatePodExecutionRole>`.

```
aws eks create-fargate-profile \
    --fargate-profile-name coredns \
    --cluster-name <my-cluster> \
    --pod-execution-role-arn arn:aws: iam::<111122223333>:role/
<AmazonEKSFargatePodExecutionRole> \
    --selectors namespace=kube-system,labels={k8s-app=kube-dns} \
    --subnets subnet-<000000000000000a> subnet-<000000000000000b> subnet-
<000000000000000c>
```

2. Déclenchez le déploiement du déploiement `coredns`.

```
kubectl rollout restart -n kube-system deployment coredns
```

Étapes suivantes

- Vous pouvez commencer à migrer vos applications existantes pour les exécuter sur Fargate avec le flux suivant.
  a. the section called "Créer un profil Fargate" qui correspond à l'espace de noms Kubernetes et aux étiquettes Kubernetes de votre application.
  b. Supprimez et recréez tous les pods existants afin qu'ils soient planifiés sur Fargate. Modifiez le `<namespace>` et `<deployment-type>` pour mettre à jour vos pods spécifiques.

  ```
  kubectl rollout restart -n <namespace> deployment <deployment-type>
  ```
- Déployez the section called "Répartition de la charge des applications" pour autoriser les objets d'entrée pour vos pods fonctionnant sur Fargate.
- Vous pouvez utiliser le the section called "Vertical Pod Autoscaler (VPA)" pour définir la taille initiale correcte du processeur et de la mémoire pour vos pods Fargate, puis utiliser le the section called "Horizontal Pod Autoscaler" pour faire évoluer ces pods. Si vous souhaitez que Vertical Pod Autoscaler redéploie automatiquement les pods vers Fargate avec des combinaisons de CPU et de mémoire plus élevées, définissez le mode de Vertical Pod Autoscaler sur `Auto` ou `Recreate`. Ceci permet de garantir une fonctionnalité correcte. Pour plus d'informations, consultez la documentation Vertical Pod Autoscaler sur GitHub.
- Vous pouvez configurer le collecteur AWS Distro pour OpenTelemetry (ADOT) pour la surveillance des applications en suivant ces instructions.

Définissez quels pods utilisent AWS Fargate lors de leur lancement

Avant de planifier des pods sur Fargate dans votre cluster, vous devez définir au moins un profil Fargate qui spécifie quels pods utilisent Fargate lors de leur lancement.

En tant qu'administrateur, vous pouvez utiliser un profil Fargate pour déclarer quels pods s'exécutent sur Fargate. Vous pouvez faire cela via les sélecteurs du profil. Vous pouvez ajouter jusqu'à cinq sélecteurs à chaque profil. Chaque sélecteur doit contenir un espace de noms. Le sélecteur peut également inclure des étiquettes. Le champ de label se compose de plusieurs paires clé-valeur facultatives. Les pods qui correspondent à un sélecteur sont programmés sur Fargate. Les pods

sont associés à l'aide d'un espace de noms et des étiquettes spécifiées dans le sélecteur. Si un sélecteur d'espace de noms est défini sans étiquettes, Amazon EKS tente de planifier tous les pods qui s'exécutent dans cet espace de noms sur Fargate à l'aide du profil. Si un pod à planifier correspond à l'un des sélecteurs du profil Fargate, ce pod est planifié sur Fargate.

Si un pod correspond à plusieurs profils Fargate, vous pouvez spécifier le profil utilisé par le pod en ajoutant le label Kubernetes suivant à la spécification du pod : `eks.amazonaws.com/fargate-profile: my-fargate-profile`. Le Pod doit correspondre à un sélecteur dans ce profil pour être planifié sur Fargate. Les règles d'affinité/anti-affinité Kubernetes ne s'appliquent pas et ne sont pas nécessaires avec les pods Amazon EKS Fargate.

Lorsque vous créez un profil Fargate, vous devez spécifier un rôle d'exécution de pod. Ce rôle d'exécution concerne les composants Amazon EKS qui s'exécutent sur l'infrastructure Fargate utilisant le profil. Il est ajouté au contrôle d'accès basé sur les rôles (RBAC) Kubernetes du cluster pour l'autorisation. Ainsi, le `kubelet` qui est exécuté sur l'infrastructure Fargate peut s'enregistrer dans votre cluster Amazon EKS et apparaître dans votre cluster comme un nœud. Le rôle d'exécution du pod fournit également des autorisations IAM à l'infrastructure Fargate pour permettre un accès en lecture aux référentiels d'images Amazon ECR. Pour de plus amples informations, consultez the section called "Rôle IAM d'exécution de pod".

Les profils Fargate ne peuvent pas être modifiés. Toutefois, vous pouvez créer un nouveau profil mis à jour pour remplacer un profil existant, puis supprimer l'original.

> **Note**
>
> Tous les pods qui s'exécutent à l'aide d'un profil Fargate sont arrêtés et placés en attente lorsque le profil est supprimé.

Si tous les profils Fargate d'un cluster ont l'état `DELETING`, vous devez attendre que ce profil Fargate soit définitivement supprimé avant de pouvoir créer d'autres profils dans ce cluster.

> **Note**
>
> Fargate ne prend actuellement pas en charge Kubernetes TopologySpreadConstraints.

Amazon EKS et Fargate répartissent les pods entre chacun des sous-réseaux définis dans le profil Fargate. Cependant, vous risquez de vous retrouver avec une répartition inégale. Si vous

avez besoin d'une répartition uniforme, utilisez deux profils Fargate. Une répartition uniforme est importante dans les scénarios où vous souhaitez déployer deux réplicas sans aucune durée d'indisponibilité. Nous vous recommandons de n'avoir qu'un seul sous-réseau pour chaque profil.

Composants de profil Fargate

Les composants suivants sont contenus dans un profil Fargate.

Rôle d'exécution du pod

Lorsque votre cluster crée des pods sur AWS Fargate, le `kubelet` qui s'exécute sur l'infrastructure Fargate doit effectuer des appels vers les API AWS en votre nom. Par exemple, il doit effectuer des appels pour extraire des images de conteneur à partir d'Amazon ECR. Pour ce faire, le rôle d'exécution de pod Amazon EKS fournit les autorisations IAM.

Lorsque vous créez un profil Fargate, vous devez spécifier un rôle d'exécution de pods à utiliser avec vos pods. Ce rôle est ajouté au contrôle d'accès basé sur les rôles (RBAC) Kubernetes du cluster à des fins d'autorisation. Ainsi, le `kubelet` exécuté sur l'infrastructure Fargate peut s'enregistrer dans votre cluster Amazon EKS et apparaître dans votre cluster comme un nœud. Pour de plus amples informations, consultez the section called "Rôle IAM d'exécution de pod".

Sous-réseaux

Les ID des sous-réseaux poury lancer des pods utilisent ce profil. Pour l'instant, les pods fonctionnant sur Fargate n'ont pas d'adresse IP publique. Par conséquent, seuls les sous-réseaux privés sans route directe vers une passerelle Internet sont acceptés pour ce paramètre.

Sélecteurs

Les sélecteurs à faire correspondre pour que les pods utilisent ce profil Fargate. Vous pouvez spécifier jusqu'à cinq sélecteurs dans un profil Fargate. Les sélecteurs intègrent les composants suivants :

- Espace de noms : vous devez spécifier un espace de noms pour un sélecteur. Le sélecteur ne correspond qu'aux pods créés dans cet espace de noms. Vous pouvez toutefois créer plusieurs sélecteurs pour cibler plusieurs espaces de noms.
- Labels : vous pouvez éventuellement spécifier des labels Kubernetes à faire correspondre pour le sélecteur. Le sélecteur ne correspond qu'aux pods qui possèdent tous les labels spécifiés dans le sélecteur.

Caractères génériques du profil Fargate

En plus des caractères autorisés par Kubernetes, vous pouvez utiliser `*` et `?` dans les critères de sélection pour les espaces de noms, les clés d'étiquette et les valeurs d'étiquette :

- `*` représente aucun, un ou plusieurs caractères. Par exemple, `prod*` peut représenter `prod` et `prod-metrics`.
- `?` représente un caractère unique (par exemple, `value?` peut représenter `valuea`). Cependant, il ne peut pas représenter `value` et `value-a`, parce que `?` ne peut représenter qu'un seul et unique caractère.

Ces caractères génériques peuvent être utilisés dans n'importe quelle position et en combinaison (par exemple, `prod*`, `*dev` et `frontend*?`). Les autres caractères génériques et formes de correspondance de modèles, tels que les expressions régulières, ne sont pas pris en charge.

S'il existe plusieurs profils correspondants pour l'espace de noms et les étiquettes dans la spécification du pod, Fargate sélectionne le profil avec un tri alphanumérique par nom de profil. Par exemple, si les deux profils A (avec le nom `beta-workload`) et le profil B (avec le nom `prod-workload`) ont des sélecteurs correspondants pour les pods à lancer, Fargate choisit le profil A (`beta-workload`) pour les pods. Les pods ont des étiquettes avec le profil A sur les pods (par exemple, `eks.amazonaws.com/fargate-profile=beta-workload`).

Si vous souhaitez migrer des pods Fargate existants vers de nouveaux profils qui utilisent des caractères génériques, vous pouvez procéder de deux manières :

- Créez un nouveau profil avec les sélecteurs correspondants, puis supprimez les anciens profils. Les pods étiquetés avec d'anciens profils sont reprogrammés vers de nouveaux profils correspondants.
- Si vous souhaitez migrer des charges de travail mais que vous ne savez pas quelles étiquettes Fargate figurent sur chaque pod Fargate, vous pouvez utiliser la méthode suivante. Créez un nouveau profil avec un nom qui trie d'abord par ordre alphanumérique parmi les profils du même cluster. Recyclez ensuite les pods Fargate qui doivent être migrés vers de nouveaux profils.

Créer un profil Fargate

Cette section décrit comment créer un profil Fargate. Vous devez également avoir créé un rôle d'exécution de pod à utiliser pour votre profil Fargate. Pour de plus amples informations, consultez

the section called "Rôle IAM d'exécution de pod". Les pods qui s'exécutent sur Fargate ne sont pris en charge que sur les sous-réseaux privés avec accès NAT Gateway aux services AWS, mais pas sur une route directe vers une passerelle Internet. Le VPC de votre cluster doit donc disposer de sous-réseaux privés.

Vous pouvez créer un profil avec les éléments suivants :

- the section called "`eksctl`"
- the section called "AWS Management Console"

`eksctl`

Pour créer un profil Fargate avec **`eksctl`**

Créez votre profil Fargate à l'aide de la commande `eksctl` suivante, en remplaçant chaque valeur d'exemple par vos propres valeurs. Vous devez spécifier un espace de noms. Cependant, l'option `--labels` n'est pas obligatoire.

```
eksctl create fargateprofile \
    --cluster my-cluster \
    --name my-fargate-profile \
    --namespace my-kubernetes-namespace \
    --labels key=value
```

Vous pouvez utiliser certains caractères génériques pour les étiquettes `my-kubernetes-namespace` et `key=value`. Pour de plus amples informations, consultez the section called "Caractères génériques du profil Fargate".

AWS Management Console

Pour créer un profil Fargate avec AWS Management Console

1. Ouvrez la console Amazon EKS.
2. Choisissez le cluster pour lequel vous voulez créer un profil Fargate.
3. Choisissez l'onglet Calcul.
4. Sous Fargate profiles (Profils Fargate), choisissez Add Fargate profile (Ajouter un profil Fargate).
5. Sur la page Configure Fargate profile (Configurer le profil Fargate), procédez comme suit :
   a. Pour Nom, saisissez un nom unique pour votre profil Fargate, tel que `my-profile`.

b. Pour le rôle d'exécution du pod, choisissez le rôle d'exécution du pod à utiliser avec votre profil Fargate. Seuls les rôles IAM avec le principal de service `eks-fargate-pods.amazonaws.com` sont affichés. Si vous ne voyez aucun rôle répertorié ici, vous devez en créer un. Pour de plus amples informations, consultez the section called "Rôle IAM d'exécution de pod".

c. Modifiez les sous-réseaux sélectionnés selon vos besoins.

> **Note**
>
> Seuls les sous-réseaux privés sont pris en charge pour les pods qui s'exécutent sur Fargate.

d. Dans Identifications, vous pouvez éventuellement étiqueter votre profil Fargate. Ces balises ne se propagent pas aux autres ressources qui sont associées au profil, comme les pods.

e. Choisissez Suivant.

6. Sur la page Configurer la sélection de pod, procédez comme suit :

a. Pour l'espace de noms, saisissez un espace de noms correspondant aux pods.

- Vous pouvez utiliser des espaces de noms spécifiques pour les faire correspondre, tels que `kube-system` ou `default`.
- Vous pouvez utiliser certains caractères génériques (par exemple, `prod-*`) pour faire correspondre plusieurs espaces de noms (par exemple, `prod-deployment` et `prod-test`). Pour de plus amples informations, consultez the section called "Caractères génériques du profil Fargate".

b. (Facultatif) Ajoutez des labels Kubernetes au sélecteur. Ajoutez-les spécifiquement à celui auquel les pods de l'espace de noms spécifié doivent correspondre.

- Vous pouvez ajouter la balise `infrastructure: fargate` au sélecteur afin que seuls les pods de l'espace de noms spécifié qui possèdent également la balise Kubernetes `infrastructure: fargate` correspondent au sélecteur.
- Vous pouvez utiliser certains caractères génériques (par exemple, `key?: value?`) pour faire correspondre plusieurs espaces de noms (par exemple, `keya: valuea` et `keyb: valueb`). Pour de plus amples informations, consultez the section called "Caractères génériques du profil Fargate".

c. Choisissez Suivant.

7. Sur la page Vérifier et créer, vérifiez les informations de votre profil Fargate et choisissez Créer.

Supprimer un profil Fargate

Cette rubrique décrit comment supprimer un profil Fargate. Lorsque vous supprimez un profil Fargate, tous les pods qui étaient programmés sur Fargate avec ce profil sont supprimés. Si ces pods correspondent à un autre profil Fargate, ils sont programmés sur Fargate avec ce profil. S'ils ne correspondent plus à aucun profil Fargate, ils ne sont pas planifiés sur Fargate et peuvent rester en attente.

Un seul profil Fargate d'un cluster peut avoir le statut `DELETING` à la fois. Vous devez attendre que la suppression d'un profil Fargate soit terminée pour pouvoir supprimer un autre profil de ce cluster.

Vous pouvez supprimer un profil à l'aide de l'un des outils suivants :

- the section called "`eksctl`"
- the section called "AWS Management Console"
- the section called "AWS CLI"

`eksctl`

Supprimer un profil Fargate avec **`eksctl`**

Utilisez la commande suivante pour supprimer un profil d'un cluster. Remplacez la *`valeur d'exemple`* par vos propres valeurs.

```
eksctl delete fargateprofile  --name my-profile --cluster my-cluster
```

AWS Management Console

Supprimer un profil Fargate avec AWS Management Console

1. Ouvrez la console Amazon EKS.
2. Dans le volet de navigation de gauche, choisissez Clusters. Dans la liste des clusters, sélectionnez celui dont vous souhaitez supprimer le profil Fargate.
3. Choisissez l'onglet Calcul.
4. Sélectionnez le profil Fargate à supprimer, puis l'option Delete (Supprimer).
5. Sur la page Delete Fargate profile (Supprimer le profil Fargate), saisissez le nom du profil, puis sélectionnez Delete (Supprimer).

AWS CLI

Supprimer un profil Fargate avec l'interface CLI AWS

Utilisez la commande suivante pour supprimer un profil d'un cluster. Remplacez la *valeur d'exemple* par vos propres valeurs.

```
aws eks delete-fargate-profile --fargate-profile-name my-profile --cluster-name my-cluster
```

Comprendre les détails de configuration des pods Fargate

Cette section décrit certains des détails de configuration uniques des pods pour l'exécution de pods Kubernetes sur AWS Fargate.

CPU et mémoire d'un pod

Avec Kubernetes, vous pouvez définir des demandes, un nombre minimal de ressources vCPU et mémoire qui sont allouées à chaque conteneur dans un pod. Les pods sont planifiés par Kubernetes pour garantir qu'au moins les ressources demandées pour chaque pod sont disponibles sur la ressource de calcul. Pour plus d'informations, consultez Gestion des ressources de calcul des conteneurs dans la documentation Kubernetes.

> **Note**
>
> Étant donné qu'Amazon EKS Fargate n'exécute qu'un seul pod par nœud, le scénario d'expulsion des pods en cas de ressources insuffisantes ne se produit pas. Tous les pods Amazon EKS Fargate s'exécutent avec une priorité garantie, de sorte que le CPU et la mémoire demandés doivent être égaux à la limite pour tous les conteneurs. Pour plus d'informations, consultez Configurer la qualité de service pour les pods dans la documentation Kubernetes.

Lorsque les pods sont planifiés sur Fargate, les réservations de vCPU et de mémoire dans la spécification du pod déterminent la quantité de CPU et de mémoire à allouer au pod.

- La demande maximale de tous les conteneurs Init est utilisée pour déterminer les besoins en vCPU et en mémoire de la demande Init.

- Les demandes de tous les conteneurs à longue durée d'exécution sont additionnées pour déterminer les besoins en vCPU et en mémoire de la demande à longue durée d'exécution.
- La plus grande des deux valeurs ci-dessus est choisie pour la demande de vCPU et la demande de mémoire à utiliser pour votre pod.
- Fargate ajoute 256 Mo à la réservation de mémoire de chaque pod pour les composants Kubernetes requis (`kubelet`, `kube-proxy`, et `containerd`).

Fargate arrondit à la configuration informatique suivante qui correspond le mieux à la somme des demandes de vCPU et de mémoire afin de garantir que les pods disposent toujours des ressources nécessaires à leur fonctionnement.

Si vous ne spécifiez pas de combinaison vCPU et mémoire, la plus petite combinaison disponible est utilisée (0,25 vCPU et 0,5 Go de mémoire).

Le tableau suivant répertorie les combinaisons de vCPU et de mémoire qui sont disponibles pour les pods fonctionnant sur Fargate.

Valeur vCPU	Valeur de mémoire
0,25 vCPU	0,5 Go, 1 Go, 2 Go
0,5 vCPU	1 Go, 2 Go, 3 Go, 4 Go
1 vCPU	2 Go, 3 Go, 4 Go, 5 Go, 6 Go, 7 Go, 8 Go
2 vCPU	Entre 4 Go et 16 Go par incrément de 1 Go
4 vCPU	Entre 8 Go et 30 Go par incrément de 1 Go
8 vCPU	Entre 16 Go et 60 Go par incréments de 4 Go
16 vCPU	Entre 32 Go et 120 Go par incréments de 8 Go

La mémoire supplémentaire réservée aux composants Kubernetes peut entraîner l'approvisionnement d'une tâche Fargate avec plus de vCPU que demandé. Par exemple, une demande pour 1 vCPU et 8 Go de mémoire verra 256 Mo ajoutés à sa demande de mémoire, et approvisionnera une tâche Fargate avec 2 vCPU et 9 Go de mémoire, puisqu'aucune tâche avec 1 vCPU et 9 Go de mémoire n'est disponible.

Il n'existe pas de corrélation entre la taille du pod fonctionnant sur Fargate et la taille du nœud signalée par Kubernetes avec `kubectl get nodes`. La taille du nœud signalée est souvent supérieure à la capacité du pod. Vous pouvez vérifier la capacité du pod avec la commande suivante. Remplacez *`default`* par l'espace de noms de votre pod et *`pod-name`* par le nom de votre pod.

```
kubectl describe pod --namespace default pod-name
```

L'exemple qui suit illustre un résultat.

```
[...]
annotations:
    CapacityProvisioned: 0.25vCPU 0.5GB
[...]
```

L'annotation `CapacityProvisioned` représente la capacité du pod appliquée et détermine le coût de votre pod fonctionnant sur Fargate. Pour plus d'informations sur la tarification de ces configurations de calcul, reportez-vous à Tarification AWS.

Stockage Fargate

Un Pod fonctionnant sur Fargate monte automatiquement un système de fichiers Amazon EFS, sans nécessiter d'étapes manuelles d'installation de pilotes. Vous ne pouvez pas utiliser l'approvisionnement dynamique des volumes persistants avec les nœuds Fargate, mais vous pouvez utiliser l'approvisionnement statique. Pour plus d'informations, voir Pilote CSI Amazon EFS sur GitHub.

Une fois provisionné, chaque pod exécuté sur Fargate reçoit par défaut un espace de stockage éphémère de 20 GiB. Ce type de stockage est supprimé après l'arrêt du pod. Lorsque de nouveaux pods sont lancés sur Fargate, le chiffrement du volume de stockage éphémère est activé par défaut. Le stockage éphémère des pods est chiffré avec un algorithme de chiffrement AES-256 utilisant des clés gérées par AWS Fargate.

> **Note**
>
> La capacité de stockage par défaut pour les pods Amazon EKS exécutés sur Fargate est inférieure à 20 GiB. En effet, une partie de l'espace est utilisée par le module `kubelet` et d'autres modules Kubernetes chargés dans le pod.

Vous pouvez augmenter la capacité totale de stockage éphémère jusqu'à un maximum de 175 GiB. Pour configurer la taille avec Kubernetes, spécifiez les demandes de ressources `ephemeral-storage` pour chaque conteneur dans un pod. Lorsque Kubernetes planifie les pods, il s'assure que la somme des demandes de ressources pour chaque pod est inférieure à la capacité de la tâche Fargate. Pour plus d'informations, consultez la section Gestion des ressources pour les pods et les conteneurs dans la documentation Kubernetes.

Amazon EKS Fargate alloue une capacité de stockage éphémère supérieure à celle demandée aux fins de l'utilisation du système. Par exemple, une demande de 100 GiB allouera à une tâche Fargate une capacité de stockage éphémère de 115 GiB.

Définir des actions pour les événements de correction du système d'exploitation AWS Fargate

Amazon EKS doit régulièrement appliquer des correctifs au système d'exploitation pour les nœuds AWS Fargate afin d'assurer leur sécurité. Dans le cadre du processus d'application des correctifs, nous recyclons les nœuds pour installer les correctifs du système d'exploitation. Les tentatives de mises à jour sont réalisées de manière à avoir des répercussions minimales sur vos services. Toutefois, si les pods ne sont pas expulsés avec succès, ils doivent parfois être supprimés. Voici les actions que vous pouvez effectuer pour minimiser les perturbations potentielles :

- Définissez des budgets de perturbation de pods (PDB) appropriés pour contrôler le nombre de pods simultanément hors service.
- Créez des règles Amazon EventBridge pour réagir aux expulsions ayant échoué avant la suppression des pods.
- Redémarrez manuellement vos pods concernés avant la date d'expulsion indiquée dans la notification que vous avez reçue.
- Créez une configuration de notification dans Notifications d'utilisateur AWS.

Amazon EKS travaille en étroite collaboration avec la communauté Kubernetes pour corriger les bugs et appliquer les correctifs de sécurité le plus rapidement possible. Tous les pods Fargate démarrent sur la version la plus récente du correctif Kubernetes, disponible auprès d'Amazon EKS pour la version Kubernetes de votre cluster. Si un pod présente une ancienne version de correctif, Amazon EKS peut le recycler pour le mettre à jour à la dernière version. Cela garantit que vos pods comprennent les dernières mises à jour de sécurité. De cette façon, en cas de problème critique inhérent aux vulnérabilités et expositions courantes (CVE), vous êtes tenu à jour pour réduire les risques de sécurité.

Lorsque le système d'exploitation AWS Fargate est mis à jour, Amazon EKS vous envoie une notification indiquant les ressources concernées et la date des prochaines expulsions de pods. Si la date d'expulsion indiquée ne vous convient pas, vous avez la possibilité de redémarrer manuellement vos pods concernés avant la date d'expulsion indiquée dans la notification. Tous les pods créés avant le moment où vous recevez la notification sont susceptibles d'être expulsés. Consultez la documentation Kubernetes pour obtenir des instructions supplémentaires sur la manière de redémarrer manuellement vos pods.

Pour limiter le nombre de pods hors service simultanément lors du recyclage des pods, vous pouvez définir des budgets de perturbation des pods (PDB). Vous pouvez utiliser les PDB pour définir une disponibilité minimale en fonction des exigences de chacune de vos applications, tout en autorisant les mises à jour. La disponibilité minimale de votre PDB doit être inférieure à 100 %. Pour plus d'informations, consultez la section Spécification d'un budget de perturbation pour votre application dans la documentation Kubernetes.

Amazon EKS utilise l'API d'expulsion pour vider le pod en toute sécurité tout en respectant les PDB que vous avez définis pour l'application. Les pods sont expulsés par zone de disponibilité pour minimiser les répercussions. Si l'expulsion réussit, le nouveau pod reçoit le dernier correctif et aucune autre action n'est requise.

Lorsque l'expulsion d'un pod échoue, Amazon EKS envoie un événement à votre compte avec des informations sur les pods dont l'expulsion a échoué. Vous pouvez agir en fonction du message avant l'heure de fin prévue. Le temps spécifique varie en fonction de l'urgence du correctif. Le moment venu, Amazon EKS tente à nouveau d'expulser les pods. Toutefois, si l'expulsion échoue lors de cette tentative, un nouvel événement n'est pas envoyé. Si l'expulsion échoue à nouveau, vos pods existants sont supprimés périodiquement afin que les nouveaux pods puissent comporter le dernier correctif.

Vous trouverez ci-dessous un exemple d'événement reçu lorsque l'expulsion d'un pod échoue. Il contient des informations sur le cluster, le nom du pod, l'espace de noms du pod, le profil Fargate et l'heure de fin planifiée.

```
{
    "version": "0",
    "id": "12345678-90ab-cdef-0123-4567890abcde",
    "detail-type": "EKS Fargate Pod Scheduled Termination",
    "source": "aws.eks",
    "account": "111122223333",
    "time": "2021-06-27T12:52:44Z",
```

```
    "region": "region-code",
    "resources": [
        "default/my-database-deployment"
    ],
    "detail": {
        "clusterName": "my-cluster",
        "fargateProfileName": "my-fargate-profile",
        "podName": "my-pod-name",
        "podNamespace": "default",
        "evictErrorMessage": "Cannot evict pod as it would violate the pod's disruption
 budget",
        "scheduledTerminationTime": "2021-06-30T12:52:44.832Z[UTC]"
    }
}
```

En outre, si plusieurs PDB sont associés à un pod, un événement d'échec d'expulsion peut survenir. Cet événement renvoie le message d'erreur suivant.

```
"evictErrorMessage": "This pod has multiple PodDisruptionBudget, which the eviction
 subresource does not support",
```

Vous pouvez créer une action souhaitée en fonction de cet événement. Par exemple, vous pouvez ajuster votre budget de perturbation de pod (PDB) pour contrôler l'expulsion des pods. Plus précisément, imaginions que vous utilisiez un PDB qui spécifie le pourcentage cible de pods disponibles. Avant que la résiliation de vos pods soit forcée lors d'une mise à niveau, vous pouvez ajuster le PDB à un pourcentage différent de pods. Pour recevoir cet événement, vous devez créer une règle Amazon EventBridge dans le compte AWS et la région AWS auxquels le cluster appartient. La règle doit utiliser le modèle personnalisé suivant. Pour plus d'informations, veuillez consulter Création d'une règle Amazon EventBridge qui réagit aux événements dans le Guide de l'utilisateur Amazon EventBridge.

```
{
  "source": ["aws.eks"],
  "detail-type": ["EKS Fargate Pod Scheduled Termination"]
}
```

Une cible appropriée peut être définie pour que l'événement le capture. Pour obtenir une liste complète des cibles disponibles, consultez Cibles Amazon EventBridge dans le Guide de l'utilisateur Amazon EventBridge. Vous pouvez également créer une configuration de notification dans Notifications d'utilisateurs AWS. Lorsque vous utilisez le AWS Management Console pour créer la

notification, sous Règles d'événement, choisissez Elastic Kubernetes Service (EKS) comme nom de service AWS et EKS Fargate Pod Scheduled Termination comme Type d'événement. Pour plus d'informations, consultez la section Prise en main des notifications d'utilisateur AWS dans le Guide de l'utilisateur des notifications d'utilisateur AWS.

Consultez la FAQ : Avis d'expulsion de pod Fargate dans re:Post AWS pour connaître les questions fréquemment posées concernant les expulsions de pod EKS.

Collectez l' AWS application Fargate et les statistiques d'utilisation

Vous pouvez collecter des métriques du système et des métriques CloudWatch d'utilisation pour AWS Fargate.

Métriques d'application

Pour les applications exécutées sur Amazon EKS et AWS Fargate, vous pouvez utiliser Distro OpenTelemetry for (AWS ADOT). ADOT vous permet de collecter des métriques du système et de les envoyer aux tableaux de bord de CloudWatch Container Insights. Pour démarrer avec ADOT pour les applications exécutées sur Fargate, consultez la section CloudWatch Utilisation de Container Insights AWS with Distro OpenTelemetry dans la documentation ADOT.

Métriques d'utilisation

Vous pouvez utiliser les statistiques CloudWatch d'utilisation pour obtenir une visibilité sur l'utilisation des ressources par votre compte. Utilisez ces indicateurs pour visualiser l'utilisation actuelle de vos services sur CloudWatch des graphiques et des tableaux de bord.

AWS Les métriques d'utilisation de Fargate correspondent aux AWS quotas de service. Vous pouvez configurer des alarmes qui vous alertent lorsque votre utilisation approche un quota de service. Pour de plus amples informations sur les Service Quotas pour Fargate, consultez the section called "Quotas de service".

AWS Fargate publie les métriques suivantes dans l'espace de noms. `AWS/Usage`

Métrique	Description
`ResourceCount`	Nombre total des ressources spécifiées exécutées sur votre compte. La ressource

Métrique	Description
	est définie par les dimensions associées à la métrique.

Les dimensions suivantes sont utilisées pour affiner les métriques d'utilisation publiées par AWS Fargate.

Dimension	Description
`Service`	Nom du AWS service contenant la ressource . Pour les métriques AWS d'utilisation de Fargate, la valeur de cette dimension est. `Fargate`
`Type`	Le type d'entité qui fait l'objet du rapport. Actuellement, la seule valeur valide pour les métriques d' AWS utilisation de Fargate est. `Resource`
`Resource`	Le type de ressource en cours d'exécution. Actuellement, AWS Fargate renvoie des informations sur votre utilisation de Fargate On-Demand. La valeur de ressource pour l'utilisa tion de Fargate On-Demand est `OnDemand`. [NOTE] ==== L'utilisation de Fargate On-Demand combine les pods Amazon EKS utilisant Fargate, les tâches Amazon ECS utilisant le type de lancement Fargate et les tâches Amazon ECS utilisant le fournisseur de capacité `FARGATE`. ====

Dimension	Description
`Class`	Classe de ressource suivie. Actuellement, AWS Fargate n'utilise pas la dimension de classe.

Création d'une CloudWatch alarme pour surveiller les métriques d'utilisation des ressources de Fargate

AWS Fargate CloudWatch fournit des métriques d'utilisation qui correspondent aux quotas de service pour AWS l'utilisation des ressources Fargate On-Demand. Dans la console Service Quotas, vous pouvez visualiser votre utilisation sur un graphique. Vous pouvez également configurer des alarmes qui vous alertent lorsque votre utilisation approche d'un quota de service. Pour de plus amples informations, veuillez consulter the section called “Collecter les métriques”.

Suivez les étapes ci-dessous pour créer une CloudWatch alarme basée sur les métriques d'utilisation des ressources de Fargate.

1. Ouvrez la console Service Quotas à l'adresse https://console.aws.amazon.com/servicequotas/.
2. Dans le volet de navigation de gauche, sélectionnez AWS services.
3. Dans la liste des AWS services, recherchez et sélectionnez AWS Fargate.
4. Dans la liste Quotas de service, sélectionnez le quota d'utilisation Fargate pour lequel vous souhaitez créer une alarme.
5. Dans la section CloudWatch Alarmes Amazon, choisissez Create.
6. Pour Alarm threshold (Seuil d'alarme), choisissez le pourcentage de la valeur de quota appliquée que vous souhaitez définir comme valeur d'alarme.
7. Pour Nom de l'alarme, saisissez un nom pour l'alarme, puis choisissez Créer.

Démarrez la journalisation AWS Fargate pour votre cluster

Amazon EKS on Fargate propose un routeur de journal intégré basé sur Fluent Bit. Cela signifie que vous n’exécutez pas explicitement un conteneur Fluent Bit comme sidecar, mais qu’Amazon l’exécute pour vous. Tout ce que vous avez à faire est de configurer le routeur de journaux. La configuration se fait par le biais d'un `ConfigMap` qui doit répondre aux critères suivants :

- Nommé `aws-logging`
- Créé dans un espace de noms dédié appelé `aws-observability`

- Ne doit pas dépasser 5 300 caractères.

Une fois que vous avez créé le `ConfigMap`, Amazon EKS on Fargate le détecte automatiquement et configure le routeur de journaux avec lui. Fargate utilise une version de AWS pour Fluent Bit, une distribution compatible en amont de Fluent Bit gérée par AWS. Pour plus d'informations, consultez AWS for Fluent Bit sur GitHub.

Le routeur de journaux vous permet d'utiliser la diversité des services sur AWSpour l'analyse des journaux et le stockage. Vous pouvez diffuser les journaux depuis Fargate directement vers Amazon CloudWatch, Amazon OpenSearch Service. Vous pouvez également diffuser les journaux vers des destinations telles qu'Amazon S3, Amazon Kinesis Data Streams et des outils partenaires via Amazon Data Firehose.

- Un profil Fargate existant qui spécifie un espace de noms Kubernetes existant dans lequel vous déployez des pods Fargate. Pour de plus amples informations, consultez the section called "Étape 3 : créer un profil Fargate pour votre cluster".
- Un rôle d'exécution de pod Fargate existant. Pour de plus amples informations, consultez the section called "Étape 2 : créer un rôle d'exécution Fargate Pod".

Configuration du routeur de journaux

> **Important**
>
> Pour que les journaux soient publiés avec succès, il doit y avoir un accès réseau depuis le VPC dans lequel se trouve votre cluster vers la destination des journaux. Cela concerne principalement les utilisateurs qui personnalisent les règles de sortie pour leur VPC. Pour un exemple d'utilisation de CloudWatch, consultez la section Utilisation des journaux CloudWatch avec les points de terminaison d'un VPC d'interface dans le Guide de l'utilisateur Amazon CloudWatch Logs.

Dans les étapes suivantes, remplacez chaque *`valeur d'exemple`* par vos propres valeurs.

1. Créez un espace de noms Kubernetes dédié nommé `aws-observability`.
   a. Enregistrez le contenu suivant dans un fichier nommé `aws-observability-namespace.yaml` sur votre ordinateur. La valeur pour `name` doit être `aws-observability` et le label `aws-observability: enabled` est obligatoire.

```
kind: Namespace
apiVersion: v1
metadata:
  name: aws-observability
  labels:
    aws-observability: enabled
```

b. Créez l'espace de noms.

```
kubectl apply -f aws-observability-namespace.yaml
```

2. Créez un `ConfigMap` avec une valeur de données `Fluent Conf` pour envoyer les journaux des conteneurs vers une destination. Fluent Conf est Fluent Bit, qui est un langage de configuration de processeur de journaux rapide et léger, utilisé pour acheminer les journaux des conteneurs vers une destination de journalisation de votre choix. Pour plus d'informations, consultez Fichier de configuration dans la documentation Fluent Bit.

> **Important**
>
> Dans une `Fluent Conf` type, les principales sections incluses sont `Service`, `Input`, `Filter` et `Output`. Le routeur de journaux Fargate n'accepte cependant que :
>
> - Les sections `Filter` et `Output`.
> - Une section `Parser`.
>
> Si vous fournissez d'autres sections, elles seront rejetées.

Le routeur de journal Fargate gère les sections `Service` et `Input`. Il comporte la section `Input` suivante, qui ne peut pas être modifiée et n'est pas nécessaire dans votre `ConfigMap`. Cependant, vous pouvez en tirer des informations, telles que la limite de la mémoire tampon et la balise appliquée pour les journaux.

```
[INPUT]
    Name tail
    Buffer_Max_Size 66KB
    DB /var/log/flb_kube.db
    Mem_Buf_Limit 45MB
    Path /var/log/containers/*.log
    Read_From_Head On
```

```
    Refresh_Interval 10
    Rotate_Wait 30
    Skip_Long_Lines On
    Tag kube.*
```

Lors de la création du `ConfigMap`, prenez en compte les règles suivantes que Fargate utilise pour valider les champs :

- `[FILTER]`, `[OUTPUT]` et `[PARSER]` sont censés être spécifiés sous chaque clé correspondante. Par exemple, `[FILTER]` doit être inférieur à `filters.conf`. Vous pouvez avoir un ou plusieurs `[FILTER]` sous `filters.conf`. `[OUTPUT]` et `[PARSER]` doivent également être sous leurs clés correspondantes. En spécifiant plusieurs sections `[OUTPUT]`, vous pouvez acheminer vos journaux vers différentes destinations en même temps.
- Fargate valide les clés requises de chaque section. `Name` et `match` sont nécessaires pour chaque `[FILTER]` et `[OUTPUT]`. `Name` et `format` sont nécessaires pour chaque `[PARSER]`. Ces noms sont sensibles à la casse.
- Les variables d'environnement telles que `${ENV_VAR}` ne sont pas autorisées dans le `ConfigMap`.
- L'indentation doit être la même pour la directive ou la paire clé-valeur dans chaque `filters.conf`, `output.conf` et `parsers.conf`. Les paires clé-valeur doivent être indentées plus que les directives.
- Fargate valide par rapport aux filtres pris en charge suivants : `grep`, `parser`, `record_modifier`, `rewrite_tag`, `throttle`, `nest`, `modify` et `kubernetes`.
- Fargate valide par rapport à la sortie prise en charge suivante : `es`, `firehose`, `kinesis_firehose`, `cloudwatch`, `cloudwatch_logs` et `kinesis`.
- Au moins un plug-in `Output` doit être fourni dans le `ConfigMap` pour activer la journalisation. `Filter` et `Parser` ne sont pas nécessaires pour activer la journalisation.

  Vous pouvez également exécuter Fluent Bit sur Amazon EC2 en utilisant la configuration souhaitée pour résoudre les problèmes qui surviennent lors de la validation. Créez votre `ConfigMap` en utilisant l'un des exemples suivants.

  > **Important**
  >
  > La journalisation Amazon EKS Fargate ne prend pas en charge la configuration dynamique d'un `ConfigMap`. Les modifications apportées à un `ConfigMap` sont

appliquées uniquement aux nouveaux pods. Les modifications ne sont pas appliquées aux pods existants.

Créez un `ConfigMap` en utilisant l'exemple pour votre destination de journal désirée.

> **Note**
>
> Vous pouvez également utiliser Amazon Kinesis Data Streams comme destination du journal. Si vous utilisez Kinesis Data Streams, assurez-vous que l'autorisation `kinesis:PutRecords` a été accordée au rôle d'exécution du pod. Pour plus d'informations, consultez Permissions d'Amazon Kinesis Data Streams dans le Manuel officiel Fluent Bit.

Example

CloudWatch

Vous disposez de deux options de sortie lorsque vous utilisez CloudWatch :

- Un plugin de sortie écrit en C
- Un plugin de sortie écrit en Golang

L'exemple suivant explique comment utiliser le plugin `cloudwatch_logs` pour envoyer des journaux à CloudWatch.

a. Enregistrez le contenu suivant dans un fichier nommé `aws-logging-cloudwatch-configmap.yaml`. Remplacez *`region-code`* par la région AWS dans laquelle votre cluster se situe. Les paramètres sous `[OUTPUT]` sont requises.

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: aws-logging
  namespace: aws-observability
data:
  flb_log_cw: "false"  # Set to true to ship Fluent Bit process logs to
 CloudWatch.
  filters.conf: |
    [FILTER]
        Name parser
```

```
        Match *
        Key_name log
        Parser crio
    [FILTER]
        Name kubernetes
        Match kube.*
        Merge_Log On
        Keep_Log Off
        Buffer_Size 0
        Kube_Meta_Cache_TTL 300s
  output.conf: |
    [OUTPUT]
        Name cloudwatch_logs
        Match   kube.*
        region region-code
        log_group_name my-logs
        log_stream_prefix from-fluent-bit-
        log_retention_days 60
        auto_create_group true
  parsers.conf: |
    [PARSER]
        Name crio
        Format Regex
        Regex ^(?<time>[^ ]+) (?<stream>stdout|stderr) (?<logtag>P|F) (?
<log>.*)$
        Time_Key    time
        Time_Format %Y-%m-%dT%H:%M:%S.%L%z
```

b. Appliquez le manifeste à votre cluster.

```
kubectl apply -f aws-logging-cloudwatch-configmap.yaml
```

Amazon OpenSearch Service

Si vous souhaitez envoyer des journaux à Amazon OpenSearch Service, vous pouvez utiliser es output, un plug-in écrit en C. L'exemple suivant vous montre comment utiliser le plug-in pour envoyer des journaux à OpenSearch.

a. Enregistrez le contenu suivant dans un fichier nommé `aws-logging-opensearch-configmap.yaml`. Remplacez la *`valeur d'exemple`* par vos propres valeurs.

```
kind: ConfigMap
apiVersion: v1
```

```
metadata:
  name: aws-logging
  namespace: aws-observability
data:
  output.conf: |
    [OUTPUT]
      Name  es
      Match *
      Host  search-example-gjxdcilagiprbglqn42jsty66y.region-
code.es.amazonaws.com
      Port  443
      Index example
      Type  example_type
      AWS_Auth On
      AWS_Region region-code
      tls   On
```

b. Appliquez le manifeste à votre cluster.

```
kubectl apply -f aws-logging-opensearch-configmap.yaml
```

Firehose

Vous disposez de deux options de sortie lorsque vous envoyez des journaux à Firehose :

- kinesis_firehose : un plug-in de sortie écrit en C.
- firehose : un plug-in de sortie écrit en Golang.

L'exemple suivant vous montre comment utiliser le plug-in `kinesis_firehose` pour envoyer des journaux à Firehose.

a. Enregistrez le contenu suivant dans un fichier nommé `aws-logging-firehose-configmap.yaml`. Remplacez *region-code* par la région AWS dans laquelle votre cluster se situe.

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: aws-logging
  namespace: aws-observability
data:
  output.conf: |
    [OUTPUT]
```

```
    Name  kinesis_firehose
    Match *
    region region-code
    delivery_stream my-stream-firehose
```

b. Appliquez le manifeste à votre cluster.

```
kubectl apply -f aws-logging-firehose-configmap.yaml
```

3. Configurez les autorisations pour le rôle d'exécution Fargate Pod afin d'envoyer les journaux vers votre destination.

a. Téléchargez la politique IAM pour votre destination sur votre ordinateur.

Example

CloudWatch

Téléchargez la politique IAM CloudWatch sur votre ordinateur. Vous pouvez également afficher la politique sur GitHub.

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-eks-fluent-
logging-examples/mainline/examples/fargate/cloudwatchlogs/permissions.json
```

Amazon OpenSearch Service

Téléchargez la politique IAM OpenSearch sur votre ordinateur. Vous pouvez également afficher la politique sur GitHub.

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-eks-
fluent-logging-examples/mainline/examples/fargate/amazon-elasticsearch/
permissions.json
```

Assurez-vous que le contrôle d'accès OpenSearch Dashboards est correctement configuré. Le `all_access role` dans OpenSearch Dashboards doit avoir le rôle d'exécution du pod Fargate et le rôle IAM mappés. Le même mappage doit être fait pour le rôle `security_manager`. Vous pouvez ajouter les mappages précédents en sélectionnant `Menu`, `Security` et `Roles`, puis sélectionner les rôles correspondants. Pour plus d'informations, consultez Comment dépanner CloudWatch Logs afin qu'il diffuse vers mon domaine Amazon ES ?.

Firehose

Téléchargez la politique IAM Firehose sur votre ordinateur. Vous pouvez également afficher la politique sur GitHub.

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-eks-fluent-logging-examples/mainline/examples/fargate/kinesis-firehose/permissions.json
```

b. Créez une politique IAM à partir du fichier de politique que vous avez téléchargé.

```
aws iam create-policy --policy-name eks-fargate-logging-policy --policy-document
 file://permissions.json
```

c. Attachez la politique IAM au rôle d'exécution de pod spécifié pour votre profil Fargate avec la commande suivante. Remplacez *111122223333* par votre ID de compte. Remplacez *AmazonEKSFargatePodExecutionRole* par votre rôle d'exécution de pod (pour plus d'informations, consultez the section called "Étape 2 : créer un rôle d'exécution Fargate Pod").

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::111122223333:policy/eks-fargate-logging-policy \
  --role-name AmazonEKSFargatePodExecutionRole
```

Prise en charge des filtres Kubernetes

Le filtre Kubernetes de Fluent Bit vous permet d'ajouter des métadonnées Kubernetes à vos fichiers journaux. Pour plus d'informations sur le filtre, consultez Kubernetes dans la documentation Fluent Bit. Vous pouvez appliquer un filtre en utilisant le point de terminaison du serveur d'API.

```
filters.conf: |
    [FILTER]
        Name             kubernetes
        Match            kube.*
        Merge_Log           On
        Buffer_Size         0
        Kube_Meta_Cache_TTL 300s
```

> **Important**
>
> - `Kube_URL`, `Kube_CA_File`, `Kube_Token_Command` et `Kube_Token_File` sont des paramètres de configuration appartenant au service et ne doivent pas être spécifiés. Amazon EKS Fargate remplit ces valeurs.
> - `Kube_Meta_Cache_TTL` est le moment où Fluent Bit attend qu'il communique avec le serveur d'API pour obtenir les dernières métadonnées. Si la valeur `Kube_Meta_Cache_TTL` n'est pas spécifiée, Amazon EKS Fargate ajoute une valeur par défaut de 30 minutes pour diminuer la charge sur le serveur d'API.

Pour envoyer les journaux de processus de Fluent Bit à votre compte

Vous pouvez aussi envoyer les journaux de processus de Fluent Bit à Amazon CloudWatch en utilisant le `ConfigMap` suivant. L'envoi des journaux de processus Fluent Bit à CloudWatch implique des coûts supplémentaires d'ingestion de journaux et de stockage. Remplacez *`region-code`* par la région AWS dans laquelle votre cluster se situe.

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: aws-logging
  namespace: aws-observability
  labels:
data:
  # Configuration files: server, input, filters and output
  # ======================================================
  flb_log_cw: "true"  # Ships Fluent Bit process logs to CloudWatch.

  output.conf: |
    [OUTPUT]
        Name cloudwatch
        Match kube.*
        region region-code
        log_group_name fluent-bit-cloudwatch
        log_stream_prefix from-fluent-bit-
        auto_create_group true
```

Les journaux se trouvent dans CloudWatch, dans la même région AWS que le cluster. Le nom du groupe de journaux est `my-cluster-fluent-bit-logs` et le nom du flux de journaux Fluent Bit est `fluent-bit-podname-pod-namespace` .

> **Note**
>
> - Les journaux de processus sont envoyés seulement lorsque le processus Fluent Bit démarre avec succès. S'il y a un échec lors du démarrage de Fluent Bit, les journaux de processus sont manqués. Vous pouvez uniquement envoyer les journaux de processus à CloudWatch.
> - Pour déboguer l'envoi des journaux de processus à votre compte, vous pouvez appliquer le `ConfigMap` précédent pour obtenir les journaux de processus. L'échec du démarrage de Fluent Bit est généralement dû au fait que votre `ConfigMap` n'est pas analysé ou accepté par Fluent Bit lors du démarrage.

Pour arrêter l'envoi des journaux de processus Fluent Bit

L'envoi des journaux de processus Fluent Bit à CloudWatch implique des coûts supplémentaires d'ingestion de journaux et de stockage. Pour exclure les journaux de processus d'une configuration de `ConfigMap` existante, procédez comme suit.

1. Localisez le groupe de journaux CloudWatch créé automatiquement pour les journaux de processus Fluent Bit de votre cluster Amazon EKS après avoir activé la journalisation Fargate. Il utilise le format `my-cluster-fluent-bit-logs`.
2. Supprimez les flux de journaux CloudWatch existants créés pour les journaux de processus de chaque pod dans le groupe de journaux CloudWatch.
3. Modifiez la `ConfigMap` et définissez `flb_log_cw: "false"`.
4. Redémarrez tous les pods existants du cluster.

Tester l'application

1. Déployez un exemple de pod.
   a. Enregistrez le contenu suivant dans un fichier nommé `sample-app.yaml` sur votre ordinateur.

```
apiVersion: apps/v1
```

```
kind: Deployment
metadata:
  name: sample-app
  namespace: same-namespace-as-your-fargate-profile
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:latest
          ports:
            - name: http
              containerPort: 80
```

b. Appliquez le fichier manifeste à votre cluster.

```
kubectl apply -f sample-app.yaml
```

2. Affichez les journaux NGINX en utilisant les destinations que vous avez configurées dans le fichier `ConfigMap`.

Considérations sur les tailles

Nous vous suggérons de prévoir jusqu'à 50 Mo de mémoire pour le routeur de journaux. Si votre application doit générer des journaux à un débit très élevé, vous devez prévoir jusqu'à 100 Mo.

Dépannage

Pour vérifier si la fonctionnalité de journalisation est activée ou désactivée pour une raison quelconque, telle qu'une valeur non valide `ConfigMap`, et pourquoi elle n'est pas valide, vérifiez les événements de votre Pod avec `kubectl describe pod` *`pod-name`* . La sortie peut inclure des événements de pods qui précisent si la journalisation est activée ou non, comme l'exemple de sortie suivant.

```
[...]
```

```
Annotations:          CapacityProvisioned: 0.25vCPU 0.5GB
                      Logging: LoggingDisabled: LOGGING_CONFIGMAP_NOT_FOUND
[...]
Events:
  Type     Reason           Age        From
              Message
  ----     ------           ----       ----
              -------
  Warning  LoggingDisabled  <unknown>  fargate-scheduler
               Disabled logging because aws-logging configmap was not found. configmap
 "aws-logging" not found
```

Les événements du pod sont éphémères avec une période de temps dépendant des paramètres. Vous pouvez également afficher les annotations d'un pod à l'aide de `kubectl describe pod` *`pod-name`* . Dans l'annotation du pod, il existe des informations sur l'activation ou la désactivation de la fonction de journalisation et la cause.

Choix du type d'instance Amazon EC2 optimal pour les nœuds

Amazon EC2 fournit un large choix de types d'instance pour les composants master. Chaque type d'instance offre des capacités de calcul, de mémoire, de stockage et de réseau différentes. Chaque instance est également regroupée dans une famille d'instances en fonction de ces capacités. Pour la liste complète, consultez Types d'instances disponibles dans le Guide de l'utilisateur Amazon EC2. Amazon EKS publie plusieurs variantes d'Amazon EC2 AMIs pour permettre le support. Pour vous assurer que le type d'instance que vous sélectionnez est compatible avec Amazon EKS, tenez compte des critères suivants.

- Tous les Amazon EKS AMIs ne prennent actuellement pas en charge `mac` cette famille.
- Arm et Amazon EKS non accéléré AMIs ne prennent pas en charge les`g3`, `g4inf`, et `p` les familles.
- Amazon EKS accéléré AMIs ne prend pas en charge les `ac`,`hpc`,`m`, et `t` les familles.
- Pour les instances basées sur ARM, Amazon Linux 2023 (AL2023) ne prend en charge que les types d'instances qui utilisent des processeurs Graviton2 ou version ultérieure. AL2023 ne prend pas en charge `A1` les instances.

Lorsque vous choisissez entre les types d'instances pris en charge par Amazon EKS, tenez compte des fonctionnalités suivantes de chaque type.

Nombre d'instances dans un groupe de nœuds

De manière générale, il est préférable d'utiliser moins d'instances, mais plus grandes, surtout si vous avez de nombreux DaemonSets. Chaque instance nécessitant des appels API vers le serveur d'API, plus le nombre d'instances est élevé, plus la charge sur le serveur d'API est importante.

Système d'exploitation

Examinez les types d'instances pris en charge pour Linux, Windows et Bottlerocket. Avant de créer des instances Windows, consultez Déploiement de nœuds Windows sur les clusters EKS.

Architecture matérielle

Avez-vous besoin d'une architecture x86 ou Arm ? Avant de déployer des instances Arm, consultez Arm Amazon Linux optimisé pour Amazon EKS AMIs. Avez-vous besoin d'instances basées sur le système Nitro (Linux ou Windows) ou disposant de capacités accélérées ? Si vous avez besoin de fonctionnalités accélérées, vous ne pouvez utiliser Linux qu'avec Amazon EKS.

Nombre maximal de pods

Comme chaque pod reçoit sa propre adresse IP, le nombre d'adresses IP prises en charge par un type d'instance est un facteur déterminant pour savoir combien de pods peuvent s'exécuter sur cette instance. Pour déterminer manuellement combien de pods un type d'instance peut prendre en charge, consultez the section called "Nombre maximal de pods recommandé par Amazon EKS pour chaque type d'instance Amazon EC2".

> **Note**
>
> Si vous utilisez une AMI Amazon EKS optimisée Amazon Linux 2 en version `v20220406` ou plus récente, vous pouvez utiliser un nouveau type d'instance sans devoir mettre à jour l'AMI vers la dernière version. Pour ceux-ci AMIs, l'AMI calcule automatiquement la `max-pods` valeur nécessaire si elle n'est pas répertoriée dans le eni-max-podsfichier .txt. Les types d'instance actuellement en prévisualisation peuvent ne pas être pris en charge par Amazon EKS par défaut. Les valeurs pour `max-pods` pour de tels types doivent encore être ajoutées à `eni-max-pods.txt` dans notre AMI.

AWS Les types d'instances Nitro System prennent éventuellement en charge un plus grand nombre d'adresses IP que les types d'instances autres que Nitro System. Cependant, toutes les

adresses IP attribuées à une instance ne sont pas disponibles pour les pods. Pour attribuer un nombre significativement plus élevé d'adresses IP à vos instances, la version `1.9.0` ou ultérieure du module complémentaire Amazon VPC CNI doit être installée dans votre cluster et configurée de manière appropriée. Pour de plus amples informations, veuillez consulter the section called "Augmentation du nombre d'adresses IP". Pour attribuer le plus grand nombre d'adresses IP à vos instances, vous devez avoir installé la version `1.10.1` ou une version ultérieure du module complémentaire Amazon VPC CNI dans votre cluster et déployer le cluster avec la famille `IPv6`.

Famille IP

Vous pouvez utiliser n'importe quel type d'instance pris en charge lorsque vous utilisez la famille `IPv4` pour un cluster, ce qui permet à votre cluster d'attribuer des adresses `IPv4` privées à vos pods et services. Mais si vous souhaitez utiliser la famille `IPv6` pour votre cluster, alors vous devez utiliser les types d'instance AWS Nitro System ou les types d'instance matériel nu. Seul `IPv4` est pris en charge pour les instances Windows. Votre cluster doit utiliser la version `1.10.1` ou une version ultérieure du module complémentaire Amazon VPC CNI. Pour plus d'informations sur l'utilisation de `IPv6`, consultez the section called "IPv6".

Version du module complémentaire Amazon VPC CNI que vous exécutez

La dernière version du plug-in CNI Amazon VPC pour Kubernetes prend en charge ces types d'instance. Il peut être nécessaire de mettre à jour la version de votre module complémentaire CNI Amazon VPC pour bénéficier des derniers types d'instance pris en charge. Pour de plus amples informations, veuillez consulter the section called "CNI Amazon VPC". La dernière version prend en charge les dernières fonctions pour une utilisation avec Amazon EKS. Les versions antérieures ne prennent pas en charge toutes les fonctionnalités. Vous pouvez afficher les fonctions prises en charge par les différentes versions dans Changelog sur GitHub.

Région AWS dans laquelle vous créez vos nœuds

Tous les types d'instances ne sont pas disponibles dans toutes les AWS régions.

Si vous utilisez des groupes de sécurité pour les pods

Si vous utilisez des groupes de sécurité pour les pods, seuls certains types d'instances sont pris en charge. Pour de plus amples informations, veuillez consulter the section called "Groupes de sécurité pour les pods".

Nombre maximal de pods recommandé par Amazon EKS pour chaque type d'instance Amazon EC2

Comme chaque pod reçoit sa propre adresse IP, le nombre d'adresses IP prises en charge par un type d'instance est un facteur déterminant pour savoir combien de pods peuvent s'exécuter sur cette instance. Amazon EKS fournit un script que vous pouvez télécharger et exécuter pour déterminer le nombre maximal de pods recommandé par Amazon EKS pour chaque type d'instance. Le script utilise les attributs matériels de chaque instance et les options de configuration pour déterminer le nombre maximum de pods. Vous pouvez utiliser le nombre renvoyé par ces étapes pour activer des fonctionnalités telles que l'attribution d'adresses IP aux pods depuis un sous-réseau différent de celui de l'instance et l'augmentation significative du nombre d'adresses IP disponibles pour votre instance. Si vous utilisez un groupe de nœuds géré avec plusieurs types d'instances, utilisez une valeur qui fonctionne pour tous les types d'instances.

1. Téléchargez un script que vous pouvez utiliser pour calculer le nombre maximum de pods pour chaque type d'instance.

   ```
   curl -O https://raw.githubusercontent.com/awslabs/amazon-eks-ami/master/templates/al2/runtime/max-pods-calculator.sh
   ```

2. Marquez le script comme exécutable sur votre ordinateur.

   ```
   chmod +x max-pods-calculator.sh
   ```

3. Exécutez le script en remplaçant *`m5.large`* par le type d'instance que vous prévoyez de déployer et *`1.9.0-eksbuild.1`* par votre version du module complémentaire CNI Amazon VPC. Pour déterminer la version de votre module complémentaire, consultez les procédures de mise à jour dans Assign IPs to Pods with the Amazon VPC CNI.

   ```
   ./max-pods-calculator.sh --instance-type m5.large --cni-version 1.9.0-eksbuild.1
   ```

   L'exemple qui suit illustre un résultat.

   ```
   29
   ```

   Vous pouvez ajouter les options suivantes au script pour connaître le nombre maximum de pods pris en charge lors de l'utilisation des fonctionnalités facultatives.

- `--cni-custom-networking-enabled` : utilisez cette option lorsque vous souhaitez attribuer des adresses IP issues d'un sous-réseau différent de celui de votre instance. Pour de plus amples informations, veuillez consulter the section called "Mise en réseau personnalisée". L'ajout de cette option au script précédent avec les mêmes valeurs d'exemple donne `20`.
- `--cni-prefix-delegation-enabled` : utilisez cette option lorsque vous souhaitez attribuer beaucoup plus d'adresses IP à chaque interface réseau Elastic. Cette fonctionnalité nécessite une instance Amazon Linux qui fonctionne sur le système Nitro et la version `1.9.0` ou ultérieure du module complémentaire CNI Amazon VPC. Pour de plus amples informations, veuillez consulter the section called "Augmentation du nombre d'adresses IP". L'ajout de cette option au script précédent avec les mêmes valeurs d'exemple donne `110`.

Vous pouvez également exécuter le script avec l'option `--help` permettant de voir toutes les options disponibles.

> **Note**
>
> Le script de calcul du nombre maximal de pods limite la valeur renvoyée à `110` conformément aux seuils de capacité de mise à l'échelle Kubernetes et aux paramètres recommandés. Si votre type d'instance est supérieur à 30 VCPUs, cette limite passe à`250`, un chiffre basé sur les tests internes de l'équipe d'évolutivité Amazon EKS. Pour plus d'informations, consultez l'article de blog Plugin CNI Amazon VPC augmente les limites du nombre de pods par nœud (français non garanti).

Comment est déterminé MaxPods

La `maxPods` valeur finale appliquée à un nœud dépend de plusieurs composants qui interagissent dans un ordre de priorité spécifique. La compréhension de cet ordre vous permet d'éviter les comportements inattendus lors de la personnalisation`maxPods`.

Ordre de priorité (du plus élevé au plus bas) :

1. Application des groupes de nœuds gérés : lorsque vous utilisez un groupe de nœuds gérés sans AMI personnalisée, Amazon EKS impose un plafonnement `maxPods` des données utilisateur du nœud. Pour les instances avec moins de 30 VCPUs, le plafond est`110`. Pour les instances avec une tension supérieure à 30 VCPUs, le plafond est`250`. Cette valeur a priorité sur toute autre `maxPods` configuration, y compris`maxPodsExpression`.

2. **`maxPods`**configuration kubelet — Si vous définissez `maxPods` directement dans la configuration kubelet (par exemple, via un modèle de lancement avec une AMI personnalisée), cette valeur a priorité sur. `maxPodsExpression`
3. nodeadm **`maxPodsExpression`** — Si vous utilisez `maxPodsExpression`in yourNodeConfig, nodeadm évalue l'expression à calculer. `maxPods` Cela n'est efficace que lorsque la valeur n'est pas déjà définie par une source de priorité supérieure.
4. Calcul par défaut basé sur l'ENI : si aucune autre valeur n'est définie, l'AMI calcule en `maxPods` fonction du nombre d'interfaces réseau élastiques et d'adresses IP prises en charge par le type d'instance. Cela équivaut à la formule`(number of ENIs × (IPs per ENI − 1)) + 2`. Les `+ 2` comptes Amazon VPC CNI s'`kube-proxy`exécutent sur chaque nœud, qui ne consomment pas d'adresse IP de pod.

> **Important**
>
> Si vous utilisez un groupe de nœuds gérés et que vous le définissez `maxPodsExpression` dans votre`NodeConfig`, l'application du groupe de nœuds gérés remplace votre expression. Pour utiliser une `maxPods` valeur personnalisée avec des groupes de nœuds gérés, vous devez spécifier une AMI personnalisée dans votre modèle de lancement et la définir `maxPods` directement. Pour de plus amples informations, veuillez consulter the section called “Modèles de lancement”.

Groupes de nœuds gérés et nœuds autogérés

Avec les groupes de nœuds gérés (sans AMI personnalisée), Amazon EKS injecte la `maxPods` valeur dans les données utilisateur bootstrap du nœud. Autrement dit :

- La `maxPods` valeur est toujours limitée à `110` ou `250` en fonction de la taille de l'instance.
- Toute configuration `maxPodsExpression` que vous configurez est remplacée par cette valeur injectée.
- Pour utiliser une `maxPods` valeur différente, spécifiez une AMI personnalisée dans votre modèle de lancement et `--use-max-pods false` transmettez-la `--kubelet-extra-args '--max-pods=my-value'` au `bootstrap.sh` script. Pour obtenir des exemples, consultez the section called “Modèles de lancement”.

Avec les nœuds autogérés, vous avez un contrôle total sur le processus de démarrage. Vous pouvez utiliser `maxPodsExpression` dans votre `NodeConfig` ou passer `--max-pods` directement à`bootstrap.sh`.

Considérations relatives au mode automatique EKS

Le mode automatique EKS limite le nombre de pods sur les nœuds à la plus faible des deux valeurs suivantes :

- Plafond strict de 110 pods
- Le résultat du calcul du nombre maximal de pods décrit ci-dessus.

Créez des nœuds avec des images optimisées prédéfinies

Vous pouvez déployer des nœuds avec des images Amazon Machine Images (AMI) optimisées pour Amazon EKS pré-construites ou vos propres AMI personnalisées lorsque vous utilisez des groupes de nœuds gérés ou des nœuds autogérés. Si vous utilisez des nœuds hybrides, consultez the section called “Préparer le système d'exploitation”. Pour plus d'informations sur chaque type d'AMI optimisées pour Amazon EKS, veuillez consulter l'une des rubriques suivantes. Pour obtenir des instructions sur la façon de créer votre propre AMI personnalisée, veuillez consulter the section called “Créez une AMI Amazon Linux personnalisée optimisée pour EKS”.

Avec le mode automatique Amazon EKS, EKS gère l'instance EC2, y compris la sélection et la mise à jour de l'AMI.

Rubriques

- Guide des fonctionnalités d' AL2 EKS et de AMIs transition AL2 accélérée
- Créez des nœuds avec Amazon Linux optimisé AMIs
- Créez des nœuds avec Bottlerocket optimisé AMIs
- Créer des nœuds avec des AMI Ubuntu Linux optimisées
- Créez des nœuds avec Windows optimisé AMIs

Guide des fonctionnalités d' AL2 EKS et de AMIs transition AL2 accélérée

> **Warning**
>
> Amazon EKS a cessé de publier Amazon Linux 2 (AL2) optimisé pour EKS AMIs le 26 novembre 2025. AL2Les versions 023 et Bottlerocket pour AMIs Amazon EKS sont disponibles pour toutes les versions de Kubernetes prises en charge, y compris les versions 1.33 et supérieures.

AWS mettra fin à la prise en charge d'EKS, AL2 AL2 optimisée et accélérée AMIs, à compter du 26 novembre 2025. Bien que vous puissiez continuer à utiliser EKS AL2 AMIs après la date end-of-support (EOS) (26 novembre 2025), EKS ne publiera plus de nouvelles versions ou mises à jour de Kubernetes AL2 AMIs, y compris les versions mineures, les correctifs et les corrections de bogues après cette date. Nous vous recommandons de passer à Amazon Linux 2023 (AL2023) ou à AMIs Bottlerocket :

- AL2Le 023 permet une secure-by-default approche avec des politiques de sécurité préconfigurées, SELinux en mode permissif, le mode IMDSv2 uniquement activé par défaut, des temps de démarrage optimisés et une gestion des packages améliorée pour une sécurité et des performances améliorées, parfaitement adapté aux infrastructures nécessitant des personnalisations importantes, telles qu'un accès direct au niveau du système d'exploitation ou des modifications importantes des nœuds. Pour en savoir plus, consultez le numéro AL2023 FAQs ou consultez nos conseils de migration détaillés à l'the section called “Passez à AL2023”adresse.
- Bottlerocket offre une sécurité renforcée, des temps de démarrage plus rapides et une surface d'attaque réduite pour une efficacité accrue grâce à sa conception spécialement optimisée pour les conteneurs, parfaitement adaptée aux approches natives des conteneurs avec un minimum de personnalisations des nœuds. Pour en savoir plus, consultez Bottlerocket FAQs ou consultez nos conseils de migration détaillés à l'adresse. the section called “Flacon Rocket”

Vous pouvez également le faire the section called “Créez une AMI Amazon Linux personnalisée optimisée pour EKS” jusqu'à la date EOS (26 novembre 2025). En outre, vous pouvez créer une AMI personnalisée avec une instance de base Amazon Linux 2 jusqu'à la date Amazon Linux 2 EOS (30 juin 2026).

Migration et support FAQs

Comment migrer de mon AMI AL2 vers une AMI AL2 023 ?

Nous vous recommandons de créer et de mettre en œuvre un plan de migration comprenant des tests approfondis de la charge de travail des applications et des procédures de restauration documentées, puis de suivre les step-by-step instructions de la documentation officielle relative à la mise à niveau d'Amazon Linux 2 vers Amazon Linux 2023 dans la documentation officielle d'EKS.

Puis-je créer une AL2 AMI personnalisée après la date EKS end-of-support (EOS) pour EKS Optimized AL2 AMIs ?

Bien que nous vous recommandions de passer à EKS, officiellement pris en charge et publié, optimisé AMIs pour AL2 023 ou Bottlerocket, vous pouvez créer des EKS personnalisés AL2 optimisés et accélérés AL2 jusqu'à AMIs la date AL2 AMI EOS (26 novembre 2025). Vous pouvez également créer une AMI personnalisée avec une instance de base Amazon Linux 2 jusqu'à la date de fin de vie d'Amazon Linux 2 (30 juin 2026). Pour step-by-step obtenir des instructions sur la création d'une AL2 AMI personnalisée AL2 optimisée et accélérée pour EKS, consultez la section Création d'une AMI Amazon Linux personnalisée dans la documentation officielle d'EKS.

La politique de prise en charge des versions d'EKS Kubernetes s'applique-t-elle aux distributions Amazon Linux ?

Non La date EOS pour EKS AL2 AL2 optimisée et accélérée AMIs est indépendante des délais de support standard et étendus pour les versions Kubernetes d'EKS. Vous devez migrer vers AL2 023 ou Bottlerocket même si vous utilisez le support étendu d'EKS.

Comment le passage de cgroupv1 à cgroupv2 affecte-t-il ma migration ?

La communauté Kubernetes a fait passer le `cgroupv1` support (utilisé par AL2) en mode maintenance, ce qui signifie qu'aucune nouvelle fonctionnalité ne sera ajoutée et que seules des mesures de sécurité critiques et des corrections de bogues majeures seront fournies. Pour adopter `cgroupv2` dans Kubernetes, vous devez vous assurer de la compatibilité entre le système d'exploitation, le noyau, l'exécution de conteneur et les composants Kubernetes. Cela nécessite une distribution Linux activée `cgroupv2` par défaut, telle que AL2 023, Bottlerocket, Red Hat Enterprise Linux (RHEL) 9+, Ubuntu 22.04+ ou Debian 11+. Ces distributions sont fournies avec des versions du noyau ≥ 5.8, qui constituent la configuration minimale requise pour la prise en charge `cgroupv2` dans Kubernetes. Pour en savoir plus, consultez À propos de cgroup v2.

Que dois-je faire si j'ai besoin de Neuron dans mon AL2 AMI personnalisée ?

Vous ne pouvez pas exécuter toutes vos applications alimentées par Neuron de manière native sur une base. AL2 AMIs Pour tirer parti de AWS Neuron sur une AL2 AMI, vous devez conteneuriser vos applications à l'aide d'un conteneur compatible avec Neuron avec une distribution autre que AL2 Linux (par exemple, Ubuntu 22.04, Amazon Linux 2023, etc.), puis déployer ces conteneurs sur une AL2 AMI basée sur laquelle le pilote Neuron () est installé. `aws-neuronx-dkms`

Dois-je passer à une instance de base Amazon Linux 2 vierge après la date EOS de l' AL2 AMI EKS (26 novembre 2025) ?

Le passage à une instance de base Amazon Linux 2 pure ne présente pas les optimisations spécifiques, les configurations d'exécution des conteneurs et les personnalisations fournies par l'EKS officiel, AL2 optimisé et accéléré. AL2 AMIs Si vous devez plutôt continuer à utiliser une solution AL2 basée, nous vous recommandons de créer une AMI personnalisée à l'aide des recettes d'AMI EKS disponibles à l'adresse the section called “Créez une AMI Amazon Linux personnalisée optimisée pour EKS” ou de la spécification de construction de l'AMI Amazon EKS. Cela garantit la compatibilité avec vos charges de travail existantes et inclut les mises à jour AL2 du noyau jusqu'à la date EOS d'Amazon Linux 2 (30 juin 2026).

Lorsque vous créez une AL2 AMI personnalisée à l'aide du GitHub référentiel EKS AMI après la date EOS de l' AL2 AMI EKS (26 novembre 2025), quel support est disponible pour les packages provenant de référentiels tels que amzn2-core et amzn2extra-docker ?

La recette EKS AMI sur Amazon EKS AMI Build Specification récupère les paquets via YUM à partir du logiciel Amazon Linux 2 standard, tel que amzn2-core et amzn2extra-docker. Après la date EKS AL2 AMI EOS (26 novembre 2025), ce logiciel continuera d'être pris en charge jusqu'à la date plus large d'Amazon Linux 2 EOS (30 juin 2026). Veuillez noter que l'assistance est limitée aux mises à jour du noyau pendant cette période, ce qui signifie que vous devrez gérer et appliquer manuellement les autres mises à jour de paquets, les correctifs de sécurité et toutes les dépendances non liées au noyau afin de maintenir la sécurité et la compatibilité.

Pourquoi les applications Java utilisant d'anciennes versions d' JDK8 Amazon EKS avec AL2 023 peuvent-elles rencontrer des exceptions de manque de mémoire (OOM) et des redémarrages de pods, et comment résoudre ce problème ?

Lorsqu'elles sont exécutées sur des nœuds Amazon EKS avec AL2 023, les applications Java utilisant des versions antérieures du JDK 8 `jdk8u372` peuvent provoquer des exceptions OOM et des redémarrages de pods car la machine virtuelle Java n'est pas compatible avec. `cgroupv2` Ce

problème provient spécifiquement de l'incapacité de la JVM à détecter les limites de mémoire du conteneur à l'aide de `cgroupv2`, la valeur par défaut dans Amazon Linux 2023. Par conséquent, il base l'allocation de tas sur la mémoire totale du nœud plutôt que sur la limite définie par le pod. Cela provient du fait que `cgroupv2` change l'emplacement de stockage des données relatives à la limite de mémoire, ce qui entraîne une mauvaise lecture de la mémoire disponible par les anciennes versions de Java et une estimation erronée des ressources au niveau des nœuds. Les options suivantes sont les suivantes :

- Mise à niveau de la version JDK : La mise à niveau vers `jdk8u372` ou une version ultérieure, ou vers une version JDK plus récente avec prise en charge `cgroupv2` complète, peut résoudre ce problème. Pour obtenir la liste des versions Java compatibles qui prennent entièrement en charge `cgroupv2`, consultez la section À propos de cgroup v2.
- Création d'une AMI personnalisée : si vous devez continuer à utiliser une solution AL2 basée, vous pouvez créer une AMI personnalisée AL2 (jusqu'au 26 novembre 2025) à l'aide the section called "Créez une AMI Amazon Linux personnalisée optimisée pour EKS" de la spécification de construction de l'AMI Amazon EKS. Par exemple, vous pouvez créer une AMI AL2 basée sur la version v1.33 (jusqu'au 26 novembre 2025). Amazon EKS fournira des informations AL2 basées sur AMIs la date AL2 EKS EOS (26 novembre 2025). Après la date EOS (26 novembre 2025), vous devrez créer votre propre AMI.
- Activer cgroupv1 : si vous devez continuer à utiliser`cgroupv1`, vous pouvez l'activer `cgroupv1` sur une AMI EKS AL2 023. Pour activer, exécuter `sudo grubby --update-kernel=ALL --args="systemd.unified_cgroup_hierarchy=0"` et redémarrer le système (par exemple, une EC2 instance ou un nœud exécutant Amazon Linux 2023). Cela modifiera les paramètres de démarrage du système (par exemple, en ajoutant le paramètre du noyau « systemd.unified_cgroup_hierarchy=0 » à la configuration GRUB, qui indique à systemd d'utiliser l'ancienne hiérarchie `cgroupv1`) et activera `cgroupv1`. Notez que lorsque vous exécutez cette commande grubby, vous reconfigurez le noyau avec `cgroupv1` activé et `cgroupv2` désactivé. Une seule de ces versions de cgroup est utilisée pour la gestion active des ressources sur un nœud. Ce n'est pas la même chose que d'assurer la rétrocompatibilité de `cgroupv2` pour l'API `cgroupv1`.

> **Warning**
>
> Nous ne recommandons pas l'utilisation continue de `cgroupv1`. Nous vous recommandons plutôt de migrer vers `cgroupv2`. La communauté Kubernetes a fait passer le `cgroupv1` support (utilisé par AL2) en mode maintenance, ce qui signifie qu'aucune nouvelle

fonctionnalité ou mise à jour ne sera ajoutée et que seuls des correctifs de sécurité essentiels et des corrections de bogues majeurs seront fournis. La suppression complète de la prise en charge `cgroupv1` est prévue dans une prochaine version, mais aucune date précise n'a encore été annoncée. Si vous rencontrez des problèmes avec`cgroupv1`, ne AWS sera pas en mesure de fournir une assistance et vous recommandera de passer à`cgroupv2`.

Compatibilité et versions

Versions de Kubernetes prises en charge pour AL2 AMIs

La version 1.32 de Kubernetes est la dernière version pour laquelle Amazon EKS publiera (AL2 Amazon Linux 2). AMIs Pour les versions de Kubernetes prises en charge jusqu'à la version 1.32, EKS continuera à publier AL2 AMIs (AL2_ARM_64, _x86_64) et à accélérer (AL2_x86_64_GPU) jusqu'au 26 novembre 2025. AL2 AMIs AL2 Après cette date, EKS cessera de publier des versions AL2 optimisées et AL2 accélérées AMIs pour toutes les versions de Kubernetes. Notez que la date EOS pour EKS AL2 AL2 optimisée et accélérée AMIs est indépendante des délais de support standard et étendu pour les versions Kubernetes d'EKS.

Comparaison des pilotes pris en charge et des versions du noyau Linux pour AL2, AL2 023 et Bottlerocket AMIs

Composant	AL2 AMI D'EKS	EKS AL2 023 AMI	EKS Bottlerocket AMI
Compatibilité du système d'exploitation de base	RHEL7/CentOS 7	Fedora/CentOS 9	N/A
pilote en mode utilisateur CUDA	12.x	12,x, 13,x	12,x, 13,x
Pilote GPU NVIDIA	R570	R580	R570, R580
AWS Pilote Neuron	2,20 ans et plus	2,20 ans et plus	2,20 ans et plus
Noyau Linux	5,10	6.1, 6.12	6.1, 6.12

Pour plus d'informations sur le pilote NVIDIA et la compatibilité CUDA, consultez la documentation NVIDIA.

AWS Compatibilité des neurones avec AL2 AMIs

À partir de AWS la version 2.20 de Neuron, le Neuron Runtime (`aws-neuronx-runtime-lib`) utilisé par EKS AL AMIs n'est plus compatible avec Amazon Linux 2 (). AL2 Le pilote Neuron (`aws-neuronx-dkms`) est désormais le seul package AWS Neuron compatible avec Amazon Linux 2. Cela signifie que vous ne pouvez pas exécuter vos applications alimentées par Neuron de manière native sur une AMI basée AL2 sur une AMI. Pour configurer Neuron on AL2 023 AMIs, consultez le guide de configuration AWS Neuron.

Compatibilité de Kubernetes avec AL2 AMIs

La communauté Kubernetes a déplacé le `cgroupv1` support (utilisé par AL2) en mode maintenance. Cela signifie qu'aucune nouvelle fonctionnalité ne sera ajoutée et que seules les corrections de sécurité critiques et les corrections de bogues majeurs seront fournies. Toutes les fonctionnalités de Kubernetes basées sur cgroupv2, telles que MemoryQo S et l'isolation améliorée des ressources, ne sont pas disponibles sur. AL2 En outre, la version 1.32 d'Amazon EKS Kubernetes était la dernière version à être prise en charge. AL2 AMIs Pour maintenir la compatibilité avec les dernières versions de Kubernetes, nous vous recommandons de migrer vers AL2 023 ou Bottlerocket, qui sont activées par défaut. `cgroupv2`

Compatibilité des versions Linux avec AL2 AMIs

Amazon Linux 2 (AL2) est pris en charge AWS jusqu'à sa date end-of-support (EOS) du 30 juin 2026. Cependant, avec le AL2 temps, le soutien apporté par l'ensemble de la communauté Linux aux nouvelles applications et fonctionnalités est devenu plus limité. AL2 AMIs sont basés sur le noyau Linux 5.10, tandis que AL2 023 utilise le noyau Linux 6.1. Contrairement à AL2 023, AL2 bénéficie d'un soutien limité de la part de la communauté Linux au sens large. Cela signifie que de nombreux packages et outils Linux en amont doivent être rétroportés pour fonctionner avec l'ancienne version AL2 du noyau, certaines fonctionnalités modernes de Linux et certaines améliorations de sécurité ne sont pas disponibles en raison de l'ancien noyau, de nombreux projets open source ont déconseillé ou limité le support des anciennes versions du noyau, comme la 5.10.

Packages obsolètes non inclus dans la version 023 AL2

Voici quelques-uns des packages les plus courants qui ne sont pas inclus ou qui ont changé en AL2 2023 :

- Certains paquets binaires source d'Amazon Linux 2 ne sont plus disponibles dans Amazon Linux 2023.
- Modifications apportées à la manière dont Amazon Linux prend en charge les différentes versions de packages (par exemple, amazon-linux-extras le système) en AL2 2023
- Les packages supplémentaires pour Enterprise Linux (EPEL) ne sont pas pris en charge dans AL2 la version 023
- Les applications 32 bits ne sont pas prises en charge dans AL2 la version 2.3

Pour en savoir plus, consultez Comparing AL2 et AL2 023.

Comparaison des validations FIPS entre AL2, AL2 023 et Bottlerocket

Amazon Linux 2 (AL2), Amazon Linux 2023 (AL2023) et Bottlerocket fournissent une assistance pour la conformité aux normes fédérales de traitement de l'information (FIPS).

- AL2 est certifié selon FIPS 140-2 et AL2 023 est certifié selon FIPS 140-3. Pour activer le mode FIPS le AL2 023, installez les packages nécessaires sur votre EC2 instance Amazon et suivez les étapes de configuration en suivant les instructions de la section Activer le mode FIPS le 023. AL2 Pour en savoir plus, consultez AL2023 FAQs.
- Bottlerocket fournit des variantes spécialement conçues pour FIPS qui limitent les composants du noyau et de l'espace utilisateur à l'utilisation de modules cryptographiques qui ont été soumis au programme de validation des modules cryptographiques FIPS 140-3.

Journal des modifications du pilote EKS AMI et des versions

Pour obtenir la liste complète de tous les composants de l'AMI EKS et de leurs versions, consultez les notes de mise à jour de l'AMI Amazon EKS sur GitHub.

Créez des nœuds avec Amazon Linux optimisé AMIs

Amazon Elastic Kubernetes Service (Amazon EKS) fournit des AMIs images Amazon Machine () spécialisées optimisées pour exécuter des nœuds de travail Kubernetes. Ces Amazon Linux (AL) optimisés pour EKS AMIs sont préconfigurés avec des composants essentiels, tels que l'AWS `kubelet` IAM Authenticator et, pour garantir une intégration et une sécurité sans faille au sein de vos clusters. `containerd` Ce guide détaille les versions d'AMI disponibles et décrit les options spécialisées pour le calcul accéléré et les architectures basées sur ARM.

Considérations

- Vous pouvez suivre les événements liés à la sécurité ou à la confidentialité pourAmazon Linux dans le centre de sécurité Amazon Linux en sélectionnant l'onglet correspondant à la version souhaitée. Vous pouvez également vous abonner au flux RSS applicable. Les événements de sécurité et de confidentialité incluent une présentation du problème, les packages concernés et la manière de mettre à jour vos instances pour résoudre le problème.
- Avant de déployer une AMI accélérée ou une AMI Arm, consultez les informations contenues dans Amazon Linux AMIs accéléré optimisé pour Amazon EKS et. the section called "Arm optimisé pour Amazon EKS | Amazon Linux AMIs"
- Les EC2 `P2` instances Amazon ne sont pas prises en charge sur Amazon EKS car elles nécessitent la version 470 ou antérieure du `NVIDIA` pilote.
- Tous les groupes de nœuds gérés nouvellement créés dans des clusters de la version `1.30` ou d'une version plus récente utiliseront automatiquement par défaut AL2 023 comme système d'exploitation du nœud.

Amazon Linux accéléré optimisé pour Amazon EKS AMIs

Amazon Linux (AL) AMIs accéléré optimisé pour Amazon EKS repose sur le système Amazon Linux standard optimisé pour EKS. AMIs Ils sont configurés pour servir d'images optionnelles pour les nœuds Amazon EKS afin de prendre en charge les charges de travail basées sur GPU, Inferentia et Trainium.

Pour de plus amples informations, veuillez consulter the section called "Utiliser le GPU EKS Linux AMIs".

Arm optimisé pour Amazon EKS | Amazon Linux AMIs

Les instances Arm apportent des économies significatives en termes de coût et sont bien adaptées aux applications dimensionnables et basées sur Arm, telles que serveurs web, microservices conteneurisés, flottes de cache et magasins de données distribuées. Lorsque vous ajoutez des nœuds Arm à votre cluster, passez en revue les considérations suivantes.

- Si votre cluster a été déployé avant le 17 août 2020, vous devez effectuer une mise à niveau unique des manifestes des modules complémentaires critiques du cluster. Ceci afin que Kubernetes puisse extraire l'image correcte pour chaque architecture matérielle utilisée dans votre cluster. Pour plus d'informations sur la mise à jour des modules complémentaires de clusters,

consultez the section called "Étape 1 : préparer la mise à niveau". Si vous avez déployé votre cluster à partir du 17 août 2020, vos modules complémentaires CoreDNS, `kube-proxy` et le plug-in CNI Amazon VPC pour Kubernetes sont déjà compatibles avec plusieurs architectures.

- Les applications déployées sur les nœuds Arm doivent être compilées pour Arm.
- Si vous en avez DaemonSets déployé dans un cluster existant, ou si vous souhaitez les déployer sur un nouveau cluster dans lequel vous souhaitez également déployer des nœuds Arm, vérifiez que vous DaemonSet pouvez les exécuter sur toutes les architectures matérielles de votre cluster.
- Vous pouvez exécuter des groupes de nœuds Arm et des groupes de nœuds x86 dans le même cluster. Si vous procédez de la sorte, envisagez de déployer des images de conteneur multi-architecture dans un registre de conteneurs tel qu'Amazon Elastic Container Registry, puis d'ajouter des sélecteurs de nœuds à vos manifestes afin que Kubernetes connaisse l'architecture matérielle dans laquelle un pod peut être déployé. Pour de plus amples informations, consultez Transmission d'une image multi-architecture dans le Guide de l'utilisateur Amazon ECR et l'article de blog Présentation d'images de conteneurs multi-architectures pour Amazon ECR.

En savoir plus

Pour plus d'informations sur l'utilisation d'Amazon Linux optimisé pour Amazon EKS AMIs, consultez les sections suivantes :

- Pour utiliser Amazon Linux avec des groupes de nœuds gérés, veuillez consulter la rubrique the section called "Groupes de nœuds gérés".
- Pour lancer des nœuds Amazon Linux autogérés, veuillez consulter la rubrique the section called "Recevez les dernières nouvelles IDs".
- Pour plus d'informations sur la version, consultez the section called "Obtenir les informations sur la version".
- Pour accéder à la dernière version IDs d'Amazon Linux optimisée pour Amazon EKS AMIs, consultez. the section called "Recevez les dernières nouvelles IDs"
- Pour les scripts open source utilisés pour créer l'Amazon EKS optimisé, consultez AMIs. the section called "Créez une AMI Amazon Linux personnalisée optimisée pour EKS"

Mise à niveau d'Amazon Linux 2 vers Amazon Linux 2023

> **Warning**
>
> Amazon EKS a cessé de publier Amazon Linux 2 (AL2) optimisé pour EKS AMIs le 26 novembre 2025. AL2023 et Bottlerocket basés sur AMIs Amazon EKS sont disponibles pour toutes les versions de Kubernetes prises en charge, y compris les versions 1.33 et supérieures.

AL2023 est un système d'exploitation basé sur Linux conçu pour fournir un environnement sécurisé, stable et performant pour vos applications cloud. Il s'agit de la nouvelle génération d'Amazon Linux proposée par Amazon Web Services et est disponible pour toutes les versions Amazon EKS prises en charge.

AL2023 propose plusieurs améliorations par rapport à AL2. Pour une comparaison complète, consultez Comparing AL2 et Amazon Linux 2023 dans le guide de l'utilisateur Amazon Linux 2023. Plusieurs packages ont été ajoutés, mis à niveau et supprimés AL2. Il est vivement recommandé de tester vos applications AL2023 avant de procéder à la mise à niveau. Pour obtenir la liste de toutes les modifications apportées aux packages AL2023, consultez la section Modifications de package dans Amazon Linux 2023 dans les notes de mise à jour d'Amazon Linux 2023.

En plus de ces modifications, prenez en compte les éléments suivants :

- AL2023 introduit un nouveau processus d'initialisation des nœuds `nodeadm` qui utilise un schéma de configuration YAML. Si vous utilisez des groupes de nœuds autogérés ou une AMI avec un modèle de lancement, vous devez désormais fournir explicitement des métadonnées de cluster supplémentaires lors de la création d'un nouveau groupe de nœuds. Un exemple des paramètres minimum requis est présenté ci-dessous, où `apiServerEndpoint`, `certificateAuthority` et le service `cidr` sont désormais obligatoires :

```
---
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
    name: my-cluster
    apiServerEndpoint: https://example.com
    certificateAuthority: Y2VydGlmaWNhdGVBdXRob3JpdHk=
```

```
    cidr: 10.100.0.0/16
```

Dans AL2, les métadonnées de ces paramètres ont été découvertes à partir de l'appel d'`DescribeCluster`API Amazon EKS. Avec AL2023, ce comportement a changé car l'appel d'API supplémentaire risque d'être limité lors de la mise à l'échelle de nœuds à grande échelle. Cette modification ne vous affecte pas si vous utilisez des groupes de nœuds gérés sans modèle de lancement, ou si vous utilisez Karpenter. Pour plus d'informations sur `certificateAuthority` et le service `cidr`, consultez `DescribeCluster` dans la Référence de l'API Amazon EKS.

- Pour AL2023, modifie `nodeadm` également le format pour appliquer des paramètres à chaque nœud utilisant `NodeConfigSpec`. `kubelet` En AL2, cela a été fait avec le `--kubelet-extra-args` paramètre. Ceci est couramment utilisé pour ajouter des étiquettes et des rejets aux nœuds. L'exemple ci-dessous illustre l'application de `maxPods` et `--node-labels` sur le nœud.

```
---
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
    name: test-cluster
    apiServerEndpoint: https://example.com
    certificateAuthority: Y2VydGlmaWNhdGVBdXRob3JpdHk=
    cidr: 10.100.0.0/16
  kubelet:
    config:
      maxPods: 110
    flags:
      - --node-labels=karpenter.sh/capacity-type=on-demand,karpenter.sh/nodepool=test
```

- La version CNI d'Amazon VPC `1.16.2` ou supérieure est requise pour. AL2023
- AL2023 nécessite `IMDSv2` par défaut. `IMDSv2`présente plusieurs avantages qui contribuent à améliorer la posture de sécurité. Il utilise une méthode d'authentification orientée session qui nécessite la création d'un jeton secret dans une simple requête HTTP PUT pour démarrer la session. La durée de validité du jeton de session peut avoir une valeur quelconque entre 1 seconde et 6 heures. Pour plus d'informations sur la manière de passer de `IMDSv1` à`IMDSv2`, consultez Transition vers l'utilisation du service de métadonnées d'instance version 2 et Profitez pleinement des avantages de votre AWS infrastructure IMDSv2 et désactivez-la IMDSv1 dans l'ensemble de celle-ci. Si vous souhaitez continuer à utiliser `IMDSv1`, vous pouvez le faire en surchargant manuellement les paramètres via les propriétés de lancement de l'instance.

> **Note**
>
> Pour `IMDSv2` with AL2023, le nombre de sauts par défaut pour les groupes de nœuds gérés peut varier :
>
> - Lorsque vous n'utilisez pas de modèle de lancement, la valeur par défaut est définie sur 1. Cela signifie que les conteneurs n'auront pas accès aux informations d'identification du nœud via IMDS. Si vous avez besoin d'un accès par conteneur aux informations d'identification du nœud, vous pouvez toujours le faire en utilisant un modèle de lancement Amazon EC2 personnalisé.
> - Lorsque vous utilisez un AMI personnalisée dans un modèle de lancement, la valeur `HttpPutResponseHopLimit` par défaut est définie sur 2. Vous pouvez modifier manuellement `HttpPutResponseHopLimit` dans le modèle de lancement.
>
> Vous pouvez également utiliser Identité du pod Amazon EKS pour fournir des informations d'identification, plutôt que d'utiliser `IMDSv2`.

- AL2023 propose la nouvelle génération de hiérarchie unifiée des groupes de contrôle (`cgroupv2`). `cgroupv2`est utilisé pour implémenter un environnement d'exécution de conteneur, et par`systemd`. Bien qu'elle contienne AL2023 toujours du code permettant au système de fonctionner`cgroupv1`, cette configuration n'est ni recommandée ni prise en charge. Cette configuration sera complètement supprimée dans une prochaine version majeure d'Amazon Linux.
- `eksctl`une version `0.176.0` ou supérieure est requise `eksctl` pour le support AL2023.

Pour les groupes de nœuds gérés existants, vous pouvez effectuer une mise à niveau sur place ou une mise à blue/green niveau en fonction de la manière dont vous utilisez un modèle de lancement :

- Si vous utilisez une AMI personnalisée avec un groupe de nœuds gérés, vous pouvez effectuer une mise à niveau sur place en remplaçant l'ID de l'AMI dans le modèle de lancement. Vous devez vous assurer que vos applications et toutes les données utilisateur sont transférées vers le AL2023 premier avant de mettre en œuvre cette stratégie de mise à niveau.
- Si vous utilisez des groupes de nœuds gérés avec le modèle de lancement standard ou avec un modèle de lancement personnalisé qui ne précise pas l'ID de l'AMI, vous devez effectuer une mise à niveau à l'aide d'une blue/green stratégie. Une blue/green mise à niveau est généralement plus complexe et implique la création d'un tout nouveau groupe de nœuds dans lequel vous devez spécifier AL2023 le type d'AMI. Le nouveau groupe de nœuds devra ensuite être configuré avec soin pour garantir que toutes les données personnalisées du groupe de AL2 nœuds sont

compatibles avec le nouveau système d'exploitation. Une fois le nouveau groupe de nœuds testé et validé avec vos applications, vous pouvez migrer les pods de l'ancien groupe vers le nouveau. Après la migration, vous pouvez supprimer l'ancien groupe de nœuds.

Si vous utilisez Karpenter et que vous souhaitez l'utiliser AL2023, vous devez modifier le `EC2NodeClass` `amiFamily` champ avec AL2023. Par défaut, la fonction de dérive est activée dans Karpenter. Cela signifie qu'une fois le champ `amiFamily` modifié, Karpenter mettra automatiquement à jour vos composants master avec la dernière AMI lorsqu'elle sera disponible.

Informations supplémentaires sur nodeadm

Lorsque vous utilisez une AMI Amazon Linux 2023 optimisée pour EKS ou que vous créez une AMI EKS Amazon Linux 2023 personnalisée via les scripts Packer fournis dans le amazon-eks-ami GitHub référentiel officiel, vous devez éviter d'exécuter explicitement nodeadm init dans les données utilisateur EC2 ou dans le cadre de votre AMI personnalisée.

Si vous souhaitez générer une dynamique NodeConfig dans vos données utilisateur, vous pouvez écrire cette configuration dans un fichier yaml ou json intégré. `/etc/eks/nodeadm.d` Ces fichiers de configuration seront fusionnés et appliqués à votre nœud lorsque nodeadm init sera automatiquement lancé ultérieurement dans le processus de démarrage. Par exemple :

```
cat > /etc/eks/nodeadm.d/additional-node-labels.yaml << EOF
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  kubelet:
    flags:
      - --node-labels=foo=bar
EOF
```

Amazon Linux 2023, optimisé pour EKS, exécute AMIs automatiquement nodeadm init en deux phases via des services systemd distincts : nodeadm-config s'exécute avant l'exécution des données utilisateur, tandis que nodeadm-run s'exécute ensuite. Le service nodeadm-config établit des configurations de base pour containerd et kubelet avant l'exécution des données utilisateur. Le service nodeadm-run exécute certains démons du système et effectue toutes les configurations finales après l'exécution des données utilisateur. Si la commande nodeadm init est exécutée une fois de plus, via les données utilisateur ou une AMI personnalisée, elle risque de briser les hypothèses concernant l'ordre d'exécution et d'entraîner des résultats inattendus, notamment des erreurs de configuration. ENIs

Récupérer les informations sur la version Amazon Linux AMI

Les AMI optimisées par Amazon EKS Linux sont versionnées par la version Kubernetes et la date de sortie de l'AMI dans le format suivant :

```
k8s_major_version.k8s_minor_version.k8s_patch_version-release_date
```

Chaque version AMI comprend différentes versions de kubelet, du noyau Linux et de containerd. Les AMI accélérées comprennent également différentes versions du pilote NVIDIA. Vous trouverez ces informations sur la version dans le journal des modifications sur GitHub.

Récupérez l'AMI Amazon Linux recommandée IDs

Lors du déploiement de nœuds, vous pouvez spécifier un identifiant pour une image Amazon Machine Image (AMI) optimisée pour Amazon EKS pré-créée. Pour récupérer un ID d'AMI correspondant à la configuration souhaitée, interrogez l'API du magasin de paramètres AWS Systems Manager. L'utilisation de cette API élimine le besoin de rechercher manuellement l'AMI optimisée pour Amazon EKS IDs. Pour de plus amples informations, veuillez consulter GetParameter. Le principal IAM que vous utilisez doit disposer de l'autorisation IAM `ssm:GetParameter` pour récupérer les métadonnées de l'AMI optimisée pour Amazon EKS.

Vous pouvez récupérer l'ID d'image de la dernière AMI Amazon Linux optimisée pour Amazon EKS recommandée à l'aide de la commande suivante, qui utilise le sous-paramètre `image_id`. Si nécessaire, apportez les modifications suivantes à la commande, puis exécutez la commande modifiée :

- Veuillez remplacer `<kubernetes-version>` par une version prise en charge par Amazon EKS.
- *`ami-type`*Remplacez-le par l'une des options suivantes. Pour plus d'informations sur les types d' EC2 instances Amazon, consultez la section Types d' EC2 instances Amazon.
  - *`amazon-linux-2023/x86_64/standard`*À utiliser pour les instances x86 basées sur Amazon Linux 2023 (AL2023).
  - *`amazon-linux-2023/arm64/standard`*À utiliser pour AL2 023 instances ARM, telles que les instances basées sur AWS Graviton.
  - *`amazon-linux-2023/x86_64/nvidia`*À utiliser pour les dernières instances NVIDIA AL2 x86 023 approuvées.
  - *`amazon-linux-2023/arm64/nvidia`*À utiliser pour les dernières instances NVIDIA AL2 `arm64` 023 approuvées.

- *amazon-linux-2023/x86_64/neuron*À utiliser pour les dernières instances AL2 023 AWS Neuron.
- <region-code>Remplacez-le par une AWS région prise en charge par Amazon EKS pour laquelle vous souhaitez obtenir l'ID AMI.

```
aws ssm get-parameter --name /aws/service/eks/optimized-ami/<kubernetes-version>/<ami-type>/recommended/image_id \
    --region <region-code> --query "Parameter.Value" --output text
```

Voici un exemple de commande après remplacement des espaces réservés.

```
aws ssm get-parameter --name /aws/service/eks/optimized-ami/1.31/amazon-linux-2023/x86_64/standard/recommended/image_id \
    --region us-west-2 --query "Parameter.Value" --output text
```

L'exemple qui suit illustre un résultat.

```
ami-1234567890abcdef0
```

Créez une AMI Amazon Linux personnalisée optimisée pour EKS

> **Warning**
>
> Amazon EKS a cessé de publier Amazon Linux 2 (AL2) optimisé pour EKS AMIs le 26 novembre 2025. AL2Les versions 023 et Bottlerocket pour AMIs Amazon EKS sont disponibles pour toutes les versions de Kubernetes prises en charge, y compris les versions 1.33 et supérieures.

Amazon EKS fournit des scripts de génération open source dans le référentiel Amazon EKS AMI Build Specification que vous pouvez utiliser pour consulter les configurations, le moteur d'exécution`kubelet`, l'authentificateur AWS IAM pour Kubernetes et créer votre propre AMI basée sur AL à partir de zéro.

Ce dépôt contient le script bootstrap spécialisé AL2 et l'outil nodeadm pour AL2 023 qui s'exécute au démarrage. Ces scripts configurent les données de certificat de votre instance, le point de

terminaison du plan de contrôle, le nom du cluster, et bien plus encore. Les scripts sont considérés comme la source de vérité pour les versions d'AMI optimisées pour Amazon EKS. Vous pouvez donc suivre le GitHub référentiel pour suivre les modifications apportées à notre. AMIs

Lors de la création personnalisée AMIs avec l'EKS Optimized AMIs comme base, il n'est pas recommandé ou pris en charge d'exécuter une mise à niveau du système d'exploitation (par ex. `dnf upgrade`) ou mettez à niveau l'un des packages Kubernetes ou GPU inclus dans l'EKS Optimized AMIs, car cela risque de compromettre la compatibilité des composants. Si vous mettez à niveau le système d'exploitation ou les packages inclus dans l'EKS Optimized AMIs, il est recommandé de procéder à des tests approfondis dans un environnement de développement ou de préparation avant le déploiement en production.

Lors de la création d'instances personnalisées AMIs pour le GPU, il est recommandé de créer une AMIs configuration personnalisée distincte pour chaque génération et famille d'instances que vous allez exécuter. L'accélérateur optimisé pour EKS installe de AMIs manière sélective les pilotes et les packages au moment de l'exécution en fonction de la génération et de la famille du type d'instance sous-jacent. Pour plus d'informations, consultez les scripts EKS AMI pour l'installation et l'exécution.

Conditions préalables

- Installation de la AWS CLI
- Installez HashiCorp Packer v1.9.4+
- Installez GNU Make

Démarrage rapide

Ce guide de démarrage rapide vous montre les commandes permettant de créer une AMI personnalisée dans votre AWS compte. Pour en savoir plus sur les configurations disponibles pour personnaliser votre AMI, consultez les variables de modèle sur la page Amazon Linux 2023.

Conditions préalables

Installez le plug-in Amazon requis. Par exemple :

```
packer plugins install github.com/hashicorp/amazon
```

Étape 1. Configurez votre environnement

Clonez ou créez une fourche du référentiel AMI officiel d'Amazon EKS. Par exemple :

```
git clone https://github.com/awslabs/amazon-eks-ami.git
cd amazon-eks-ami
```

Vérifiez que Packer est installé :

```
packer --version
```

Étape 2. Pour créer une AMI personnalisée

Vous trouverez ci-dessous des exemples de commandes pour différentes commandes personnalisées AMIs.

AL2 AMI NVIDIA de base :

```
make k8s=1.31 os_distro=al2 \
  enable_accelerator=nvidia \
  nvidia_driver_major_version=560 \
  enable_efa=true
```

AMI NVIDIA AL2 023 de base :

```
make k8s=1.31 os_distro=al2023 \
  enable_accelerator=nvidia \
  nvidia_driver_major_version=560 \
  enable_efa=true
```

AMI Neuron AL2 023 conforme aux STIG :

```
make k8s=1.31 os_distro=al2023 \
  enable_accelerator=neuron \
  enable_fips=true \
  source_ami_id=ami-0abcd1234efgh5678 \
  kms_key_id=alias/aws-stig
```

Après avoir exécuté ces commandes, Packer effectuera les opérations suivantes :* Lancer une EC2 instance Amazon temporaire. * Installez les composants, les pilotes et les configurations de Kubernetes. * Créez l'AMI dans votre AWS compte.

Le résultat attendu devrait ressembler à ceci :

```
==> Wait completed after 8 minutes 42 seconds

==> Builds finished. The artifacts of successful builds are:
--> amazon-ebs: AMIs were created:
us-west-2: ami-0e139a4b1a7a9a3e9

--> amazon-ebs: AMIs were created:
us-west-2: ami-0e139a4b1a7a9a3e9

--> amazon-ebs: AMIs were created:
us-west-2: ami-0e139a4b1a7a9a3e9
```

Étape 3. Afficher les valeurs par défaut

Pour afficher les valeurs par défaut et les options supplémentaires, exécutez la commande suivante :

```
make help
```

Créez des nœuds avec Bottlerocket optimisé AMIs

Bottlerocket est une distribution Linux open source sponsorisée et soutenue par. AWS Bottlerocket est spécialement conçu pour héberger des charges de travail conteneurisées. Avec Bottlerocket, vous pouvez améliorer la disponibilité des déploiements conteneurisés et réduire les coûts opérationnels en automatisant les mises à jour de votre infrastructure de conteneurs. Bottlerocket ne comprend que les logiciels essentiels à l'exécution des conteneurs, ce qui améliore l'utilisation des ressources, réduit les menaces de sécurité et diminue les frais généraux liés à la gestion. L'AMI Bottlerocket `containerd` inclut`kubelet`, AWS et l'authentificateur IAM. En plus des groupes de nœuds gérés et des nœuds autogérés, Bottlerocket est également pris en charge par Karpenter.

Avantages

L'utilisation de Bottlerocket avec votre cluster Amazon EKS présente les bénéfices suivants :

- Disponibilité accrue, coûts d'exploitation réduits et gestion simplifiée : Bottlerocket nécessite moins de ressources, démarre plus rapidement et est moins vulnérable aux menaces de sécurité que les autres distributions Linux. L'empreinte réduite de Bottlerocket permet de réduire les coûts en utilisant moins de ressources de stockage, de calcul et de réseau.
- Sécurité améliorée grâce aux mises à jour automatiques du système d'exploitation – les mises à jour de Bottlerocket sont appliquées en une seule unité et peuvent être annulées si nécessaire.

Cela élimine le risque de mises à jour corrompues ou échouées qui peuvent laisser le système dans un état inutilisable. Avec Bottlerocket, les mises à jour de sécurité peuvent être appliquées automatiquement dès qu'elles sont disponibles, de manière peu perturbatrice, et être annulées en cas de défaillance.

- Support premium : à AWS condition que les versions de Bottlerocket sur Amazon EC2 soient couvertes par les mêmes plans de AWS support qui couvrent également AWS des services tels qu'Amazon, EC2 Amazon EKS et Amazon ECR.

Considérations

Tenez compte des éléments suivants lorsque vous utilisez Bottlerocket pour votre type d'AMI :

- Bottlerocket prend en charge les EC2 instances Amazon avec `x86_64` et les processeurs. `arm64`
- Bottlerocket prend en charge les instances Amazon EC2 avec. GPUs Pour de plus amples informations, veuillez consulter the section called "Utiliser le GPU EKS Linux AMIs".
- Les images Bottlerocket n'incluent pas de serveur SSH ni de shell. Vous pouvez utiliser des méthodes out-of-band d'accès pour autoriser le SSH. Ces approches activent le conteneur d'administration et transmettent certaines étapes de configuration de l'action d'amorçage avec des données utilisateur. Pour plus d'informations, consultez les sections suivantes de Bottlerocket OS sur : GitHub
  - Exploration
  - Conteneur d'administration
  - Paramètres Kubernetes
- Bottlerocket utilise différents types de conteneur :
  - Par défaut, est activé un conteneur de contrôle. Ce conteneur exécute l'agent AWS Systems Manager que vous pouvez utiliser pour exécuter des commandes ou démarrer des sessions shell sur des instances Amazon EC2 Bottlerocket. Pour plus d'informations, consultez la section Configuration du gestionnaire de session dans le guide de l'utilisateur de AWS Systems Manager.
  - Un conteneur d'administration est activé si une clé SSH est fournie lors de la création du groupe de nœuds. Nous recommandons d'utiliser le conteneur d'administration uniquement pour les scénarios de développement et de test. Nous ne recommandons pas de l'utiliser pour les environnements de production. Pour plus d'informations, consultez Conteneur d'administration sur GitHub.

En savoir plus

Pour plus d'informations sur l'utilisation de Bottlerocket optimisé pour Amazon EKS AMIs, consultez les sections suivantes :

- Pour plus de détails sur Bottlerocket, consultez la documentation de Bottlerocket.
- Pour les ressources d'informations sur les versions, voir the section called "Obtenir les informations sur la version".
- Pour utiliser Bottlerocket avec des groupes de nœuds gérés, consultez the section called "Groupes de nœuds gérés".
- Pour lancer des nœuds Bottlerocket autogérés, consultez the section called "Flacon Rocket".
- Pour récupérer la dernière version IDs du Bottlerocket AMIs optimisé pour Amazon EKS, consultez. the section called "Obtenir les derniers identifiants"
- Pour plus de détails sur la prise en charge en matière de conformité, veuillez consulter la rubrique the section called "Prise en charge de la conformité".

Récupérer les informations relatives à la version AMI de Bottlerocket

Chaque version de Bottlerocket AMI comprend différentes versions de kubelet, du noyau Bottlerocket et de containerd. Les variantes AMI accélérées comprennent également différentes versions du pilote NVIDIA. Vous trouverez ces informations sur la version dans la rubrique OS de la documentation Bottlerocket. À partir de cette page, accédez à la sous-rubrique Informations sur la version qui vous intéresse.

La documentation Bottlerocket peut parfois être en retard par rapport aux versions disponibles sur GitHub. Vous trouverez la liste des modifications apportées aux dernières versions dans les publications sur GitHub.

Récupérer les ID d'AMI Bottlerocket recommandés

Lors du déploiement de nœuds, vous pouvez spécifier un identifiant pour une image Amazon Machine Image (AMI) optimisée pour Amazon EKS pré-créée. Pour récupérer un identifiant AMI qui correspond à la configuration souhaitée, interrogez l'API AWS Systems Manager Parameter Store. L'utilisation de cette API élimine le besoin de rechercher manuellement les identifiants AMI optimisés pour Amazon EKS. Pour plus d'informations, consultez GetParameter. Le principal IAM que vous utilisez doit disposer de l'autorisation IAM `ssm:GetParameter` pour récupérer les métadonnées de l'AMI optimisée pour Amazon EKS.

Vous pouvez récupérer l'ID d'image de la dernière AMI Bottlerocket optimisée pour Amazon EKS recommandée à l'aide de la commande AWS CLI suivante, qui utilise le sous-paramètre `image_id`. Si nécessaire, apportez les modifications suivantes à la commande, puis exécutez la commande modifiée :

- Remplacez *`kubernetes-version`* par une platform-version prise en charge.
- Remplacez *`-flavor`* par l'une des options suivantes.
  - Supprimez *`-flavor`* pour les variantes sans GPU.
  - Utilisez *`-nvidia`* pour les variantes compatibles avec le GPU.
  - Utilisez *`-fips`* pour les variantes compatibles avec FIPS.
- Remplacez *`architecture`* par l'une des options suivantes.
  - Utilisez *`x86_64`* pour les instances basées sur x86.
  - Utilisez *`arm64`* pour les instances ARM.
- Remplacez *`region-code`* par une région AWS prise en charge par Amazon EKS pour laquelle vous voulez obtenir l'ID d'AMI.

```
aws ssm get-parameter --name /aws/service/bottlerocket/aws-k8s-kubernetes-version-
flavor/architecture/latest/image_id \
    --region region-code --query "Parameter.Value" --output text
```

Voici un exemple de commande après remplacement des espaces réservés.

```
aws ssm get-parameter --name /aws/service/bottlerocket/aws-k8s-1.31/x86_64/latest/
image_id \
    --region us-west-2 --query "Parameter.Value" --output text
```

L'exemple qui suit illustre un résultat.

```
ami-1234567890abcdef0
```

Respect des exigences en matière de conformité avec Bottlerocket

Bottlerocket est conforme aux recommandations définies par plusieurs organisations :

- Un Référentiel CIS spécifique a été défini pour Bottlerocket. Dans une configuration par défaut, l'image Bottlerocket applique déjà la majorité des contrôles requis par le profil de configuration CIS

de niveau 1. Vous pouvez implémenter les contrôles requis pour un profil de configuration CIS de niveau 2. Pour plus d'informations, consultez Validation de l'AMI Bottlerocket optimisée pour Amazon EKS par rapport au référentiel CIS sur le blog AWS.

- Grâce à son ensemble de fonctionnalités optimisées et à sa surface d'attaque réduite, Bottlerocket nécessite moins de configuration pour satisfaire aux exigences PCI DSS. Le Référentiel CIS pour Bottlerocket constitue une excellente référence pour le renforcement de la sécurité et vous aide à respecter les exigences liées aux normes de configuration sécurisée définies dans l'exigence PCI DSS 2.2. Vous pouvez également utiliser Fluent Bit pour satisfaire aux exigences en matière de journalisation d'audit au niveau du système d'exploitation, conformément à l'exigence PCI DSS 10.2. AWS publie régulièrement de nouvelles images Bottlerocket mises à jour (corrigées) afin de vous aider à répondre à l'exigence PCI DSS 6.2 (pour v3.2.1) et à l'exigence 6.3.3 (pour v4.0).
- Bottlerocket est une fonctionnalité admissible à HIPAA, autorisée pour les charges de travail réglementées sur Amazon EC2 et Amazon EKS. Pour plus d'informations, consultez la Référence des services admissibles à HIPAA.
- Les AMI Bottlerocket sont disponibles et préconfigurées pour utiliser des modules cryptographiques validés FIPS 140-3. Cela inclut le module cryptographique Amazon Linux 2023 Kernel Crypto API et le module cryptographique AWS-LC. Pour de plus amples informations, consultez the section called "Bottlerocket FIPS AMIs".

Préparez vos nœuds de travail à la norme FIPS avec Bottlerocket FIPS AMIs

La norme Federal Information Processing Standard (FIPS) Publication 140-3 est un standard gouvernemental des États-Unis et du Canada qui définit les exigences de sécurité applicables aux modules cryptographiques protégeant des informations sensibles. Bottlerocket facilite l'adhésion à la norme FIPS en proposant un AMIs noyau FIPS.

Ils AMIs sont préconfigurés pour utiliser des modules cryptographiques validés par la norme FIPS 140-3. Cela inclut le module cryptographique de l'API Kernel Crypto d'Amazon Linux 2023 et le module cryptographique AWS-LC.

L'utilisation de Bottlerocket FIPS AMIs rend vos nœuds de travail « prêts pour la norme FIPS », mais pas automatiquement « conformes à la norme FIPS ». Pour plus d'informations, consultez la norme fédérale de traitement de l'information (FIPS) 140-3.

Considérations

- Si votre cluster utilise des sous-réseaux isolés, le point de terminaison FIPS d'Amazon ECR risque de ne pas être accessible. Cela peut entraîner l'échec de l'amorçage des nœuds. Assurez-vous que votre configuration réseau permet d'accéder aux points de terminaison FIPS nécessaires. Pour plus d'informations, consultez la section Accès à une ressource via un point de terminaison VPC de ressource dans le AWS PrivateLink Guide.
- Si votre cluster utilise un sous-réseau avec PrivateLink, les extractions d'images échoueront car les points de terminaison FIPS Amazon ECR ne sont pas disponibles via. PrivateLink

Création d'un groupe de nœuds géré avec une AMI Bottlerocket FIPS

L'AMI FIPS Bottlerocket est disponible en quatre variantes pour prendre en charge vos charges de travail :

- `BOTTLEROCKET_x86_64_FIPS`
- `BOTTLEROCKET_ARM_64_FIPS`
- `BOTTLEROCKET_x86_64_NVIDIA_FIPS`
- `BOTTLEROCKET_ARM_64_NVIDIA_FIPS`

Pour créer un groupe de nœuds géré utilisant une AMI Bottlerocket FIPS, choisissez le type d'AMI approprié lors du processus de création. Pour de plus amples informations, veuillez consulter the section called "Créer".

Pour plus d'informations sur la sélection des variantes compatibles FIPS, consultez the section called "Obtenir les derniers identifiants".

Désactiver le point de terminaison FIPS pour les régions non prises en charge AWS

Les systèmes FIPS Bottlerocket AMIs sont pris en charge directement aux États-Unis d'Amérique, y compris dans les régions (américaines). AWS GovCloud Pour AWS les régions où AMIs ils sont disponibles mais ne sont pas directement pris en charge, vous pouvez toujours les utiliser AMIs en créant un groupe de nœuds gérés avec un modèle de lancement.

Les AMI Bottlerocket FIPS s'appuient sur le point de terminaison FIPS d'Amazon ECR lors de l'amorçage, mais ce point de terminaison n'est généralement pas disponible en dehors des États-Unis. Pour utiliser l'AMI pour son noyau FIPS dans les AWS régions où le point de terminaison

Amazon ECR FIPS n'est pas disponible, procédez comme suit pour désactiver le point de terminaison FIPS :

1. Créez un nouveau fichier de configuration avec le contenu ci-dessous, ou ajoutez-le à votre fichier de configuration existant.

```
[default]
use_fips_endpoint=false
```

1. Encodez le contenu du fichier au format Base64.
2. Dans la section `UserData` de votre modèle de lancement, ajoutez la chaîne encodée ci-dessus au format TOML :

```
[settings.aws]
config = "<your-base64-encoded-string>"
```

Pour les autres paramètres, consultez la description des paramètres de Bottlerocket sur. GitHub

Voici un exemple de `UserData` d'un modèle de lancement :

```
[settings]
motd = "Hello from eksctl!"
[settings.aws]
config = "W2RlZmF1bHRdCnVzZV9maXBzX2VuZHBvaW50PWZhbHNlCg==" # Base64-encoded string.
[settings.kubernetes]
api-server = "<api-server-endpoint>"
cluster-certificate = "<cluster-certificate-authority>"
cluster-name = "<cluster-name>"
...<other-settings>
```

Pour plus d'informations sur la création d'un modèle de lancement contenant des données utilisateur, consultez the section called "Modèles de lancement".

Créer des nœuds avec des AMI Ubuntu Linux optimisées

Canonical a établi un partenariat avec Amazon EKS pour créer des AMI de nœuds que vous pouvez utiliser dans vos clusters.

Canonical fournit une image du système d'exploitation Kubernetes Node conçue pour l'utilisateur. Cette image Ubuntu réduite est optimisée pour Amazon EKS et inclut le noyau AWS personnalisé qui est développé conjointement avec AWS. Pour plus d'informations, consultez Ubuntu sur Amazon Elastic Kubernetes Service (EKS) et the section called "Ubuntu Linux". Pour plus d'informations sur la prise en charge, consultez la section Logiciels tiers de Questions fréquentes sur AWS Premium Support.

Créez des nœuds avec Windows optimisé AMIs

Les versions optimisées pour Windows Amazon EKS AMIs sont basées sur Windows Server 2019, Windows Server 2022 et Windows Server 2025. Elles sont configurées pour servir d'image de base pour les nœuds Amazon EKS. Par défaut, ils AMIs incluent les composants suivants :

- kubelet
- kube-proxy
- AWS Authentificateur IAM pour Kubernetes
- csi-proxy
- containerd

> **Note**
>
> Vous pouvez suivre les événements de sécurité ou de confidentialité pour Windows Server à l'aide du Guide de mise à jour de sécurité Microsoft.

Amazon EKS propose AMIs des offres optimisées pour les conteneurs Windows dans les variantes suivantes :

- AMI Windows Server 2019 Core optimisée pour Amazon EKS
- AMI Windows Server 2019 Full optimisée pour Amazon EKS
- AMI Windows Server 2022 Core optimisée pour Amazon EKS
- AMI Windows Server 2022 Full optimisée pour Amazon EKS
- AMI principale pour Windows Server 2025 optimisée pour Amazon EKS
- AMI complète pour Windows Server 2025 optimisée pour Amazon EKS

⚠ Important

- L'AMI Windows Server 20H2 Core optimisée pour Amazon EKS est obsolète. Aucune nouvelle version de cette AMI ne sera publiée.
- Pour garantir que vous disposez des dernières mises à jour de sécurité par défaut, Amazon EKS gère Windows optimisé au cours AMIs des 4 derniers mois. Chaque nouvelle AMI sera disponible pendant 4 mois à compter de la date de publication initiale. Après cette période, AMIs les anciens fichiers deviennent privés et ne sont plus accessibles. Nous vous encourageons à utiliser la dernière version AMIs afin d'éviter les failles de sécurité et de perdre l'accès aux versions plus anciennes AMIs qui ont atteint la fin de leur durée de vie prise en charge. Bien que nous ne puissions pas garantir que nous puissions fournir un accès à AMIs ce qui a été rendu privé, vous pouvez demander l'accès en déposant un ticket auprès du AWS Support.

Calendrier des versions

Le tableau suivant répertorie les dates de publication et de fin de support pour les versions Windows sur Amazon EKS. Si une date de fin est vide, c'est parce que la version est toujours prise en charge.

Version Windows	Version d'Amazon EKS	Fin de support pour Amazon EKS
Windows Server 2025 Core	27/01/2026	
Windows Server 2025 complet	27/01/2026	
Windows Server 2022 Core	17 octobre 2022	
Windows Server 2022 Full	17 octobre 2022	
Windows Server 20H2 Core	8/12/2021	09/08/2022
Windows Server 2004 Core	19/08/2020	14/12/2021
Windows Server 2019 Core	07/10/2019	
Windows Server 2019 Full	07/10/2019	

Version Windows	Version d'Amazon EKS	Fin de support pour Amazon EKS
Windows Server 1909 Core	07/10/2019	8/12/2020

Paramètres de configuration du script d'amorçage

Lorsque vous créez un nœud Windows, un script sur le nœud permet la configuration de différents paramètres. En fonction de votre configuration, ce script peut être trouvé sur le nœud à un emplacement similaire à :`C:\Program Files\Amazon\EKS\Start-EKSBootstrap.ps1`. Vous pouvez indiquer des valeurs de paramètres personnalisées en les spécifiant comme arguments dans le script d'amorçage. Par exemple, vous pouvez mettre à jour les données utilisateur dans le modèle de lancement. Pour de plus amples informations, veuillez consulter the section called “Données utilisateur Amazon EC2”.

Le script comprend les paramètres de ligne de commande suivants :

- `-EKSClusterName` : spécifie le nom du cluster Amazon EKS que ce composant master doit rejoindre.
- `-KubeletExtraArgs` : spécifie des arguments supplémentaires pour `kubelet` (facultatif).
- `-KubeProxyExtraArgs` : spécifie des arguments supplémentaires pour `kube-proxy` (facultatif).
- `-APIServerEndpoint` : spécifie le point de terminaison du serveur d'API de cluster Amazon EKS (facultatif). Uniquement valable lorsqu'il est utilisé avec `-Base64ClusterCA`. Évite d'appeler `Get-EKSCluster`.
- `-Base64ClusterCA` : spécifie le contenu de l'autorité de certification du cluster codée en base64 (facultatif). Uniquement valable lorsqu'il est utilisé avec `-APIServerEndpoint`. Évite d'appeler `Get-EKSCluster`.
- `-DNSClusterIP` : remplace l'adresse IP à utiliser pour les requêtes DNS au sein du cluster (facultatif). La valeur par défaut est `10.100.0.10` ou `172.20.0.10` en fonction de l'adresse IP de l'interface principale.
- `-ServiceCIDR` : remplace la plage d'adresses IP du service Kubernetes à partir de laquelle les services du cluster sont adressés. La valeur par défaut est `172.20.0.0/16` ou `10.100.0.0/16` en fonction de l'adresse IP de l'interface principale.
- `-ExcludedSnatCIDRs`— Liste des adresses `IPv4` CIDRs à exclure de la traduction des adresses réseau source (SNAT). Cela signifie que l'adresse IP privée du pod, qui est adressable par VPC,

ne serait pas traduite en adresse IP de l'adresse principale `IPv4` de l'ENI de l'instance pour le trafic sortant. Par défaut, le CIDR `IPv4` du VPC pour le nœud Amazon EKS Windows est ajouté. La spécification CIDRs de ce paramètre exclut également le paramètre spécifié CIDRs. Pour de plus amples informations, veuillez consulter the section called "Trafic sortant".

En plus des paramètres de la ligne de commande, vous pouvez également spécifier certains paramètres de la variable d'environnement. Lorsqu'un paramètre de ligne de commande est spécifié, il est prioritaire sur la variable d'environnement correspondante. La ou les variables d'environnement doivent être définies au niveau de la machine (ou du système), car le script d'amorçage ne lira que les variables au niveau de la machine.

Le script prend en compte les variables d'environnement suivantes :

- `SERVICE_IPV4_CIDR`— Reportez-vous au paramètre de la ligne de commande `ServiceCIDR` pour la définition.
- `EXCLUDED_SNAT_CIDRS`— Il doit s'agir d'une chaîne de caractères séparée par des virgules. Reportez-vous au paramètre de la ligne de commande `ExcludedSnatCIDRs` pour la définition.

Prise en charge de l'authentification gMSA

Les pods Windows Amazon EKS autorisent différents types d'authentification de compte de service géré de groupe (gMSA).

- Amazon EKS prend en charge les identités de domaine Active Directory à des fins d'authentification. Pour plus d'informations sur les GMSA joints à un domaine, consultez Windows Authentication on Amazon EKS Windowspods sur le blog. AWS
- Amazon EKS propose un plug-in qui permet aux nœuds non-domain-joined Windows de récupérer les informations d'identification GMSA à l'aide d'une identité utilisateur portable. Pour plus d'informations sur les GMSA sans domaine, consultez Domainless Windows Authentication for Amazon EKS Windowspods sur le blog. AWS

Images de conteneur en cache

Certaines images de conteneur AMIs optimisées pour Amazon EKS pour Windows sont mises en cache pour l'`containerd`exécution. Les images de conteneur sont mises en cache lors de la création personnalisée à AMIs l'aide de composants de génération gérés par Amazon. Pour de plus

amples informations, veuillez consulter the section called "Utilisation du composant de génération géré par Amazon".

Les images de conteneur mises en cache suivantes sont destinées à l'exécution `containerd` :

- `amazonaws.com/eks/pause-windows`
- `mcr.microsoft.com/windows/nanoserver`
- `mcr.microsoft.com/windows/servercore`

En savoir plus

Pour plus d'informations sur l'utilisation de Windows optimisé pour Amazon EKS AMIs, consultez les sections suivantes :

- Pour en savoir plus sur l'exécution des charges de travail sur un système Windows accéléré optimisé pour Amazon EKS AMIs, consultezthe section called "Configurer les AMI GPU Windows".
- Pour utiliser Windows avec des groupes de nœuds gérés, consultez the section called "Groupes de nœuds gérés".
- Pour lancer des nœuds Windows autogérés, veuillez consulter la rubrique the section called "Windows".
- Pour plus d'informations sur la version, consultez the section called "Obtenir les informations sur la version".
- Pour accéder à la dernière version IDs de Windows optimisée pour Amazon EKS AMIs, consultezthe section called "Recevez les dernières nouvelles IDs".
- Pour utiliser Amazon EC2 Image Builder afin de créer des fenêtres personnalisées optimisées pour Amazon EKS AMIs, consultez. the section called "Builds personnalisés"
- Pour connaître les meilleures pratiques, consultez la section Gestion des AMI Windows optimisées pour Amazon EKS dans le Guide des meilleures pratiques EKS.

Créer des nœuds Windows Server 2022 autogérés avec **`eksctl`**

Vous pouvez utiliser le fichier `test-windows-2022.yaml` suivant comme référence pour la création de nœuds Windows Server 2022 autogérés. Remplacez la *`valeur d'exemple`* par vos propres valeurs.

> **Note**
>
> Vous devez utiliser `eksctl` version 0.116.0 ou ultérieure pour exécuter des nœuds Windows Server 2022 autogérés.

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: windows-2022-cluster
  region: region-code
  version: '1.33'

nodeGroups:
  - name: windows-ng
    instanceType: m5.2xlarge
    amiFamily: WindowsServer2022FullContainer
    volumeSize: 100
    minSize: 2
    maxSize: 3
  - name: linux-ng
    amiFamily: AmazonLinux2
    minSize: 2
    maxSize: 3
```

Les groupes de nœuds peuvent ensuite être créés à l'aide de la commande suivante.

```
eksctl create cluster -f test-windows-2022.yaml
```

Récupérez les informations relatives à la version des AMI Windows

Cette rubrique répertorie les versions de Windows optimisé pour Amazon EKS AMIs et les versions correspondantes de kubelet, containerd etcsi-proxy.

Les métadonnées d'AMI optimisées pour Amazon EKS, notamment l'ID d'AMI, peuvent être extraites par programmation pour chaque variante. Pour de plus amples informations, veuillez consulter the section called “Recevez les dernières nouvelles IDs”.

AMIs sont versionnés en fonction de la version de Kubernetes et de la date de sortie de l'AMI au format suivant :

```
k8s_major_version.k8s_minor_version-release_date
```

> **Note**
>
> Les groupes de nœuds gérés par Amazon EKS sont compatibles avec les versions de novembre 2022 et ultérieures de Windows AMIs.

Pour recevoir des notifications concernant toutes les modifications apportées aux fichiers sources de cette page de documentation spécifique, vous pouvez vous abonner à l'URL suivante à l'aide d'un lecteur RSS :

```
https://github.com/awsdocs/amazon-eks-user-guide/commits/mainline/latest/ug/nodes/eks-ami-versions-windows.adoc.atom
```

AMI principale optimisée pour Windows Server 2025 pour Amazon EKS

Les tableaux suivants répertorient les versions actuelles et précédentes de l'AMI Windows Server 2025 Core optimisée pour Amazon EKS.

Example

Version 1.35 de Kubernetes

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
`1.35-2026-01-22`	`1.35.0`	`2.1.6`	`1.2.1`	

AMI complète optimisée pour Windows Server 2025 pour Amazon EKS

Les tableaux suivants répertorient les versions actuelles et précédentes de l'AMI complète Windows Server 2025 optimisée pour Amazon EKS.

Example

Version 1.35 de Kubernetes

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
`1.35-2026-01-22`	`1.35.0`	`2.1.6`	`1.2.1`	

AMI Windows Server 2022 Core optimisée pour Amazon EKS

Les tableaux suivants répertorient les versions actuelles et précédentes de l'AMI Windows Server 2022 Core optimisée pour Amazon EKS.

Example

Version 1.35 de Kubernetes

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
`1.35-2026-01-22`	`1.35.0`	`2.1.6`	`1.2.1`	

Version 1.34 de Kubernetes

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
`1.34-2026.01.22`	`1.34.2`	`2.1.6`	`1.2.1`	Mise à niveau `containerd` vers `2.1.6`.
`1.34-2025.12.15`	`1.34.2`	`2.1.4`	`1.2.1`	

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
1.34-2025.11.14	1.34.1	2.1.4	1.2.1	
1.34-2025.10.18	1.34.1	2.1.4	1.2.1	
1.34-2025.09.13	1.34.0	2.1.4	1.2.1	

Kubernetes version 1.33

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
1.33-2026.01.22	1.33.5	1.7.30	1.2.1	Mise à niveau `containerd` vers `1.7.30`.
1.33-2025.12.15	1.33.5	1.7.28	1.2.1	
1.33-2025.11.14	1.33.5	1.7.28	1.2.1	
1.33-2025.10.18	1.33.5	1.7.28	1.2.1	
1.33-2025.09.13	1.33.4	1.7.28	1.2.1	Le plug-in GMSA modifié enregistre les événements Windows
1.33-2025.08.18	1.33.3	1.7.27	1.2.1	

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
1.33-2025.07.16	1.33.1	1.7.27	1.2.1	
1.33-2025.06.13	1.33.1	1.7.27	1.2.1	
1.33-2025.05.17	1.33.1	1.7.27	1.2.1	

Kubernetes version 1.32

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
1.32-2026.01.22	1.32.9	1.7.30	1.2.1	Mise à niveau `containerd` vers `1.7.30`.
1.32-2025.12.15	1.32.9	1.7.28	1.2.1	
1.32-2025.11.14	1.32.9	1.7.28	1.2.1	
1.32-2025.10.18	1.32.9	1.7.28	1.2.1	
1.32-2025.09.13	1.32.8	1.7.28	1.2.1	Le plug-in GMSA modifié enregistre les événements Windows
1.32-2025.08.18	1.32.7	1.7.27	1.2.1	

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
1.32-2025.07.16	1.32.5	1.7.27	1.2.1	
1.32-2025.06.13	1.32.5	1.7.27	1.2.1	Mise à niveau `containerd` vers `1.7.27`.
1.32-2025.05.17	1.32.5	1.7.20	1.2.1	
1.32-2025.04.14	1.32.1	1.7.20	1.1.3	
1.32-2025.03.14	1.32.1	1.7.20	1.1.3	
1.32-2025.02.18	1.32.1	1.7.20	1.1.3	
1.32-2025.01.15	1.32.0	1.7.20	1.1.3	

Kubernetes version 1.31

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
1.31-2026.01.22	1.31.13	1.7.30	1.2.1	Mise à niveau `containerd` vers `1.7.30`.
1.31-2025.12.15	1.31.13	1.7.28	1.2.1	

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
1.31-2025.11.14	1.31.13	1.7.28	1.2.1	
1.31-2025.10.18	1.31.13	1.7.28	1.2.1	
1.31-2025.09.13	1.31.12	1.7.28	1.2.1	Le plug-in GMSA modifié enregistre les événements Windows
1.31-2025.08.18	1.31.11	1.7.27	1.2.1	
1.31-2025.07.16	1.31.9	1.7.27	1.2.1	
1.31-2025.06.13	1.31.9	1.7.27	1.2.1	Mise à niveau `containerd` vers `1.7.27`.
1.31-2025.05.17	1.31.9	1.7.20	1.2.1	
1.31-2025.04.14	1.31.5	1.7.20	1.1.3	
1.31-2025.03.14	1.31.5	1.7.20	1.1.3	
1.31-2025.02.15	1.31.5	1.7.20	1.1.3	
1.31-2025.01.15	1.31.4	1.7.20	1.1.3	

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
1.31-2025.01.01	1.31.4	1.7.20	1.1.3	Comprend des correctifs pour CVE-2024-9042 .
1.31-2024.12.13	1.31.3	1.7.20	1.1.3	
1.31-2024.11.12	1.31.1	1.7.20	1.1.3	
1.31-2024.10.08	1.31.1	1.7.20	1.1.3	
1.31-2024.10.01	1.31.1	1.7.20	1.1.3	
1.31-2024.09.10	1.31.0	1.7.20	1.1.3	

Version 1.30 de Kubernetes

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
1.30-2026.01.22	1.30.14	1.7.30	1.2.1	Mise à niveau containerd vers 1.7.30.
1.30-2025.12.15	1.30.14	1.7.28	1.2.1	
1.30-2025.11.21	1.30.14	1.7.28	1.2.1	

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
1.30-2025.10.18	1.30.14	1.7.28	1.2.1	
1.30-2025.09.13	1.30.14	1.7.28	1.2.1	Le plug-in GMSA modifié enregistre les événements Windows
1.30-2025.08.18	1.30.14	1.7.27	1.2.1	
1.30-2025.07.16	1.30.13	1.7.27	1.2.1	
1.30-2025.06.13	1.30.13	1.7.27	1.2.1	Mise à niveau `containerd` vers `1.7.27`.
1.30-2025.05.17	1.30.13	1.7.20	1.2.1	
1.30-2025.04.14	1.30.9	1.7.20	1.1.3	
1.30-2025.03.14	1.30.9	1.7.20	1.1.3	Mise à niveau `containerd` vers `1.7.20`.
1.30-2025.02.15	1.30.9	1.7.14	1.1.3	
1.30-2025.01.15	1.30.8	1.7.14	1.1.3	

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
1.30-2025.01.01	1.30.8	1.7.14	1.1.3	Comprend des correctifs pour CVE-2024-9042 .
1.30-2024.12.11	1.30.7	1.7.14	1.1.3	
1.30-2024.11.12	1.30.4	1.7.14	1.1.3	
1.30-2024.10.08	1.30.4	1.7.14	1.1.3	
1.30-2024.09.10	1.30.2	1.7.14	1.1.3	
1.30-2024.08.13	1.30.2	1.7.14	1.1.3	
1.30-2024.07.10	1.30.2	1.7.14	1.1.2	Comprend des correctifs pour CVE-2024-5321 .
1.30-2024.06.17	1.30.0	1.7.14	1.1.2	Mise à niveau `containerd` vers `1.7.14`.
1.30-2024.05.15	1.30.0	1.6.28	1.1.2	

Version 1.29 de Kubernetes

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
1.29-2026.01.22	1.29.15	1.7.30	1.2.1	Mise à niveau containerd vers 1.7.30.
1.29-2025.12.15	1.29.15	1.7.28	1.2.1	
1.29-2025.11.14	1.29.15	1.7.28	1.2.1	
1.29-2025.10.18	1.29.15	1.7.28	1.2.1	
1.29-2025.09.13	1.29.15	1.7.28	1.2.1	Le plug-in GMSA modifié enregistre les événements Windows
1.29-2025.08.18	1.29.15	1.7.27	1.2.1	
1.29-2025.07.16	1.29.15	1.7.27	1.2.1	
1.29-2025.06.13	1.29.15	1.7.27	1.2.1	Mise à niveau containerd vers 1.7.27.
1.29-2025.05.17	1.29.15	1.7.20	1.2.1	
1.29-2025.04.14	1.29.13	1.7.20	1.1.3	

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
1.29-2025.03.14	1.29.13	1.7.20	1.1.3	Mise à niveau `containerd` vers `1.7.20`.
1.29-2025.02.15	1.29.13	1.7.14	1.1.3	
1.29-2025.01.15	1.29.12	1.7.14	1.1.3	
1.29-2025.01.01	1.29.12	1.7.14	1.1.3	Comprend des correctifs pour `CVE-2024-9042` .
1.29-2024.12.11	1.29.10	1.7.14	1.1.3	
1.29-2024.11.12	1.29.8	1.7.14	1.1.3	
1.29-2024.10.08	1.29.8	1.7.14	1.1.3	
1.29-2024.09.10	1.29.6	1.7.14	1.1.3	
1.29-2024.08.13	1.29.6	1.7.14	1.1.3	
1.29-2024.07.10	1.29.6	1.7.11	1.1.2	Comprend des correctifs pour `CVE-2024-5321` .

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
1.29-2024.06.17	1.29.3	1.7.11	1.1.2	
1.29-2024.05.15	1.29.3	1.7.11	1.1.2	Mise à niveau `containerd` vers `1.7.11`. Mise à niveau `kubelet` vers `1.29.3`.
1.29-2024.04.09	1.29.0	1.6.28	1.1.2	Mise à niveau `containerd` vers `1.6.28`. CNI reconstruit et `csi-proxy` utilise `golang` `1.22.1`.
1.29-2024.03.12	1.29.0	1.6.25	1.1.2	
1.29-2024.02.13	1.29.0	1.6.25	1.1.2	

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
1.29-2024.02.06	1.29.0	1.6.25	1.1.2	Correction d'un bug qui entraînait la suppression incorrecte de l'image de pause par le processus de récupération de mémoire kubelet.
1.29-2024.01.11	1.29.0	1.6.18	1.1.2	Windows Update autonome exclu KB5034439sur Windows Server 2022 Core AMIs. La base de connaissances s'appliqu e uniquement aux installations Windows avec une partition WinRE distincte , qui ne sont incluses dans aucun de nos systèmes Windows AMIs optimisés pour Amazon EKS.

Kubernetes version 1.28

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
1.28-2025.11.14	1.28.15	1.7.28	1.2.1	
1.28-2025.10.18	1.28.15	1.7.28	1.2.1	
1.28-2025.09.13	1.28.15	1.7.28	1.2.1	Le plug-in GMSA modifié enregistre les événements Windows
1.28-2025.08.18	1.28.15	1.7.27	1.2.1	
1.28-2025.07.16	1.28.15	1.7.27	1.2.1	
1.28-2025.06.13	1.28.15	1.7.27	1.2.1	Mise à niveau `containerd` vers `1.7.27`.
1.28-2025.05.17	1.28.15	1.7.20	1.2.1	
1.28-2025.04.14	1.28.15	1.7.20	1.1.3	
1.28-2025.03.14	1.28.15	1.7.20	1.1.3	Mise à niveau `containerd` vers `1.7.20`.
1.28-2025.02.15	1.28.15	1.7.14	1.1.3	

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
1.28-2025.01.15	1.28.15	1.7.14	1.1.3	
1.28-2025-01-01	1.28.15	1.7.14	1.1.3	Comprend des correctifs pour CVE-2024-9042 .
1.28-2024.12.11	1.28.15	1.7.14	1.1.3	
1.28-2024.11.12	1.28.13	1.7.14	1.1.3	
1.28-2024.10.08	1.28.13	1.7.14	1.1.3	
1.28-2024.09.10	1.28.11	1.7.14	1.1.3	
1.28-2024.08.13	1.28.11	1.7.14	1.1.3	
1.28-2024.07.10	1.28.11	1.7.11	1.1.2	Comprend des correctifs pour CVE-2024-5321 .
1.28-2024.06.17	1.28.8	1.7.11	1.1.2	Mise à niveau `containerd` vers `1.7.11`.

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
1.28-2024.05.14	1.28.8	1.6.28	1.1.2	Mise à niveau `containerd` vers `1.6.28`. Mise à niveau `kubelet` vers `1.28.8`.
1.28-2024.04.09	1.28.5	1.6.25	1.1.2	Mise à niveau `containerd` vers `1.6.25`. CNI reconstruit et `csi-proxy` utilise `golang 1.22.1`.
1.28-2024.03.12	1.28.5	1.6.18	1.1.2	
1.28-2024.02.13	1.28.5	1.6.18	1.1.2	

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
1.28-2024.01.11	1.28.5	1.6.18	1.1.2	Windows Update autonome exclu KB5034439sur Windows Server 2022 Core AMIs. La base de connaissances s'appliqu e uniquement aux installations Windows avec une partition WinRE distincte , qui ne sont incluses dans aucun de nos systèmes Windows AMIs optimisés pour Amazon EKS.
1.28-2023.12.12	1.28.3	1.6.18	1.1.2	
1.28-2023.11.14	1.28.3	1.6.18	1.1.2	Comprend des correctifs pour CVE-2023-5528 .

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
1.28-2023.10.19	1.28.2	1.6.18	1.1.2	Mise à niveau containerd vers 1.6.18. Ajout de nouvelles variables d'environnement pour le script d'amorçage (SERVICE_IPV4_CIDR et EXCLUDED_SNAT_CIDRS).
1.28-2023-09.27	1.28.2	1.6.6	1.1.2	Correction d'une recommandation de sécurité dans kubelet.
1.28-2023.09.12	1.28.1	1.6.6	1.1.2	

AMI Windows Server 2022 Full optimisée pour Amazon EKS

Les tableaux suivants répertorient les versions actuelles et précédentes de l'AMI complète Windows Server 2022 optimisée pour Amazon EKS.

Example

Version 1.35 de Kubernetes

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
`1.35-2026-01-22`	`1.35.0`	`2.1.6`	`1.2.1`	

Version 1.34 de Kubernetes

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
`1.34-2026.01.22`	`1.34.2`	`2.1.6`	`1.2.1`	Mise à niveau `containerd` vers `2.1.6`.
`1.34-2025.12.15`	`1.34.2`	`2.1.4`	`1.2.1`	
`1.34-2025.11.14`	`1.34.1`	`2.1.4`	`1.2.1`	
`1.34-2025.10.18`	`1.34.1`	`2.1.4`	`1.2.1`	
`1.34-2025.09.13`	`1.34.0`	`2.1.4`	`1.2.1`	

Kubernetes version 1.33

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
1.33-2026.01.22	1.33.5	1.7.30	1.2.1	Mise à niveau `containerd` vers `1.7.30`.
1.33-2025.12.15	1.33.5	1.7.28	1.2.1	
1.33-2025.11.14	1.33.5	1.7.28	1.2.1	
1.33-2025.10.18	1.33.5	1.7.28	1.2.1	
1.33-2025.09.13	1.33.4	1.7.28	1.2.1	Le plug-in GMSA modifié enregistre les événements Windows
1.33-2025.08.18	1.33.3	1.7.27	1.2.1	
1.33-2025.07.16	1.33.1	1.7.27	1.2.1	
1.33-2025.06.13	1.33.1	1.7.27	1.2.1	
1.33-2025.05.17	1.33.1	1.7.27	1.2.1	

Kubernetes version 1.32

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
1.32-2026.01.22	1.32.9	1.7.30	1.2.1	Mise à niveau `containerd` vers `1.7.30`.
1.32-2025.12.15	1.32.9	1.7.28	1.2.1	
1.32-2025.11.14	1.32.9	1.7.28	1.2.1	
1.32-2025.10.18	1.32.9	1.7.28	1.2.1	
1.32-2025.09.13	1.32.8	1.7.28	1.2.1	Le plug-in GMSA modifié enregistre les événements Windows
1.32-2025.08.18	1.32.7	1.7.27	1.2.1	
1.32-2025.07.16	1.32.5	1.7.27	1.2.1	
1.32-2025.06.13	1.32.5	1.7.27	1.2.1	Mise à niveau de `containrd` vers `1.7.27`
1.32-2025.05.17	1.32.5	1.7.20	1.2.1	
1.32-2025.04.14	1.32.1	1.7.20	1.1.3	

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
1.32-2025.03.14	1.32.1	1.7.20	1.1.3	
1.32-2025.02.18	1.32.1	1.7.20	1.1.3	
1.32-2025.01.01	1.32.0	1.7.20	1.1.3	

Kubernetes version 1.31

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
1.31-2026.01.22	1.31.13	1.7.30	1.2.1	Mise à niveau `containerd` vers `1.7.30`.
1.31-2025.12.15	1.31.13	1.7.28	1.2.1	
1.31-2025.11.14	1.31.13	1.7.28	1.2.1	
1.31-2025.10.18	1.31.13	1.7.28	1.2.1	
1.31-2025.09.13	1.31.12	1.7.28	1.2.1	Le plug-in GMSA modifié enregistre les événements Windows
1.31-2025.08.18	1.31.11	1.7.27	1.2.1	

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
1.31-2025.07.16	1.31.9	1.7.27	1.2.1	
1.31-2025.06.13	1.31.9	1.7.27	1.2.1	Mise à niveau `containerd` vers `1.7.27`.
1.31-2025.05.17	1.31.9	1.7.20	1.2.1	
1.31-2025.04.14	1.31.5	1.7.20	1.1.3	
1.31-2025.03.14	1.31.5	1.7.20	1.1.3	
1.31-2025.02.15	1.31.5	1.7.20	1.1.3	
1.31-2025.01.15	1.31.4	1.7.20	1.1.3	
1.31-2025.01.01	1.31.4	1.7.20	1.1.3	Comprend des correctifs pour `CVE-2024-9042` .
1.31-2024.12.13	1.31.3	1.7.20	1.1.3	
1.31-2024.11.12	1.31.1	1.7.20	1.1.3	
1.31-2024.10.08	1.31.1	1.7.20	1.1.3	

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
1.31-2024.10.01	1.31.1	1.7.20	1.1.3	
1.31-2024.09.10	1.31.0	1.7.20	1.1.3	

Version 1.30 de Kubernetes

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
1.30-2026.01.22	1.30.14	1.7.30	1.2.1	Mise à niveau `containerd` vers `1.7.30`.
1.30-2025.12.15	1.30.14	1.7.28	1.2.1	
1.30-2025.11.21	1.30.14	1.7.28	1.2.1	
1.30-2025.10.18	1.30.14	1.7.28	1.2.1	
1.30-2025.09.13	1.30.14	1.7.28	1.2.1	Le plug-in GMSA modifié enregistre les événements Windows
1.30-2025.08.18	1.30.14	1.7.27	1.2.1	
1.30-2025.07.16	1.30.13	1.7.27	1.2.1	

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
1.30-2025.06.13	1.30.13	1.7.27	1.2.1	Mise à niveau `containerd` vers `1.7.27`.
1.30-2025.05.17	1.30.13	1.7.20	1.2.1	
1.30-2025.04.14	1.30.9	1.7.20	1.1.3	
1.30-2025.03.14	1.30.9	1.7.20	1.1.3	Mise à niveau `contianerd` vers `1.7.20`.
1.30-2025.02.15	1.30.9	1.7.14	1.1.3	
1.30-2025.01.15	1.30.8	1.7.14	1.1.3	
1.30-2025.01.01	1.30.8	1.7.14	1.1.3	Comprend des correctifs pour `CVE-2024-9042` .
1.30-2024.12.11	1.30.7	1.7.14	1.1.3	
1.30-2024.11.12	1.30.4	1.7.14	1.1.3	
1.30-2024.10.08	1.30.4	1.7.14	1.1.3	
1.30-2024.09.10	1.30.2	1.7.14	1.1.3	

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
1.30-2024.08.13	1.30.2	1.7.14	1.1.3	
1.30-2024.07.10	1.30.2	1.7.14	1.1.2	Comprend des correctifs pour CVE-2024-5321 .
1.30-2024.06.17	1.30.0	1.7.14	1.1.2	Mise à niveau containerd vers 1.7.14.
1.30-2024.05.15	1.30.0	1.6.28	1.1.2	

Version 1.29 de Kubernetes

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
1.29-2026.01.22	1.29.15	1.7.30	1.2.1	Mise à niveau containerd vers 1.7.30.
1.29-2025.12.15	1.29.15	1.7.28	1.2.1	
1.29-2025.11.14	1.29.15	1.7.28	1.2.1	
1.29-2025.10.18	1.29.15	1.7.28	1.2.1	
1.29-2025.09.13	1.29.15	1.7.28	1.2.1	Le plug-in GMSA modifié

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
				enregistre les événements Windows
1.29-2025.08.18	1.29.15	1.7.27	1.2.1	
1.29-2025.07.16	1.29.15	1.7.27	1.2.1	
1.29-2025.06.13	1.29.15	1.7.27	1.2.1	Mise à niveau `containerd` vers `1.7.27`.
1.29-2025.05.17	1.29.15	1.7.20	1.2.1	
1.29-2025.04.14	1.29.13	1.7.20	1.1.3	
1.29-2025.03.14	1.29.13	1.7.20	1.1.3	Mise à niveau `containerd` vers `1.7.20`.
1.29-2025.02.15	1.29.13	1.7.14	1.1.3	
1.29-2025.01.15	1.29.12	1.7.14	1.1.3	
1.29-2025.01.01	1.29.12	1.7.14	1.1.3	Comprend des correctifs pour `CVE-2024-9042` .

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
1.29-2024.12.11	1.29.10	1.7.14	1.1.3	
1.29-2024.11.12	1.29.8	1.7.14	1.1.3	
1.29-2024.10.08	1.29.8	1.7.14	1.1.3	
1.29-2024.09.10	1.29.6	1.7.14	1.1.3	
1.29-2024.08.13	1.29.6	1.7.14	1.1.3	
1.29-2024.07.10	1.29.6	1.7.11	1.1.2	Comprend des correctifs pour CVE-2024-5321 .
1.29-2024.06.17	1.29.3	1.7.11	1.1.2	
1.29-2024.05.15	1.29.3	1.7.11	1.1.2	Mise à niveau containerd vers 1.7.11. Mise à niveau kubelet vers 1.29.3.

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
1.29-2024.04.09	1.29.0	1.6.28	1.1.2	Mise à niveau `containerd` vers `1.6.28`. CNI reconstruit et `csi-proxy` utilise `golang` `1.22.1`.
1.29-2024.03.12	1.29.0	1.6.25	1.1.2	
1.29-2024.02.13	1.29.0	1.6.25	1.1.2	
1.29-2024.02.06	1.29.0	1.6.25	1.1.2	Correction d'un bug qui entraînait la suppression incorrecte de l'image de pause par le processus de récupération de mémoire `kubelet`.
1.29-2024.01.09	1.29.0	1.6.18	1.1.2	

Kubernetes version 1.28

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
1.28-2025.11.14	1.28.15	1.7.28	1.2.1	
1.28-2025.10.18	1.28.15	1.7.28	1.2.1	
1.28-2025.09.13	1.28.15	1.7.28	1.2.1	Le plug-in GMSA modifié enregistre les événements Windows
1.28-2025.08.18	1.28.15	1.7.27	1.2.1	
1.28-2025.07.16	1.28.15	1.7.27	1.2.1	
1.28-2025.06.13	1.28.15	1.7.27	1.2.1	Mise à niveau `containerd` vers `1.7.27`.
1.28-2025.05.17	1.28.15	1.7.20	1.2.1	
1.28-2025.04.14	1.28.15	1.7.20	1.1.3	
1.28-2025.03.14	1.28.15	1.7.20	1.1.3	Mise à niveau `containerd` vers `1.7.20`.
1.28-2025.02.15	1.28.15	1.7.14	1.1.3	

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
1.28-2025.01.15	1.28.15	1.7.14	1.1.3	
1.28-2025.01.01	1.28.15	1.7.14	1.1.3	Comprend des correctifs pour CVE-2024-9042 .
1.28-2024.12.11	1.28.15	1.7.14	1.1.3	
1.28-2024.11.12	1.28.13	1.7.14	1.1.3	
1.28-2024.10.08	1.28.13	1.7.14	1.1.3	
1.28-2024.09.10	1.28.11	1.7.14	1.1.3	
1.28-2024.08.13	1.28.11	1.7.14	1.1.3	
1.28-2024.07.10	1.28.11	1.7.11	1.1.2	Comprend des correctifs pour CVE-2024-5321 .
1.28-2024.06.17	1.28.8	1.7.11	1.1.2	Mise à niveau `containerd` vers `1.7.11`.

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
1.28-2024.05.14	1.28.8	1.6.28	1.1.2	Mise à niveau `containerd` vers `1.6.28`. Mise à niveau `kubelet` vers `1.28.8`.
1.28-2024.04.09	1.28.5	1.6.25	1.1.2	Mise à niveau `containerd` vers `1.6.25`. CNI reconstruit et `csi-proxy` utilise `golang 1.22.1`.
1.28-2024.03.12	1.28.5	1.6.18	1.1.2	
1.28-2024.02.13	1.28.5	1.6.18	1.1.2	
1.28-2024.01.09	1.28.5	1.6.18	1.1.2	
1.28-2023.12.12	1.28.3	1.6.18	1.1.2	
1.28-2023.11.14	1.28.3	1.6.18	1.1.2	Comprend des correctifs pour `CVE-2023-5528` .

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
1.28-2023.10.19	1.28.2	1.6.18	1.1.2	Mise à niveau containerd vers 1.6.18. Ajout de nouvelles variables d'environnement pour le script d'amorçage (SERVICE_IPV4_CIDR et EXCLUDED_SNAT_CIDRS).
1.28-2023-09.27	1.28.2	1.6.6	1.1.2	Correction d'une recommandation de sécurité dans kubelet.
1.28-2023.09.12	1.28.1	1.6.6	1.1.2	

AMI Windows Server 2019 Core optimisée pour Amazon EKS

Les tableaux suivants répertorient les versions actuelles et précédentes de l'AMI Windows Server 2019 Core optimisée pour Amazon EKS.

Example

Version 1.35 de Kubernetes

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
`1.35-2026-01-22`	`1.35.0`	`2.1.6`	`1.2.1`	

Version 1.34 de Kubernetes

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
`1.34-2026.01.22`	`1.34.2`	`2.1.6`	`1.2.1`	Mise à niveau `containerd` vers `2.1.6`.
`1.34-2025.12.15`	`1.34.2`	`2.1.4`	`1.2.1`	
`1.34-2025.11.14`	`1.34.1`	`2.1.4`	`1.2.1`	
`1.34-2025.10.18`	`1.34.1`	`2.1.4`	`1.2.1`	
`1.34-2025.09.13`	`1.34.0`	`2.1.4`	`1.2.1`	

Kubernetes version 1.33

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
1.33-2026.01.22	1.33.5	1.7.30	1.2.1	Mise à niveau `containerd` vers `1.7.30`.
1.33-2025.12.15	1.33.5	1.7.28	1.2.1	
1.33-2025.11.14	1.33.5	1.7.28	1.2.1	
1.33-2025.10.18	1.33.5	1.7.28	1.2.1	
1.33-2025.09.13	1.33.4	1.7.28	1.2.1	Le plug-in GMSA modifié enregistre les événements Windows
1.33-2025.08.18	1.33.3	1.7.27	1.2.1	
1.33-2025.07.16	1.33.1	1.7.27	1.2.1	
1.33-2025.06.13	1.33.1	1.7.27	1.2.1	
1.33-2025.05.17	1.33.1	1.7.27	1.2.1	

Kubernetes version 1.32

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
1.32-2026.01.22	1.32.9	1.7.30	1.2.1	Mise à niveau containerd vers 1.7.30.
1.32-2025.12.15	1.32.9	1.7.28	1.2.1	
1.32-2025.11.14	1.32.9	1.7.28	1.2.1	
1.32-2025.10.18	1.32.9	1.7.28	1.2.1	
1.32-2025.09.13	1.32.8	1.7.28	1.2.1	Le plug-in GMSA modifié enregistre les événements Windows
1.32-2025.08.18	1.32.7	1.7.27	1.2.1	
1.32-2025.07.16	1.32.5	1.7.27	1.2.1	
1.32-2025.06.13	1.32.5	1.7.27	1.2.1	Mise à niveau containerd vers 1.7.27.
1.32-2025.05.17	1.32.5	1.7.20	1.2.1	
1.32-2025.04.14	1.32.1	1.7.20	1.1.3	

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
1.32-2025.03.14	1.32.1	1.7.20	1.1.3	
1.32-2025.02.18	1.32.1	1.7.20	1.1.3	
1.32-2025.01.15	1.32.4	1.7.20	1.1.3	

Kubernetes version 1.31

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
1.31-2026.01.22	1.31.13	1.7.30	1.2.1	Mise à niveau `containerd` vers `1.7.30`.
1.31-2025.12.15	1.31.13	1.7.28	1.2.1	
1.31-2025.11.14	1.31.13	1.7.28	1.2.1	
1.31-2025.10.18	1.31.13	1.7.28	1.2.1	
1.31-2025.09.13	1.31.12	1.7.28	1.2.1	Le plug-in GMSA modifié enregistre les événements Windows
1.31-2025.08.18	1.31.11	1.7.27	1.2.1	

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
1.31-2025.07.16	1.31.9	1.7.27	1.2.1	
1.31-2025.06.13	1.31.9	1.7.27	1.2.1	Mise à niveau `containerd` vers `1.7.27`.
1.31-2025.05.17	1.31.9	1.7.20	1.2.1	
1.31-2025.04.14	1.31.5	1.7.20	1.1.3	
1.31-2025.03.14	1.31.5	1.7.20	1.1.3	
1.31-2025.02.15	1.31.5	1.7.20	1.1.3	
1.31-2025.01.15	1.31.4	1.7.20	1.1.3	
1.31-2025.01.01	1.31.4	1.7.20	1.1.3	Comprend des correctifs pour `CVE-2024-9042` .
1.31-2024.12.13	1.31.3	1.7.20	1.1.3	
1.31-2024.11.12	1.31.1	1.7.20	1.1.3	
1.31-2024.10.08	1.31.1	1.7.20	1.1.3	

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
1.31-2024.10.01	1.31.1	1.7.20	1.1.3	
1.31-2024.09.10	1.31.0	1.7.20	1.1.3	

Version 1.30 de Kubernetes

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
1.30-2026.01.22	1.30.14	1.7.30	1.2.1	Mise à niveau `containerd` vers `1.7.30`.
1.30-2025.12.15	1.30.14	1.7.28	1.2.1	
1.30-2025.11.21	1.30.14	1.7.28	1.2.1	
1.30-2025.10.18	1.30.14	1.7.28	1.2.1	
1.30-2025.09.13	1.30.14	1.7.28	1.2.1	Le plug-in GMSA modifié enregistre les événements Windows
1.30-2025.08.18	1.30.14	1.7.27	1.2.1	
1.30-2025.07.16	1.30.13	1.7.27	1.2.1	

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
1.30-2025.06.13	1.30.13	1.7.27	1.2.1	Mise à niveau `containerd` vers `1.7.27`.
1.30-2025.05.17	1.30.13	1.7.20	1.2.1	
1.30-2025.04.14	1.30.9	1.7.20	1.1.3	
1.30-2025.03.14	1.30.9	1.7.20	1.1.3	Mise à niveau `containerd` vers `1.7.20`.
1.30-2025-02-15	1.30.9	1.7.14	1.1.3	
1.30-2025.01.15	1.30.8	1.7.14	1.1.3	
1.30-2025.01.01	1.30.8	1.7.14	1.1.3	Comprend des correctifs pour `CVE-2024-9042` .
1.30-2024.12.11	1.30.7	1.7.14	1.1.3	
1.30-2024.11.12	1.30.4	1.7.14	1.1.3	
1.30-2024.10.08	1.30.4	1.7.14	1.1.3	
1.30-2024.09.10	1.30.2	1.7.14	1.1.3	

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
1.30-2024.08.13	1.30.2	1.7.14	1.1.3	
1.30-2024.07.10	1.30.2	1.7.14	1.1.2	Comprend des correctifs pour CVE-2024-5321 .
1.30-2024.06.17	1.30.0	1.7.14	1.1.2	Mise à niveau containerd vers 1.7.14.
1.30-2024.05.15	1.30.0	1.6.28	1.1.2	

Version 1.29 de Kubernetes

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
1.29-2026.01.22	1.29.15	1.7.30	1.2.1	Mise à niveau containerd vers 1.7.30.
1.29-2025.12.15	1.29.15	1.7.28	1.2.1	
1.29-2025.11.14	1.29.15	1.7.28	1.2.1	
1.29-2025.10.18	1.29.15	1.7.28	1.2.1	
1.29-2025.09.13	1.29.15	1.7.28	1.2.1	Le plug-in GMSA modifié

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
				enregistre les événements Windows
1.29-2025.08.18	1.29.15	1.7.27	1.2.1	
1.29-2025.07.16	1.29.15	1.7.27	1.2.1	
1.29-2025.06.13	1.29.15	1.7.27	1.2.1	Mise à niveau `containerd` vers `1.7.27`.
1.29-2025.05.17	1.29.15	1.7.20	1.2.1	
1.29-2025.04.14	1.29.13	1.7.20	1.1.3	
1.29-2025.03.14	1.29.13	1.7.20	1.1.3	Mise à niveau `containerd` vers `1.7.20`.
1.29-2025.02.15	1.29.13	1.7.14	1.1.3	
1.29-2025.01.15	1.29.12	1.7.14	1.1.3	
1.29-2025.01.01	1.29.12	1.7.14	1.1.3	Comprend des correctifs pour `CVE-2024-9042` .

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
1.29-2024.12.11	1.29.10	1.7.14	1.1.3	
1.29-2024.11.12	1.29.8	1.7.14	1.1.3	
1.29-2024.10.08	1.29.8	1.7.14	1.1.3	
1.29-2024.09.10	1.29.6	1.7.14	1.1.3	
1.29-2024.08.13	1.29.6	1.7.14	1.1.3	
1.29-2024.07.10	1.29.6	1.7.11	1.1.2	Comprend des correctifs pour CVE-2024-5321 .
1.29-2024.06.17	1.29.3	1.7.11	1.1.2	
1.29-2024.05.15	1.29.3	1.7.11	1.1.2	Mise à niveau containerd vers 1.7.11. Mise à niveau kubelet vers 1.29.3.

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
1.29-2024.04.09	1.29.0	1.6.28	1.1.2	Mise à niveau `containerd` vers `1.6.28`. CNI reconstruit et `csi-proxy` utilise `golang` `1.22.1`.
1.29-2024.03.13	1.29.0	1.6.25	1.1.2	
1.29-2024.02.13	1.29.0	1.6.25	1.1.2	
1.29-2024.02.06	1.29.0	1.6.25	1.1.2	Correction d'un bug qui entraînait la suppression incorrecte de l'image de pause par le processus de récupération de mémoire `kubelet`.
1.29-2024.01.09	1.29.0	1.6.18	1.1.2	

Kubernetes version 1.28

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
1.28-2025.11.14	1.28.15	1.7.28	1.2.1	
1.28-2025.10.18	1.28.15	1.7.28	1.2.1	
1.28-2025.09.13	1.28.15	1.7.28	1.2.1	Le plug-in GMSA modifié enregistre les événements Windows
1.28-2025.08.18	1.28.15	1.7.27	1.2.1	
1.28-2025.07.16	1.28.15	1.7.27	1.2.1	
1.28-2025.06.13	1.28.15	1.7.20	1.2.1	Mise à niveau `containerd` vers `1.7.27`.
1.28-2025.05.17	1.28.15	1.7.20	1.2.1	
1.28-2025.04.14	1.28.15	1.7.20	1.1.3	
1.28-2025.03.14	1.28.15	1.7.20	1.1.3	Mise à niveau `containerd` vers `1.7.20`.
1.28-2025.02.15	1.28.15	1.7.14	1.1.3	

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
1.28-2025-01-15	1.28.15	1.7.14	1.1.3	
1.28-2025-01-01	1.28.15	1.7.14	1.1.3	Comprend des correctifs pour CVE-2024-9042 .
1.28-2024.12.11	1.28.15	1.7.14	1.1.3	
1.28-2024.11.12	1.28.13	1.7.14	1.1.3	
1.28-2024.10.08	1.28.13	1.7.14	1.1.3	
1.28-2024.09.10	1.28.11	1.7.14	1.1.3	
1.28-2024.08.13	1.28.11	1.7.14	1.1.3	
1.28-2024.07.10	1.28.11	1.7.11	1.1.2	Comprend des correctifs pour CVE-2024-5321 .
1.28-2024.06.17	1.28.8	1.7.11	1.1.2	Mise à niveau `containerd` vers `1.7.11`.

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
1.28-2024.05.14	1.28.8	1.6.28	1.1.2	Mise à niveau `containerd` vers `1.6.28`. Mise à niveau `kubelet` vers `1.28.8`.
1.28-2024.04.09	1.28.5	1.6.25	1.1.2	Mise à niveau `containerd` vers `1.6.25`. CNI reconstruit et `csi-proxy` utilise `golang 1.22.1`.
1.28-2024.03.13	1.28.5	1.6.18	1.1.2	
1.28-2024.02.13	1.28.5	1.6.18	1.1.2	
1.28-2024.01.09	1.28.5	1.6.18	1.1.2	
1.28-2023.12.12	1.28.3	1.6.18	1.1.2	
1.28-2023.11.14	1.28.3	1.6.18	1.1.2	Comprend des correctifs pour `CVE-2023-5528` .

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
1.28-2023.10.19	1.28.2	1.6.18	1.1.2	Mise à niveau containerd vers 1.6.18. Ajout de nouvelles variables d'environnement pour le script d'amorçage (SERVICE_IPV4_CIDR et EXCLUDED_SNAT_CIDRS).
1.28-2023-09.27	1.28.2	1.6.6	1.1.2	Correction d'une recommandation de sécurité dans kubelet.
1.28-2023.09.12	1.28.1	1.6.6	1.1.2	

AMI Windows Server 2019 Full optimisée pour Amazon EKS

Les tableaux suivants répertorient les versions actuelles et précédentes de l'AMI complète Windows Server 2019 optimisée pour Amazon EKS.

Example

Version 1.35 de Kubernetes

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
`1.35-2026-01-22`	`1.35.0`	`2.1.6`	`1.2.1`	

Version 1.34 de Kubernetes

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
`1.34-2026.01.22`	`1.34.2`	`2.1.6`	`1.2.1`	Mise à niveau `containerd` vers `2.1.6`.
`1.34-2025.12.15`	`1.34.2`	`2.1.4`	`1.2.1`	
`1.34-2025.11.14`	`1.34.1`	`2.1.4`	`1.2.1`	
`1.34-2025.10.18`	`1.34.1`	`2.1.4`	`1.2.1`	
`1.34-2025.09.13`	`1.34.0`	`2.1.4`	`1.2.1`	

Kubernetes version 1.33

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
1.33-2026.01.22	1.33.5	1.7.30	1.2.1	Mise à niveau `containerd` vers `1.7.30`.
1.33-2025.12.15	1.33.5	1.7.28	1.2.1	
1.33-2025.11.14	1.33.5	1.7.28	1.2.1	
1.33-2025.10.18	1.33.5	1.7.28	1.2.1	
1.33-2025.09.13	1.33.4	1.7.28	1.2.1	Le plug-in GMSA modifié enregistre les événements Windows
1.33-2025.08.18	1.33.3	1.7.27	1.2.1	
1.33-2025.07.16	1.33.1	1.7.27	1.2.1	
1.33-2025.06.13	1.33.1	1.7.27	1.2.1	
1.33-2025.05.17	1.33.1	1.7.27	1.2.1	

Kubernetes version 1.32

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
1.32-2026.01.22	1.32.9	1.7.30	1.2.1	Mise à niveau containerd vers 1.7.30.
1.32-2025.12.15	1.32.9	1.7.28	1.2.1	
1.32-2025.11.14	1.32.9	1.7.28	1.2.1	
1.32-2025.10.18	1.32.9	1.7.28	1.2.1	
1.32-2025.09.13	1.32.8	1.7.28	1.2.1	Le plug-in GMSA modifié enregistre les événements Windows
1.32-2025.08.18	1.32.7	1.7.27	1.2.1	
1.32-2025.07.16	1.32.5	1.7.27	1.2.1	
1.32-2025.06.13	1.32.5	1.7.27	1.2.1	Mise à niveau containerd vers 1.7.27.
1.32-2025.05.17	1.32.5	1.7.20	1.2.1	
1.32-2025.04.14	1.32.1	1.7.20	1.1.3	

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
1.32-2025.03.14	1.32.1	1.7.20	1.1.3	
1.32-2025.02.18	1.32.1	1.7.20	1.1.3	
1.32-2025.01.15	1.32.0	1.7.20	1.1.3	

Kubernetes version 1.31

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
1.31-2026.01.22	1.31.13	1.7.30	1.2.1	Mise à niveau `containerd` vers `1.7.30`.
1.31-2025.12.15	1.31.13	1.7.28	1.2.1	
1.31-2025.11.14	1.31.13	1.7.28	1.2.1	
1.31-2025.10.18	1.31.13	1.7.28	1.2.1	
1.31-2025.09.13	1.31.12	1.7.28	1.2.1	Le plug-in GMSA modifié enregistre les événements Windows
1.31-2025.08.18	1.31.11	1.7.27	1.2.1	

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
1.31-2025.07.16	1.31.9	1.7.27	1.2.1	
1.31-2025.06.13	1.31.9	1.7.27	1.2.1	Mise à niveau `containerd` vers `1.7.27`.
1.31-2025.05.17	1.31.9	1.7.20	1.2.1	
1.31-2025.04.14	1.31.5	1.7.20	1.1.3	
1.31-2025.03.14	1.31.5	1.7.20	1.1.3	
1.31-2025.02.15	1.31.5	1.7.20	1.1.3	
1.31-2025.01.15	1.31.4	1.7.20	1.1.3	
1.31-2025.01.01	1.31.4	1.7.20	1.1.3	Comprend des correctifs pour `CVE-2024-9042` .
1.31-2024.12.13	1.31.3	1.7.20	1.1.3	
1.31-2024.11.12	1.31.1	1.7.20	1.1.3	
1.31-2024.10.08	1.31.1	1.7.20	1.1.3	

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
1.31-2024.10.01	1.31.1	1.7.20	1.1.3	
1.31-2024.09.10	1.31.0	1.7.20	1.1.3	

Version 1.30 de Kubernetes

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
1.30-2026.01.22	1.30.14	1.7.30	1.2.1	Mise à niveau `containerd` vers `1.7.30`.
1.30-2025.12.15	1.30.14	1.7.28	1.2.1	
1.30-2025.11.21	1.30.14	1.7.28	1.2.1	
1.30-2025.10.18	1.30.14	1.7.28	1.2.1	
1.30-2025.09.13	1.30.14	1.7.28	1.2.1	Le plug-in GMSA modifié enregistre les événements Windows
1.30-2025.08.18	1.30.14	1.7.27	1.2.1	
1.30-2025.07.16	1.30.13	1.7.27	1.2.1	

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
1.30-2025.06.13	1.30.13	1.7.27	1.2.1	Mise à niveau `containerd` vers `1.7.27`.
1.30-2025.05.17	1.30.13	1.7.20	1.2.1	
1.30-2025.04.14	1.30.9	1.7.20	1.1.3	
1.30-2025.03.14	1.30.9	1.7.20	1.1.3	Mise à niveau `containerd` vers `1.7.20`.
1.30-2025.02.15	1.30.9	1.7.14	1.1.3	
1.30-2025.01.15	1.30.8	1.7.14	1.1.3	
1.30-2025.01.01	1.30.8	1.7.14	1.1.3	Comprend des correctifs pour `CVE-2024-9042` .
1.30-2024.12.11	1.30.7	1.7.14	1.1.3	
1.30-2024.11.12	1.30.4	1.7.14	1.1.3	
1.30-2024.10.08	1.30.4	1.7.14	1.1.3	
1.30-2024.09.10	1.30.2	1.7.14	1.1.3	

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
1.30-2024.08.13	1.30.2	1.7.14	1.1.3	
1.30-2024.07.10	1.30.2	1.7.14	1.1.2	Comprend des correctifs pour CVE-2024-5321 .
1.30-2024.06.17	1.30.0	1.7.14	1.1.2	Mise à niveau containerd vers 1.7.14.
1.30-2024.05.15	1.30.0	1.6.28	1.1.2	

Version 1.29 de Kubernetes

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
1.29-2026.01.22	1.29.15	1.7.30	1.2.1	Mise à niveau containerd vers 1.7.30.
1.29-2025.12.15	1.29.15	1.7.28	1.2.1	
1.29-2025.11.14	1.29.15	1.7.28	1.2.1	
1.29-2025.10.18	1.29.15	1.7.28	1.2.1	
1.29-2025.09.13	1.29.15	1.7.28	1.2.1	Le plug-in GMSA modifié

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
				enregistre les événements Windows
1.29-2025.08.18	1.29.15	1.7.27	1.2.1	
1.29-2025.07.16	1.29.15	1.7.27	1.2.1	
1.29-2025.06.13	1.29.15	1.7.27	1.2.1	Mise à niveau `containerd` vers `1.7.27`.
1.29-2025.05.17	1.29.15	1.7.20	1.2.1	
1.29-2025.04.14	1.29.13	1.7.20	1.1.3	
1.29-2025.03.14	1.29.13	1.7.20	1.1.3	Mise à niveau `containerd` vers `1.7.20`.
1.29-2025.02.15	1.29.13	1.7.14	1.1.3	
1.29-2025.01.15	1.29.12	1.7.14	1.1.3	
1.29-2025.01.01	1.29.12	1.7.14	1.1.3	Comprend des correctifs pour `CVE-2024-9042` .

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
1.29-2024.12.11	1.29.10	1.7.14	1.1.3	
1.29-2024.11.12	1.29.8	1.7.14	1.1.3	
1.29-2024.10.08	1.29.8	1.7.14	1.1.3	
1.29-2024.09.10	1.29.6	1.7.14	1.1.3	
1.29-2024.08.13	1.29.6	1.7.14	1.1.3	
1.29-2024.07.10	1.29.6	1.7.11	1.1.2	Comprend des correctifs pour CVE-2024-5321 .
1.29-2024.06.17	1.29.3	1.7.11	1.1.2	
1.29-2024.05.15	1.29.3	1.7.11	1.1.2	Mise à niveau containerd vers 1.7.11. Mise à niveau kubelet vers 1.29.3.

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
1.29-2024.04.09	1.29.0	1.6.28	1.1.2	Mise à niveau `containerd` vers `1.6.28`. CNI reconstruit et `csi-proxy` utilise `golang` `1.22.1`.
1.29-2024.03.13	1.29.0	1.6.25	1.1.2	
1.29-2024.02.13	1.29.0	1.6.25	1.1.2	
1.29-2024.02.06	1.29.0	1.6.25	1.1.2	Correction d'un bug qui entraînait la suppression incorrecte de l'image de pause par le processus de récupération de mémoire `kubelet`.
1.29-2024.01.09	1.29.0	1.6.18	1.1.2	

Kubernetes version 1.28

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
1.28-2025.11.14	1.28.15	1.7.28	1.2.1	
1.28-2025.10.18	1.28.15	1.7.28	1.2.1	
1.28-2025.09.13	1.28.15	1.7.28	1.2.1	Le plug-in GMSA modifié enregistre les événements Windows
1.28-2025.08.18	1.28.15	1.7.27	1.2.1	
1.28-2025.07.16	1.28.15	1.7.27	1.2.1	
1.28-2025.06.13	1.28.15	1.7.27	1.2.1	Mise à niveau `containerd` vers `1.7.27`.
1.28-2025.05.17	1.28.15	1.7.20	1.2.1	
1.28-2025.04.14	1.28.15	1.7.20	1.1.3	
1.28-2025.03.14	1.28.15	1.7.20	1.1.3	Mise à niveau `containerd` vers `1.7.20`.
1.28-2025.02.15	1.28.15	1.7.14	1.1.3	

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
1.28-2025-01-15	1.28.15	1.7.14	1.1.3	
1.28-2025-01-01	1.28.15	1.7.14	1.1.3	Comprend des correctifs pour CVE-2024-9042 .
1.28-2024.12.11	1.28.15	1.7.14	1.1.3	
1.28-2024.11.12	1.28.13	1.7.14	1.1.3	
1.28-2024.10.08	1.28.13	1.7.14	1.1.3	
1.28-2024.09.10	1.28.11	1.7.14	1.1.3	
1.28-2024.08.13	1.28.11	1.7.14	1.1.3	
1.28-2024.07.10	1.28.11	1.7.11	1.1.2	Comprend des correctifs pour CVE-2024-5321 .
1.28-2024.06.17	1.28.8	1.7.11	1.1.2	Mise à niveau `containerd` vers `1.7.11`.

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
1.28-2024.05.14	1.28.8	1.6.28	1.1.2	Mise à niveau `containerd` vers `1.6.28`. Mise à niveau `kubelet` vers `1.28.8`.
1.28-2024.04.09	1.28.5	1.6.25	1.1.2	Mise à niveau `containerd` vers `1.6.25`. CNI reconstruit et `csi-proxy` utilise `golang 1.22.1`.
1.28-2024.03.13	1.28.5	1.6.18	1.1.2	
1.28-2024.02.13	1.28.5	1.6.18	1.1.2	
1.28-2024.01.09	1.28.5	1.6.18	1.1.2	
1.28-2023.12.12	1.28.3	1.6.18	1.1.2	
1.28-2023.11.14	1.28.3	1.6.18	1.1.2	Comprend des correctifs pour `CVE-2023-5528` .

Version d'AMI	version kubelet	version de containerd	version csi-proxy	Notes de mise à jour
1.28-2023.10.19	1.28.2	1.6.18	1.1.2	Mise à niveau containerd vers 1.6.18. Ajout de nouvelles variables d'environnement pour le script d'amorçage (SERVICE_IPV4_CIDR et EXCLUDED_SNAT_CIDRS).
1.28-2023-09.27	1.28.2	1.6.6	1.1.2	Correction d'une recommandation de sécurité dans kubelet.
1.28-2023.09.12	1.28.1	1.6.6	1.1.2	

Récupérez l'AMI Microsoft Windows recommandée IDs

Lors du déploiement de nœuds, vous pouvez spécifier un identifiant pour une image Amazon Machine Image (AMI) optimisée pour Amazon EKS pré-créée. Pour récupérer un ID d'AMI correspondant à la configuration souhaitée, interrogez l'API du magasin de paramètres AWS Systems Manager. L'utilisation de cette API élimine le besoin de rechercher manuellement l'AMI optimisée pour Amazon EKS IDs. Pour de plus amples informations, veuillez consulter GetParameter. Le principal IAM que vous utilisez doit disposer de l'autorisation IAM `ssm:GetParameter` pour récupérer les métadonnées de l'AMI optimisée pour Amazon EKS.

Vous pouvez récupérer l'ID d'image de la dernière AMI Windows optimisée Amazon EKS recommandée à l'aide de la commande suivante, qui utilise le sous-paramètre `image_id`. Si nécessaire, apportez les modifications suivantes à la commande, puis exécutez la commande modifiée :

- *`release`*Remplacez-le par l'une des options suivantes.
  - À utiliser *`2025`* pour Windows Server 2025.
  - À utiliser *`2022`* pour Windows Server 2022.
  - À utiliser *`2019`* pour Windows Server 2019.
- *`installation-option`*Remplacez-le par l'une des options suivantes. Pour plus d'informations, consultez Qu'est-ce que l'option d'installation Server Core dans Windows Server.
  - *`Core`*À utiliser pour une installation minimale avec une surface d'attaque plus petite.
  - *`Full`*À utiliser pour inclure l'expérience de bureau Windows.
- *`kubernetes-version`*Remplacez-le par une version de plate-forme prise en charge.
- *`region-code`*Remplacez-le par une AWS région prise en charge par Amazon EKS pour laquelle vous souhaitez obtenir l'ID AMI.

```
aws ssm get-parameter --name /aws/service/ami-windows-latest/Windows_Server-release-
English-installation-option-EKS_Optimized-kubernetes-version/image_id \
    --region region-code --query "Parameter.Value" --output text
```

Voici un exemple de commande après remplacement des espaces réservés.

```
aws ssm get-parameter --name /aws/service/ami-windows-latest/Windows_Server-2022-
English-Core-EKS_Optimized-k8s-n-2/image_id \
    --region us-west-2 --query "Parameter.Value" --output text
```

L'exemple qui suit illustre un résultat.

```
ami-1234567890abcdef0
```

Créer une AMI Windows personnalisée avec Image Builder

Vous pouvez utiliser EC2 Image Builder pour créer des fenêtres personnalisées optimisées pour Amazon EKS AMIs avec l'une des options suivantes :

- Utilisation d'une AMI optimisée pour Amazon EKS comme base
- Utilisation du composant de compilation géré par Amazon

Vous devez créer votre propre recette Image Builder avec les deux méthodes. Pour plus d'informations, veuillez consulter la rubrique Créer une nouvelle version d'une recette d'image dans le Guide de l'utilisateur Image Builder.

> **Important**
>
> Les composants gérés par Amazon suivants pour `eks` incluent des correctifs pour `CVE-2024-5321`.
>
> - `1.28.2` et ultérieures
> - `1.29.2` et ultérieures
> - `1.30.1` et ultérieures
> - Toutes les versions pour Kubernetes 1.31 et versions supérieures

Utilisation d'une AMI Windows optimisée pour Amazon EKS comme base

Cette option est la méthode recommandée pour créer votre Windows personnalisé AMIs. Les fenêtres optimisées pour Amazon EKS que AMIs nous proposons sont mises à jour plus fréquemment que le composant de compilation géré par Amazon.

1. Lancez une nouvelle recette Image Builder.
   a. Ouvrez la console EC2 Image Builder à l'adresse https://console.aws.amazon.com / imagebuilder.
   b. Dans le panneau de navigation de gauche, choisissez Recettes d'image.
   c. Choisissez Créer une recette d'image.
2. Dans la section Détails de la recette, saisissez un Nom et une Version.
3. Spécifiez l'ID de l'AMI Windows optimisée pour Amazon EKS dans la section Image de base.
   a. Choisissez Saisir un ID d'AMI personnalisé.
   b. Récupérez l'ID AMI correspondant à la version du système d'exploitation Windows dont vous avez besoin. Pour de plus amples informations, veuillez consulter the section called "Recevez les dernières nouvelles IDs".

c. Saisissez l'ID d'AMI personnalisé. Si l'ID d'AMI est introuvable, assurez-vous que la AWS région correspondant à l'ID d'AMI correspond à la AWS région affichée dans le coin supérieur droit de votre console.

4. (Facultatif) Pour obtenir les dernières mises à jour de sécurité, ajoutez le composant `update-windows` dans la section Composants de génération.
   a. Dans la liste déroulante située à droite du champ de recherche Rechercher des composants par nom, sélectionnez Géré par Amazon.
   b. Dans le champ de recherche Rechercher des composants par nom, saisissez `update-windows`.
   c. Cochez la case du résultat de recherche **`update-windows`** . Ce composant comprend les derniers correctifs Windows pour le système d'exploitation.
5. Complétez les entrées de recette d'image restantes avec les configurations requises. Pour plus d'informations, veuillez consulter la rubrique Créer une nouvelle version d'une recette d'image (console) dans le Guide de l'utilisateur Image Builder.
6. Choisissez Créer une recette.
7. Utilisez la nouvelle recette d'image dans un pipeline d'images nouveau ou existant. Une fois que votre pipeline d'images s'exécute correctement, votre AMI personnalisée sera répertoriée en tant qu'image de sortie et sera prête à être utilisée. Pour plus d'informations, voir Création d'un pipeline d'images à l'aide de l'assistant de console EC2 Image Builder.

Utilisation du composant de génération géré par Amazon

Lorsque l'utilisation d'une AMI Windows optimisée pour Amazon EKS comme base n'est pas viable, vous pouvez utiliser à la place le composant de build géré par Amazon. Cette option peut être en retard par rapport aux versions Kubernetes les plus récentes prises en charge.

1. Lancez une nouvelle recette Image Builder.
   a. Ouvrez la console EC2 Image Builder à l'adresse https://console.aws.amazon.com / imagebuilder.
   b. Dans le panneau de navigation de gauche, choisissez Recettes d'image.
   c. Choisissez Créer une recette d'image.
2. Dans la section Détails de la recette, saisissez un Nom et une Version.
3. Déterminez l'option que vous utiliserez pour créer votre AMI personnalisée dans la section Image de base :

- Sélectionnez les images gérées : choisissez Windows pour votre Système d'exploitation (OS) de l'image. Choisissez ensuite l'une des options suivantes pour Origine de l'image.
  - Démarrage rapide (géré par Amazon) : dans le menu déroulant Nom de l'image, choisissez une version Windows Server prise en charge par Amazon EKS. Pour de plus amples informations, veuillez consulter the section called "Windows".
  - Mes images : pour Nom de l'image, sélectionnez l'ARN de votre propre image avec votre propre licence. Les composants Amazon EKS ne peuvent pas déjà être installés sur l'image que vous fournissez.
- Saisir l'ID d'AMI personnalisée : pour l'ID D'AMI, saisissez l'ID de votre AMI avec votre propre licence. Les composants Amazon EKS ne peuvent pas déjà être installés sur l'image que vous fournissez.

4. Dans la section Composants de génération – Windows, procédez comme suit :
   a. Dans la liste déroulante située à droite du champ de recherche Rechercher des composants par nom, sélectionnez Géré par Amazon.
   b. Dans le champ de recherche Rechercher des composants par nom, saisissez `eks`.
   c. Cochez la case du résultat de recherche **`eks-optimized-ami-windows`** , même si le résultat renvoyé peut ne pas être la version que vous voulez.
   d. Dans le champ de recherche Rechercher des composants par nom, saisissez `update-windows`.
   e. Cochez la case du résultat de recherche update-windows. Ce composant comprend les derniers correctifs Windows pour le système d'exploitation.
5. Dans la section Composants sélectionnés, procédez comme suit :
   a. Choisissez Options de gestion des versions pour **`eks-optimized-ami-windows`** .
   b. Choisissez Spécifier la version du composant.
   c. Dans le champ Version du composant, saisissez*`version.x`*, en *`version`* remplaçant par une version Kubernetes prise en charge. La saisie *`x`* d'une partie du numéro de version indique que vous devez utiliser la dernière version du composant qui correspond également à la partie de la version que vous avez explicitement définie. Faites attention à la sortie de la console, car elle vous indiquera si la version souhaitée est disponible en tant que composant géré. Gardez à l'esprit que les versions les plus récentes de Kubernetes peuvent ne pas être disponibles pour le composant de compilation. Pour plus d'informations sur les versions disponibles, consultez the section called "Récupération d'informations sur les versions des composants `eks-optimized-ami-windows`".

6. Complétez les entrées de recette d'image restantes avec les configurations requises. Pour plus d'informations, veuillez consulter la rubrique Créer une nouvelle version d'une recette d'image (console) dans le Guide de l'utilisateur Image Builder.
7. Choisissez Créer une recette.
8. Utilisez la nouvelle recette d'image dans un pipeline d'images nouveau ou existant. Une fois que votre pipeline d'images s'exécute correctement, votre AMI personnalisée sera répertoriée en tant qu'image de sortie et sera prête à être utilisée. Pour plus d'informations, voir Création d'un pipeline d'images à l'aide de l'assistant de console EC2 Image Builder.

Récupération d'informations sur les versions des composants **`eks-optimized-ami-windows`**

Vous pouvez récupérer des informations spécifiques concernant ce qui est installé avec chaque composant. Par exemple, vous pouvez vérifier quelle version de `kubelet` est installée. Les composants sont soumis à des tests fonctionnels sur les versions des systèmes d'exploitation Windows prises en charge par Amazon EKS. Pour de plus amples informations, veuillez consulter the section called "Calendrier des versions". Toutes les autres versions du système d'exploitation Windows qui ne sont pas répertoriées comme prises en charge ou qui ont atteint la fin de leur prise en charge peuvent ne pas être compatibles avec le composant.

1. Ouvrez la console EC2 Image Builder à l'adresse https://console.aws.amazon.com /imagebuilder.
2. Dans le panneau de navigation de gauche, sélectionnez Composants.
3. Dans la liste déroulante située à droite du champ de recherche Rechercher des composants par nom, modifiez M'appartenant en Démarrage rapide (géré par Amazon).
4. Dans la case Rechercher des composants par nom, saisissez `eks`.
5. (Facultatif) Si vous utilisez une version récente, triez la colonne Version par ordre décroissant en la choisissant deux fois.
6. Choisissez le lien **`eks-optimized-ami-windows`** avec la version souhaitée.

La Description de la page qui s'affiche contient les informations spécifiques.

Activation de la réparation automatique des nœuds et examen des problèmes d'état de ces derniers

L'état d'un nœud fait référence à son état opérationnel et à sa capacité à exécuter efficacement des charges de travail. Un nœud en bon état maintient la connectivité attendue, dispose de ressources

suffisantes et peut exécuter avec succès des pods sans interruption. Pour plus d'informations sur vos nœuds, consultez the section called "Afficher l'état de santé des nœuds" et the section called "Extraction des journaux de nœuds".

Afin de vous aider à maintenir vos nœuds en bon état de fonctionnement, Amazon EKS propose l'agent de surveillance des nœuds et la réparation automatique des nœuds.

> **Important**
>
> L'agent de surveillance des nœuds et la réparation automatique des nœuds ne sont disponibles que sous Linux. Ces fonctionnalités ne sont pas disponibles sous Windows.

Agent de surveillance des nœuds

L'agent de surveillance des nœuds journalise automatiquement les journaux des nœuds afin de détecter certains problèmes de leur état. Il analyse les journaux des nœuds afin de détecter les défaillances et affiche diverses informations sur l'état des composants master. Une `NodeCondition` dédiée est appliquée aux composants master pour chaque catégorie de problèmes détectés, tels que les problèmes de stockage et de réseau. Les descriptions des problèmes d'état détectés sont disponibles dans le tableau de bord d'observabilité. Pour de plus amples informations, veuillez consulter the section called "Problèmes d'état du nœud".

L'agent de surveillance des nœuds est inclus en tant que fonctionnalité pour tous les clusters du mode automatique Amazon EKS. Pour les autres types de clusters, vous pouvez ajouter l'agent de surveillance en tant que module complémentaire Amazon EKS. Pour de plus amples informations, veuillez consulter the section called "Créer un module complémentaire".

Réparation automatique de nœuds

La réparation automatique des nœuds est une fonctionnalité supplémentaire qui surveille en permanence l'état des nœuds, réagit automatiquement aux problèmes détectés et remplace les nœuds lorsque cela est possible. Cela contribue à la disponibilité globale du cluster avec un minimum d'intervention manuelle. Si une surveillance de l'état échoue, le nœud est automatiquement isolé afin qu'aucun nouveau pod ne soit planifié sur ce nœud.

En soi, la réparation automatique des nœuds peut réagir à la condition `Ready` de `kubelet` et à tous les objets de nœuds qui sont supprimés manuellement. Associée à l'agent de surveillance des nœuds, la réparation automatique des nœuds peut réagir à davantage de conditions qui

ne seraient pas détectées autrement. Ces conditions supplémentaires incluent `KernelReady`, `NetworkingReady` et `StorageReady`.

Cette restauration automatique des nœuds résout automatiquement les problèmes intermittents liés aux nœuds, notamment les échecs de connexion au cluster, des kubelets qui ne répondent pas et l'augmentation des erreurs de l'accélérateur (appareil). La fiabilité accrue permet de réduire la durée d'indisponibilité des applications et d'améliorer le fonctionnement des clusters. Par défaut, la réparation automatique des nœuds ne répare pas automatiquement les nœuds pour certaines conditions telles que`DiskPressure`, `MemoryPressurePIDPressure`, et les erreurs de l'outil de diagnostic ou de surveillance DCGM (NVIDIA Data Center GPU Manager). Ces conditions indiquent souvent des problèmes liés au comportement des applications, à la configuration de la charge de travail ou aux limites de ressources plutôt que des défaillances au niveau des nœuds, ce qui complique la détermination d'une action de réparation par défaut appropriée. Cependant, vous pouvez personnaliser ce comportement en activant `nodeRepairConfigOverrides` les actions de réparation automatiques pour ces conditions en fonction de votre cas d'utilisation. Amazon EKS attend 10 min avant d'agir sur les `AcceleratedHardwareReady` `NodeConditions` et 30 min pour toutes les autres conditions.

Les groupes de nœuds gérés désactivent également automatiquement les réparations de nœuds pour des raisons de sécurité dans deux cas de figure. Toutes les opérations de réparation déjà en cours se poursuivent dans les deux cas.

- Si un changement de zone pour votre cluster a été déclenché par le contrôleur de récupération d'application (ARC), toutes les opérations de réparation ultérieures sont interrompues.
- Si votre groupe de nœuds comporte plus de cinq nœuds et que plus de 20 % des nœuds de votre groupe de nœuds sont dans un état non sain, les opérations de réparation sont interrompues.

Vous pouvez activer la réparation automatique des nœuds lors de la création ou de la modification d'un groupe de nœuds gérés.

- Lorsque vous utilisez la console Amazon EKS, cochez la case Activer la réparation automatique des nœuds pour le groupe de nœuds gérés. Pour de plus amples informations, veuillez consulter the section called "Créer".
- Lorsque vous utilisez la AWS CLI, ajoutez la `eks update-nodegroup-config`commande `--node-repair-config enabled=true` to the `eks create nodegroup`or.
- Pour un exemple d'utilisation `eksctl` `ClusterConfig` d'un groupe de nœuds géré avec réparation automatique des nœuds, consultez 44-node-repair.yaml on. GitHub

Amazon EKS offre un contrôle plus granulaire sur le comportement de réparation automatique des nœuds grâce aux éléments suivants :

- `maxUnhealthyNodeThresholdCount` et `maxUnhealthyNodeThresholdPercentage`
  - Ces champs vous permettent de spécifier un seuil en nombre ou en pourcentage de nœuds défectueux, au-delà duquel les actions de réparation automatique des nœuds s'arrêtent. Cela permet de mieux contrôler la « portée » des réparations automatiques des nœuds.
  - Vous pouvez définir soit le nombre absolu, soit le pourcentage, mais pas les deux.
- `maxParallelNodesRepairedCount` et `maxParallelNodesRepairedPercentage`
  - Ces champs vous permettent de spécifier le nombre maximal de nœuds pouvant être réparés avec simultanéité ou en parallèle, exprimé en nombre ou en pourcentage de tous les nœuds défectueux. Cela vous offre un contrôle plus précis sur le rythme des remplacements de nœuds.
  - Comme pour le seuil de nœuds défectueux, vous pouvez définir soit le nombre absolu, soit le pourcentage, mais pas les deux.
- `nodeRepairConfigOverrides`
  - Il s'agit d'une structure complexe qui vous permet de définir des remplacements granulaires pour des actions de réparation spécifiques. Ces remplacements contrôlent l'action de réparation et le délai de réparation avant qu'un nœud ne soit considéré comme éligible à la réparation.
  - Les champs spécifiques de cette structure sont les suivants :
    - `nodeMonitoringCondition` : l'état non sain signalé par l'agent de surveillance des nœuds.
    - `nodeUnhealthyReason` : la raison pour laquelle l'agent de surveillance des nœuds a identifié le nœud comme non sain.
    - `minRepairWaitTimeMins` : le temps minimum (en minutes) pendant lequel l'état de réparation et la raison de la défaillance doivent persister avant que le nœud ne soit éligible à la réparation.
    - `repairAction` : l'action que le système de réparation doit effectuer lorsque les conditions ci-dessus sont remplies.
  - Si vous utilisez ce champ, vous devez spécifier tous les champs de la structure. Vous pouvez également fournir une liste de ces remplacements.
  - Les champs `nodeMonitoringCondition` et `nodeUnhealthyReason` sont des entrées de texte manuelles que vous définissez pour indiquer que vous voulez vous écarter du comportement par défaut du système.
  - Les champs `minRepairWaitTimeMins` et `repairAction` vous permettent de spécifier des écarts par rapport au comportement par défaut du système.

- L'exemple suivant montre comment remplacer le temps d'attente à 20 minutes avant qu'Amazon EKS ne redémarre un nœud présentant des problèmes`NvidiaXID13Error`. Par défaut, Amazon EKS attend 10 minutes avant de prendre des mesures de réparation sous certaines `AcceleratedHardwareReady` conditions.

```
aws eks update-nodegroup-config \
  --cluster-name my-cluster \
  --nodegroup-name my-nodegroup \
  --node-repair-config
 'enabled=true,nodeRepairConfigOverrides=[{nodeMonitoringCondition=AcceleratedHardwareReady
```

Problèmes d'état du nœud

Les tableaux suivants décrivent les problèmes d'état des nœuds que l'agent de surveillance des nœuds peut détecter. Il existe deux types de problèmes :

- Condition : problème terminal qui nécessite une action corrective, telle que le remplacement d'une instance ou un redémarrage. Lorsque la réparation automatique est activée, Amazon EKS effectue une action de réparation, soit en remplaçant le nœud, soit en le redémarrant. Pour de plus amples informations, veuillez consulter the section called "Conditions des nœuds".
- Événement : problème temporaire ou configuration sous-optimale du nœud. Aucune action de réparation automatique n'est effectuée. Pour de plus amples informations, veuillez consulter the section called "Événements des nœuds".

AcceleratedHardware problèmes de santé des nœuds

La condition de surveillance est `AcceleratedHardwareReady` pour les problèmes du tableau suivant qui ont une sévérité « Condition ».

Si la réparation automatique est activée, les actions de réparation répertoriées commencent 10 min après la détection du problème. Pour plus d'informations sur les erreurs XID, consultez Erreurs XID dans la Documentation sur le déploiement et la gestion des GPU NVIDIA. Pour plus d'informations sur les messages XID individuels, consultez Comprendre les messages XID dans la Documentation sur le déploiement et la gestion des GPU NVIDIA.

Name	Sévérité	Description	Action de réparation
DCGMDiagnosticDéfaillance	Condition	Un cas de test de la suite de tests de diagnostic actif DCGM a échoué.	Aucune
DCGMError	Condition	La connexion au processus hôte DCGM a été perdue ou n'a pas pu être établie.	Aucune
DCGMFieldErreur [Code]	Événement	Le DCGM a détecté une dégradation du GPU grâce à un identifiant de champ.	Aucune
DCGMHealthCode [Code]	Événement	Un bilan de santé du DCGM a échoué de manière non fatale.	Aucune
DCGMHealthCode [Code]	Condition	Un bilan de santé du DCGM a échoué de manière fatale.	Aucune
Neurone DMAError	Condition	Un moteur DMA a rencontré une erreur irrécupérable.	Remplacez
Erreur neuronale HBMUncorrectable	Condition	Une mémoire HBM a rencontré une erreur incorrigible et a produit des résultats incorrects.	Remplacez
Erreur neuronale NCUncorrectable	Condition	Une erreur mémoire incorrigible du	Remplacez

Name	Sévérité	Description	Action de réparation
		cœur Neuron a été détectée.	
Erreur neuronale SRAMUncorrectable	Condition	Une mémoire SRAM sur puce a rencontré une erreur de parité et a produit des résultats incorrects.	Remplacez
NvidiaDeviceCountMismatch	Événement	Le nombre de périphériques GPUs visibles via NVML ne correspond pas au nombre de périphériques NVIDIA présents sur le système de fichiers.	Aucune
NvidiaDoubleBitError	Condition	Le pilote GPU a généré une erreur double bit.	Remplacez
Nvidia NCCLError	Événement	Une erreur de segmentation s'est produite dans la bibliothèque NVIDIA Collective Communications (`libnccl`).	Aucune
NVLinkErreur Nvidia	Condition	NVLink des erreurs ont été signalées par le pilote du GPU.	Remplacez

Name	Sévérité	Description	Action de réparation
PCIeErreur Nvidia	Événement	PCIe des rediffusions ont été déclenchées pour remédier à des erreurs de transmiss ion.	Aucune
NvidiaPageRetireme nt	Event	Le pilote GPU a marqué une page mémoire pour mise hors service. Cela peut se produire si une seule erreur double bit ou deux erreurs simple bit sont détectées à la même adresse.	Aucune
NvidiaPowerError	Event	La consommation d'énergie GPUs a dépassé les seuils autorisés.	Aucune
NvidiaThermalError	Event	L'état thermique GPUs a dépassé les seuils autorisés.	Aucune
Erreur NvidiaXid [Code]	Condition	Une erreur critique du processeur graphique s'est produite.	Remplacer ou redémarrer
NvidiaXID[Code]War ning	Événement	Une erreur GPU non critique s'est produite.	Aucune

ContainerRuntime problèmes de santé des nœuds

La condition de surveillance est `ContainerRuntimeReady` pour les problèmes du tableau suivant qui ont une sévérité « Condition ».

Name	Sévérité	Description	Action de réparation
ContainerRuntimeFailed	Événement	L'exécution du conteneur n'a pas réussi à créer un conteneur, ce qui est probablement lié à des problèmes signalés s'ils se produisent de manière répétée.	Aucune
DeprecatedContainerdConfiguration	Event	Une image de conteneur utilisant le manifeste d'image obsolète version 2, schéma 1, a récemment été transférée sur le nœud via. `containerd`	Aucune
KubeletFailed	Event	Le kubelet est passé à l'état d'échec.	Aucune
LivenessProbeFailures	Event	Une défaillance de la sonde de vivacité a été détectée, ce qui peut indiquer des problèmes de code d'application ou des valeurs de délai d'expiration insuffisa	Aucune

Name	Sévérité	Description	Action de réparation
		ntes si cela se produit de manière répétée.	
PodStuckTerminating	Condition	Un pod est ou était bloqué pendant une durée excessive , ce qui peut être dû à des erreurs CRI empêchant la progression de l'état du pod.	Remplacez
ReadinessProbeFailures	Événement	Une défaillance de la sonde de disponibilité a été détectée, ce qui peut indiquer des problèmes de code d'application ou des valeurs de délai d'expiration insuffisantes si cela se produit de manière répétée.	Aucune
[Nom] RepeatedRestart	Événement	Une unité systemd redémarre fréquemment.	Aucune
ServiceFailedToStart	Event	Une unité systemd n'a pas pu démarrer.	Aucune

Problèmes d'état du nœud du noyau

La condition de surveillance est `KernelReady` pour les problèmes du tableau suivant qui ont une sévérité « Condition ».

Name	Sévérité	Description	Action de réparation
AppBlocked	Événement	La tâche a été bloquée pendant une longue période à partir de la planification, généralement en raison d'un blocage au niveau de l'entrée ou de la sortie.	Aucune
AppCrash	Event	Une application sur le nœud a planté.	Aucune
ApproachingKernelPidMax	Event	Le nombre de processus approche le nombre maximum de processus disponibles PIDs selon le `kernel.pid_max` paramètre actuel, après quoi aucun autre processus ne pourra être lancé.	Aucune
ApproachingMaxOpenFiles	Event	Le nombre de fichiers ouverts approche le nombre maximal de fichiers ouverts possibles selon les paramètres actuels du noyau, après quoi l'ouverture de nouveaux fichiers échouera.	Aucune

Name	Sévérité	Description	Action de réparation
ConntrackExceededKernel	Event	Le suivi des connexions a dépassé le maximum pour le noyau et le système n'a pas pu établir de nouvelles connexions, ce qui peut entraîner une perte de paquets.	Aucune
ExcessiveZombieProcesses	Event	Les processus que le système ne peut pas entièrement récupérer s'accumulent en grand nombre, ce qui indique des problèmes d'application et peut conduire à atteindre les limites des processus du système.	Aucune
ForkFailedOutOfPIDs	Condition	Un appel fork ou exec a échoué en raison d'un manque de processus IDs ou de mémoire du système, ce qui peut être dû à des processus zombies ou à un épuisement physique de la mémoire.	Remplacez

Name	Sévérité	Description	Action de réparation
KernelBug	Événement	Un bogue du noyau a été détecté et signalé par le noyau Linux lui-même, bien que cela puisse parfois être causé par des nœuds avec une utilisation élevée du processeur ou de la mémoire, entraînant un retard dans le traitement des événements.	Aucune
LargeEnvironment	Event	Le nombre de variables d'environnement associées à ce processus est supérieur aux prévisions, ce qui peut être dû au fait que de nombreux services sont `enableServiceLinks` définis sur true, ce qui peut entraîner des problèmes de performances.	Aucune

Name	Sévérité	Description	Action de réparation
RapidCron	Event	Une tâche cron s'exécute plus rapidement que toutes les cinq minutes sur ce nœud, ce qui peut avoir un impact sur les performances si la tâche consomme des ressources important es.	Aucune
SoftLockup	Event	Le CPU s'est bloqué pendant un certain temps.	Aucune

Problèmes d'état du nœud de réseau

La condition de surveillance est `NetworkingReady` pour les problèmes du tableau suivant qui ont une sévérité « Condition ».

Name	Sévérité	Description	Action de réparation
BandwidthInExceeded	Événement	La file d'attente ou la suppression de paquets s'explique par le dépasseme nt du maximum de bande passante agrégée entrante pour l'instance.	Aucune
BandwidthOutExceed ed	Event	La file d'attente ou la suppression de paquets s'explique par le dépasseme	Aucune

Name	Sévérité	Description	Action de réparation
		nt du maximum de bande passante agrégée sortante pour l’instance.	
ConntrackExceeded	Event	Le suivi des connexions a dépassé le maximum pour l’instance et le système n’a pas pu établir de nouvelles connexions, ce qui peut entraîner une perte de paquets.	Aucune
IPAMDInconsistent État	Événement	L'état du point de contrôle IPAMD sur le disque ne reflète pas l'environnement d' IPs exécution du conteneur.	Aucune
IPAMDNoIPs	Event	Il n'y a plus d'adresse s IP sur l'IPAMD.	Aucune
IPAMDNotPrêt	Condition	IPAMD ne parvient pas à se connecter au serveur API.	Remplacez
IPAMDNotCourir	Condition	Le processus Amazon VPC CNI n'a pas été détecté comme étant en cours d'exécution.	Remplacez

Name	Sévérité	Description	Action de réparation
IPAMDRepeatedlyRedémarrer	Événement	Le service IPAMD s'est redémarré plusieurs fois.	Aucune
InterfaceNotRunning	Condition	Cette interface semble ne pas fonctionner ou il y a des problèmes de réseau.	Remplacez
InterfaceNotUp	Condition	Cette interface semble ne pas être active ou il y a des problèmes de réseau.	Remplacez
KubeProxyNotReady	Événement	Kube-proxy n'a pas réussi à surveiller ou à répertorier les ressources.	Aucune
LinkLocalExceeded	Event	Le système a supprimé des paquets car le PPS du trafic vers les services mandataires locaux a dépassé le maximum de l'interface réseau.	Aucune
MACAddressPolicyMisconfigured	Event	La valeur de la configuration du lien systemd-networkd est incorrecte. `MACAddressPolicy`	Aucune

Name	Sévérité	Description	Action de réparation
MissingDefaultRoutes	Event	Il manque des règles de routage par défaut.	Aucune
Manquant IPRoutes	Événement	Il manque des itinérair es pour Pod IPs.	Aucune
Manquant IPRules	Événement	Il manque des règles pour Pod IPs.	Aucune
MissingLoopbackInt erface	Condition	L'interface de bouclage est manquante dans cette instance, ce qui entraîne l'échec des services dépendant de la connectivité locale.	Remplacez
NetworkSysctl	Événement	Les `sysctl` paramètres réseau de ce nœud sont potentiellement incorrects.	Aucune
PPSExceeded	Event	Des paquets ont été mis en file d'attente ou supprimés car le PPS bidirectionnel a dépassé le maximum pour l'instance.	Aucune

Name	Sévérité	Description	Action de réparation
PortConflict	Event	Si un Pod utilise HostPort, il peut écrire des `iptables` règles qui remplacent les ports déjà liés de l'hôte, empêchant potentiellement l'accès du serveur API à. `kubelet`	Aucune
UnexpectedRejectRule	Event	Une `DROP` règle `REJECT` ou un élément inattendu a été détecté dans le`iptables`, bloquant potentiellement le trafic attendu.	Aucune

Problèmes d'état du nœud de stockage

La condition de surveillance est `StorageReady` pour les problèmes du tableau suivant qui ont une sévérité « Condition ».

Name	Sévérité	Description	Action de réparation
EBSInstanceIOPSExceeded	Événement	Le nombre maximal d'IOPS pour l'instance a été dépassé.	Aucune
EBSInstanceThroughputExceeded	Event	Le débit maximal de l'instance a été dépassé.	Aucune

Name	Sévérité	Description	Action de réparation
EBSVolume IOPSExceeded	Event	Le nombre maximal d'IOPS pour un volume EBS donné a été dépassé.	Aucune
EBSVolume ThroughputExceeded	Event	Le débit maximal pour un volume Amazon EBS spécifique a été dépassé.	Aucune
EtcHostsMountFailed	Event	Le montage du kubelet généré `/etc/hosts` a échoué en raison du `/var/lib/kubelet/pods` remontage des données utilisateur pendant le fonctionnement. `kubelet-container`	Aucune
IODelays	Event	Un retard d'entrée ou de sortie a été détecté dans un processus, ce qui peut indiquer un provisionnement d'entrée-sortie insuffisant s'il est excessif.	Aucune

Name	Sévérité	Description	Action de réparation
KubeletDiskUsageSlow	Event	Le signale `kubelet` une utilisation lente du disque lors de la tentative d'accès au système de fichiers. Cela peut indiquer une insuffisance des entrées-sorties du disque ou des problèmes de système de fichiers.	Aucune
XFSSmallAverageClusterSize	Event	La taille moyenne du cluster XFS est faible, ce qui indique une fragmentation excessive de l'espace libre. Cela peut empêcher la création de fichiers malgré les inodes disponibles ou l'espace libre.	Aucune

Afficher l'état de santé de vos nœuds

Cette rubrique explique les outils et les méthodes disponibles pour surveiller l'état de santé des nœuds dans les clusters Amazon EKS. Les informations couvrent les conditions des nœuds, les événements et les cas de détection qui vous aident à identifier et à diagnostiquer les problèmes au niveau des nœuds. Utilisez les commandes et les modèles décrits ici pour inspecter les ressources de santé des nœuds, interpréter les conditions d'état et analyser les événements des nœuds à des fins de dépannage opérationnel.

Vous pouvez obtenir certaines informations sur l'état de santé des nœuds à l'aide des commandes Kubernetes pour tous les nœuds. Si vous utilisez l'agent de surveillance des nœuds via le mode automatique Amazon EKS ou le module complémentaire géré par Amazon EKS, vous obtiendrez une

plus grande variété de signaux de nœuds pour vous aider à résoudre les problèmes. Les descriptions des problèmes de santé détectés par l'agent de surveillance des nœuds sont également disponibles dans le tableau de bord d'observabilité. Pour de plus amples informations, consultez the section called "État du nœud".

Conditions des nœuds

Les conditions des nœuds représentent des problèmes terminaux nécessitant des mesures correctives telles que le remplacement ou le redémarrage d'une instance.

Pour obtenir les conditions de tous les nœuds :

```
kubectl get nodes -o 'custom-
columns=NAME:.metadata.name,CONDITIONS:.status.conditions[*].type,STATUS:.status.conditions[*].
```

Pour obtenir les conditions détaillées d'un nœud spécifique :

```
kubectl describe node node-name
```

Exemple de sortie de condition d'un nœud en bon état :

```
  - lastHeartbeatTime: "2024-11-21T19:07:40Z"
    lastTransitionTime: "2024-11-08T03:57:40Z"
    message: Monitoring for the Networking system is active
    reason: NetworkingIsReady
    status: "True"
    type: NetworkingReady
```

Exemple de condition d'un nœud en mauvais état présentant un problème de réseau :

```
  - lastHeartbeatTime: "2024-11-21T19:12:29Z"
    lastTransitionTime: "2024-11-08T17:04:17Z"
    message: IPAM-D has failed to connect to API Server which could be an issue with
      IPTable rules or any other network configuration.
    reason: IPAMDNotReady
    status: "False"
    type: NetworkingReady
```

Événements des nœuds

Les événements des nœuds indiquent des problèmes temporaires ou des configurations sous-optimales.

Pour obtenir tous les événements signalés par l'agent de surveillance des nœuds :

Lorsque l'agent de surveillance des nœuds est disponible, vous pouvez exécuter la commande suivante.

```
kubectl get events --field-selector=reportingComponent=eks-node-monitoring-agent
```

Exemple de sortie :

```
LAST SEEN   TYPE      REASON       OBJECT
 MESSAGE
4s          Warning   SoftLockup   node/ip-192-168-71-251.us-west-2.compute.internal
 CPU stuck for 23s
```

Pour obtenir les événements de tous les nœuds

```
kubectl get events --field-selector involvedObject.kind=Node
```

Pour obtenir les événements d'un nœud spécifique

```
kubectl get events --field-selector involvedObject.kind=Node,involvedObject.name=node-name
```

Pour surveiller les événements en temps réel

```
kubectl get events -w --field-selector involvedObject.kind=Node
```

Exemple de sortie d'événement :

```
LAST SEEN   TYPE     REASON           OBJECT         MESSAGE
2m          Warning  MemoryPressure   Node/node-1    Node experiencing memory pressure
5m          Normal   NodeReady        Node/node-1    Node became ready
```

Commandes de dépannage courantes

```
# Get comprehensive node status
kubectl get node node-name -o yaml

# Watch node status changes
kubectl get nodes -w

# Get node metrics
kubectl top node
```

Extraction des journaux d'un nœud géré à l'aide de kubectl et S3

Découvrez comment extraire les journaux d'un nœud géré Amazon EKS disposant de l'agent de surveillance des nœuds.

Conditions préalables

Vérifiez que vous avez les éléments suivants :

- Un cluster Amazon EKS existant avec l'agent de surveillance des nœuds installé. Pour de plus amples informations, veuillez consulter the section called "État du nœud".
- L'outil en ligne de commande `kubectl` installé et configuré pour communiquer avec votre cluster.
- La AWS CLI s'est installée et s'est connectée avec des autorisations suffisantes pour créer des compartiments et des objets S3.
- Une version récente de Python 3 installée
- Le AWS SDK pour Python 3, Boto 3, est installé.

Étape 1 : créer un compartiment S3 de destination (facultatif)

Si vous ne disposez pas encore d'un compartiment S3 pour stocker les journaux, créez-en un. Utilisez la commande AWS CLI suivante. Le compartiment utilise par défaut la liste de contrôle d'accès `private`. *bucket-name*Remplacez-le par le nom de compartiment unique que vous avez choisi.

```
aws s3api create-bucket --bucket <bucket-name>
```

Étape 2 : créer une URL S3 pré-signée pour HTTP PUT

Amazon EKS renvoie les journaux du nœud par une opération HTTP PUT vers l'URL que vous indiquez. Dans ce tutoriel, nous allons générer une URL HTTP PUT S3 pré-signée.

Les journaux seront renvoyés sous la forme d'une archive gzip, avec l'extension `.tar.gz`.

> **Note**
>
> Vous devez utiliser l' AWS API ou un SDK pour créer l'URL de téléchargement S3 pré-signée afin qu'EKS télécharge le fichier journal. Vous ne pouvez pas créer d'URL de téléchargement S3 pré-signée à l'aide de la AWS CLI.

1. Déterminez où vous souhaitez stocker les journaux dans le compartiment. Par exemple, vous pouvez utiliser *`2024-11-12/logs1.tar.gz`* comme clé.
2. Enregistrez le code Python suivant dans le fichier *`presign-upload.py`*. Remplacez *`<bucket-name>`* et *`<key>`*. La clé doit se terminer par `.tar.gz`.

```
import boto3; print(boto3.client('s3').generate_presigned_url(
   ClientMethod='put_object',
   Params={'Bucket': '<bucket-name>', 'Key': '<key>'},
   ExpiresIn=1000
))
```

3. Exécutez le script avec

```
python presign-upload.py
```

4. Notez le résultat de l'URL. Utilisez cette valeur à l'étape suivante comme `http-put-destination`.

Pour plus d'informations, consultez Générer une URL présignée pour télécharger un fichier dans la documentation du SDK AWS Boto3 pour Python.

Étape 3 : Création d'une NodeDiagnostic ressource

Identifiez le nom du nœud à partir duquel vous souhaitez collecter les journaux.

Créez un manifeste `NodeDiagnostic` qui utilise le nom du nœud comme nom de la ressource et fournissant une destination URL HTTP PUT.

```
apiVersion: eks.amazonaws.com/v1alpha1
kind: NodeDiagnostic
metadata:
    name: <node-name>
spec:
    logCapture:
        destination: http-put-destination
```

Appliquez le fichier manifeste à votre cluster.

```
kubectl apply -f nodediagnostic.yaml
```

Vous pouvez vérifier l'état de la collecte en décrivant la ressource `NodeDiagnostic` :

- Un état `Success` ou `SuccessWithErrors` indique que la tâche est terminée et que les journaux ont été téléversés vers la destination indiquée (l'état `SuccessWithErrors` indique que certains journaux peuvent être manquants)
- Si l'état est Échec, vérifiez que l'URL de téléversement est bien formée et qu'elle n'a pas expiré.

```
kubectl describe nodediagnostics.eks.amazonaws.com/<node-name>
```

Étape 4 : télécharger des journaux depuis S3

Attendez environ une minute avant d'essayer de télécharger les journaux. Utilisez ensuite la CLI S3 pour télécharger les journaux.

```
# Once NodeDiagnostic shows Success status, download the logs
aws s3 cp s3://<bucket-name>/key ./<path-to-node-logs>.tar.gz
```

Étape 5 : Nettoyer les NodeDiagnostic ressources

- Les ressources `NodeDiagnostic` ne sont pas supprimées automatiquement. Vous devez les supprimer manuellement après avoir récupéré les artefacts de vos journaux

```
# Delete the NodeDiagnostic resource
kubectl delete nodediagnostics.eks.amazonaws.com/<node-name>
```

Présentation des nœuds hybrides Amazon EKS

Avec les nœuds hybrides Amazon EKS, vous pouvez utiliser votre infrastructure sur site et en périphérie comme nœuds dans les clusters Amazon EKS. AWS gère le plan de contrôle Kubernetes hébergé par AWS du cluster Amazon EKS, et vous gérez les nœuds hybrides qui s'exécutent dans vos environnements sur site ou en périphérie. Cela unifie la gestion de Kubernetes dans tous vos environnements et décharge la gestion du plan de contrôle Kubernetes vers AWS pour vos applications sur site et en périphérie.

Les nœuds hybrides Amazon EKS fonctionnent avec n'importe quel matériel ou machine virtuelle sur site, apportant l'efficacité, la capacité de mise à l'échelle et la disponibilité d'Amazon EKS partout où vos applications doivent être exécutées. Vous pouvez utiliser un large éventail de fonctionnalités Amazon EKS avec les nœuds hybrides Amazon EKS, notamment les modules complémentaires Amazon EKS, l'identité du pod Amazon EKS, les entrées d'accès au cluster, les informations sur le cluster et la prise en charge étendue des versions Kubernetes. Les nœuds hybrides Amazon EKS s'intègre nativement à des services AWS tels que Systems Manager AWS, Rôles Anywhere IAM AWS, le service géré Amazon pour Prometheus et Amazon CloudWatch pour la surveillance centralisée, la journalisation et la gestion des identités.

Avec les nœuds hybrides Amazon EKS, il n'y a pas d'engagement préalable ni de frais minimums, et vous êtes facturé à l'heure pour les ressources vCPU de vos nœuds hybrides lorsqu'ils sont connectés à vos clusters Amazon EKS. Pour plus d'informations sur les tarifs, consultez la page Tarifs Amazon EKS.

Fonctionnalités

Les nœuds hybrides EKS présentent les caractéristiques générales suivantes :

- Plan de contrôle Kubernetes géré : AWS gère le plan de contrôle Kubernetes hébergé AWS du cluster EKS, et vous gérez les nœuds hybrides qui s'exécutent dans vos environnements sur site ou en périphérie. Cela unifie la gestion de Kubernetes dans tous vos environnements et décharge la gestion du plan de contrôle Kubernetes vers AWS pour vos applications sur site et en périphérie.

En déplaçant le plan de contrôle Kubernetes vers AWS, vous pouvez conserver la capacité sur site pour vos applications et être assuré que le plan de contrôle Kubernetes s'adapte à vos charges de travail.

- Expérience EKS cohérente : la plupart des fonctionnalités EKS sont prises en charge par les nœuds hybrides EKS pour une expérience EKS cohérente dans vos environnements sur site et dans le cloud, notamment les modules complémentaires EKS, l'identité du pod EKS, les entrées d'accès au cluster, les informations sur le cluster, la prise en charge étendue des versions Kubernetes, etc. Pour plus d'informations, consultez la section the section called "Configurer les modules complémentaires" consacrée aux modules complémentaires EKS pris en charge par les nœuds hybrides EKS.
- Observabilité centralisée et gestion des identités : les nœuds hybrides EKS s'intègrent nativement à des services AWS tels qu'AWS Systems Manager, Rôles Anywhere IAM AWS, le service géré Amazon pour Prometheus et Amazon CloudWatch pour la surveillance centralisée, la journalisation et la gestion des identités.
- Burst-to-cloud ou ajout de capacité sur site : Un seul cluster EKS peut être utilisé pour exécuter des nœuds hybrides et des nœuds dans des régions AWS, des zones locales AWS ou de AWS Outposts afin d'ajouter de la capacité burst-to-cloud ou sur site à vos clusters EKS. Pour plus d'informations, consultez la section Considérations relatives aux clusters en mode mixte.
- Infrastructure flexible : les nœuds hybrides EKS suivent une approche « apportez votre propre infrastructure » et sont indépendants de l'infrastructure que vous utilisez pour les nœuds hybrides. Vous pouvez exécuter des nœuds hybrides sur des machines physiques ou virtuelles, ainsi que sur des architectures x86 et ARM, ce qui permet de migrer les charges de travail sur site s'exécutant sur des nœuds hybrides vers différents types d'infrastructures.
- Réseau flexible : Avec les nœuds hybrides EKS, la communication entre le plan de contrôle EKS et les nœuds hybrides est acheminée via le VPC et les sous-réseaux que vous transmettez lors de la création du cluster, ce qui s'appuie sur le mécanisme existant dans EKS pour la mise en réseau du plan de contrôle vers les nœuds. Cette solution s'adapte à votre méthode préférée pour connecter vos réseaux sur site à un VPC dans AWS. Plusieurs options documentées sont disponibles, notamment le VPN site à site AWS, Direct Connect AWS ou votre propre solution VPN. Vous pouvez choisir la méthode qui correspond le mieux à votre cas d'utilisation.

Limites

- Jusqu'à 15 CIDR pour les réseaux de nœuds distants et 15 CIDR pour les réseaux de pods distants par cluster sont pris en charge.

Considérations

- Les nœuds hybrides EKS peuvent être utilisés avec des clusters EKS nouveaux ou existants.
- Les nœuds hybrides EKS sont disponibles dans toutes les régions AWS, à l'exception des régions AWS GovCloud (États-Unis) et AWS Chine.
- Les nœuds hybrides EKS doivent disposer d'une connexion fiable entre votre environnement sur site et AWS. Les nœuds hybrides EKS ne sont pas adaptés aux environnements déconnectés, perturbés, intermittents ou limités (DDIL). Si vous utilisez un environnement DDIL, envisagez Amazon EKS Anywhere. Consultez les meilleures pratiques pour les nœuds hybrides EKS pour obtenir des informations sur le comportement des nœuds hybrides en cas de déconnexion réseau.
- L'exécution de nœuds hybrides EKS sur une infrastructure cloud, y compris les régions AWS, les zones locales AWS, AWS Outposts ou d'autres clouds, n'est pas prise en charge. Des frais liés aux nœuds hybrides vous seront facturés si vous exécutez des nœuds hybrides sur des instances Amazon EC2.
- La facturation des nœuds hybrides commence lorsque les nœuds rejoignent le cluster EKS et s'arrête lorsqu'ils sont supprimés du cluster. Veillez à supprimer vos nœuds hybrides de votre cluster EKS si vous ne les utilisez pas.

Ressources supplémentaires

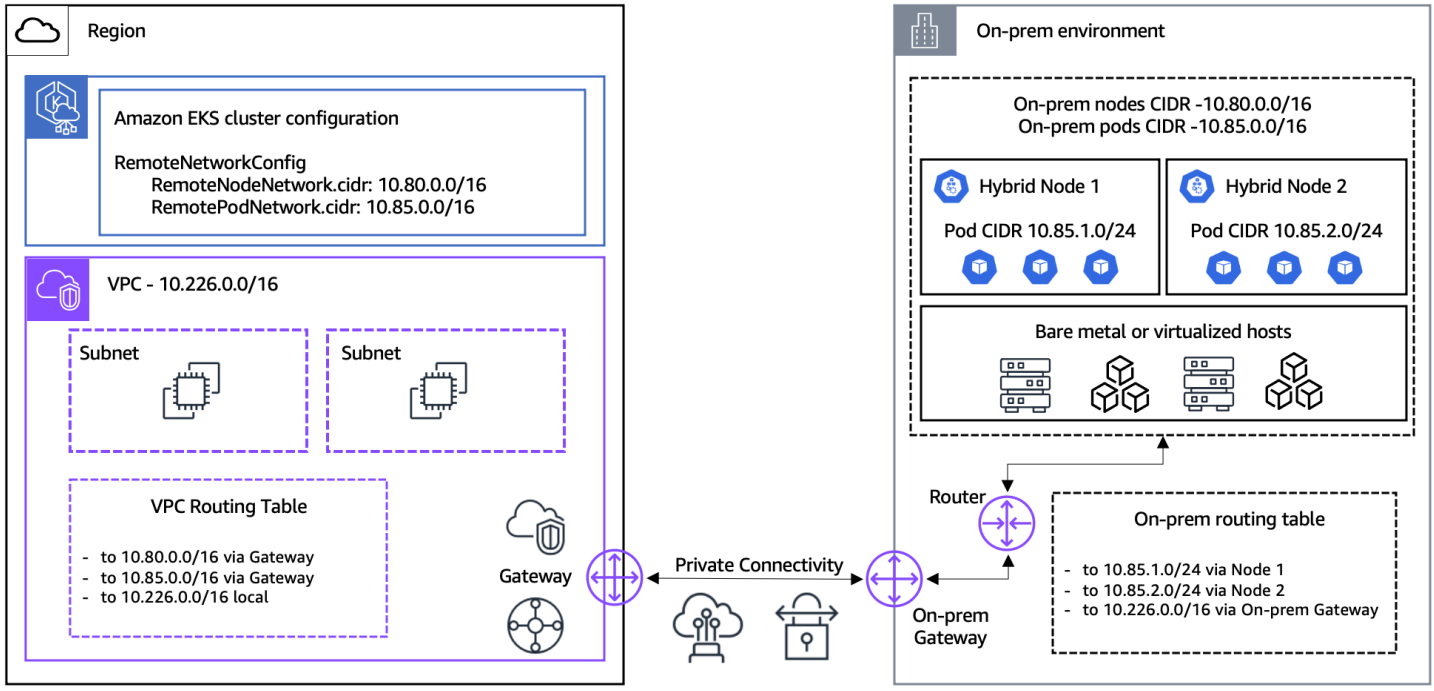
- Atelier sur les nœuds hybrides EKS : instructions étape par étape pour déployer des nœuds hybrides EKS dans un environnement de démonstration.
- AWS re:Invent : Nœuds hybrides EKS : AWS session re:Invent présentant le lancement des nœuds hybrides EKS avec un client qui montre comment il utilise les nœuds hybrides EKS dans son environnement.
- AWSRe:post : Mise en réseau de clusters pour les nœuds hybrides EKS : article expliquant différentes méthodes de configuration de la mise en réseau pour les nœuds hybrides EKS.
- AWS blog: Run GenAI inference across environments with EKS Hybrid Nodes : article de blog expliquant comment exécuter l'inférence de l'IA générative dans différents environnements avec les nœuds hybrides EKS.

Configuration préalable requise pour les nœuds hybrides

Pour utiliser les nœuds hybrides Amazon EKS, vous devez disposer d'une connectivité privée depuis votre environnement sur site vers/depuis AWS, des serveurs de matériel nu ou des machines

virtuelles avec un système d'exploitation pris en charge, et avoir configuré les activations hybrides Rôles Anywhere IAM AWS ou AWS Systems Manager (SSM). Vous êtes responsable de la gestion de ces prérequis tout au long du cycle de vie des nœuds hybrides.

- Connectivité réseau hybride depuis votre environnement sur site vers/depuis AWS
- Infrastructure sous forme de machines physiques ou virtuelles
- Système d'exploitation compatible avec les nœuds hybrides
- Fournisseur d'informations d'identification IAM sur site configuré



Connectivité réseau hybride

La communication entre le plan de contrôle Amazon EKS et les nœuds hybrides est acheminée via le VPC et les sous-réseaux que vous transmettez lors de la création du cluster, qui s'appuie sur le mécanisme existant dans Amazon EKS pour la mise en réseau du plan de contrôle vers les nœuds. Plusieurs options documentées sont disponibles pour connecter votre environnement sur site à votre VPC, notamment le VPN site à site AWS, Direct Connect AWS ou votre propre connexion VPN. Consultez les guides d'utilisation du VPN site à site AWS et de Direct Connect AWS pour plus d'informations sur l'utilisation de ces solutions pour votre connexion réseau hybride.

Pour une expérience optimale, nous vous recommandons de disposer d'une connexion réseau fiable d'au moins 100 Mbit/s et d'une latence aller-retour maximale de 200 ms pour la connexion

des nœuds hybrides à la région AWS. Il s'agit d'une recommandation générale qui s'applique à la plupart des cas d'utilisation, mais qui ne constitue pas une exigence stricte. Les exigences en matière de bande passante et de latence peuvent varier en fonction du nombre de nœuds hybrides et des caractéristiques de votre charge de travail, telles que la taille de l'image de l'application, l'élasticité de l'application, les configurations de surveillance et de journalisation, ainsi que les dépendances de l'application pour accéder aux données stockées dans d'autres services AWS. Nous vous recommandons d'effectuer des tests avec vos propres applications et environnements avant le déploiement en production afin de vérifier que votre configuration réseau répond aux exigences de vos charges de travail.

Configuration du réseau local

Vous devez activer l'accès réseau entrant depuis le plan de contrôle Amazon EKS vers votre environnement sur site afin de permettre au plan de contrôle Amazon EKS de communiquer avec les nœuds hybrides en cours d'exécution `kubelet` et, éventuellement, avec les webhooks exécutés sur vos nœuds hybrides. En outre, vous devez activer l'accès au réseau sortant pour vos nœuds hybrides et les composants qui s'exécutent sur ceux-ci afin de communiquer avec le plan de contrôle Amazon EKS. Vous pouvez configurer cette communication pour qu'elle reste entièrement privée sur votre connexion Direct Connect AWS, VPN site à site AWS ou votre propre connexion VPN.

Les plages Routage inter-domaines sans classe (CIDR) que vous utilisez pour vos réseaux de nœuds et de pods sur site doivent utiliser les plages d'adresses IPv4 RFC-1918. Votre routeur local doit être configuré avec des routes vers vos nœuds locaux et, éventuellement, vers vos pods. Pour plus d'informations sur les exigences réseau sur site, y compris la liste complète des ports et protocoles requis qui doivent être activés dans votre pare-feu et votre environnement sur site, consultez the section called "Configuration réseau sur site".

Configuration du cluster EKS

Pour réduire au minimum la latence, nous vous recommandons de créer votre cluster Amazon EKS dans la région AWS la plus proche de votre environnement sur site ou périphérique. Vous transmettez les CIDR de votre nœud et de votre pod sur site lors de la création du cluster Amazon EKS via deux champs API : `RemoteNodeNetwork` et `RemotePodNetwork`. Vous devrez peut-être discuter avec votre équipe réseau sur site afin d'identifier votre nœud sur site et les CIDR des pods. Le CIDR du nœud est attribué à partir de votre réseau local et le CIDR du pod est attribué à partir de l'interface réseau de conteneur (CNI) que vous utilisez si vous utilisez un réseau superposé pour votre CNI. Cilium et Calico utilisent par défaut des réseaux superposés.

Les CIDR des nœuds et pods sur site que vous configurez via les champs `RemoteNodeNetwork` et `RemotePodNetwork` sont utilisés pour configurer le plan de contrôle Amazon EKS afin d'acheminer le trafic via votre VPC vers le `kubelet` et les pods s'exécutant sur vos nœuds hybrides. Les CIDR de vos nœuds et pods sur site ne peuvent pas se chevaucher entre eux, ni avec le CIDR VPC que vous transmettez lors de la création du cluster, ni avec la configuration IPv4 du service pour votre cluster Amazon EKS. De plus, les CIDR Pod doivent être uniques pour chaque cluster EKS afin que votre routeur sur site puisse acheminer le trafic.

Nous vous recommandons d'utiliser un accès public ou privé pour le point de terminaison du serveur API Amazon EKS Kubernetes. Si vous choisissez « Public et privé », le point de terminaison du serveur API Amazon EKS Kubernetes sera toujours résolu vers les adresses IP publiques des nœuds hybrides s'exécutant en dehors de votre VPC, ce qui peut empêcher vos nœuds hybrides de rejoindre le cluster. Lorsque vous utilisez l'accès au point de terminaison public, le point de terminaison du serveur API Kubernetes est résolu en adresses IP publiques et la communication entre les nœuds hybrides et le plan de contrôle Amazon EKS est acheminée via Internet. Lorsque vous choisissez l'accès à un point de terminaison privé, le point de terminaison du serveur API Kubernetes est résolu en adresses IP privées et la communication entre les nœuds hybrides et le plan de contrôle Amazon EKS est acheminée via votre liaison de connectivité privée, dans la plupart des cas Direct Connect AWS ou VPN site à site AWS.

Configuration VPC

Vous devez configurer le VPC que vous transmettez lors de la création du cluster Amazon EKS avec des routes dans sa table de routage pour votre nœud sur site et, éventuellement, des réseaux de pods avec votre passerelle privée virtuelle (VGW) ou votre passerelle de transit (TGW) comme cible. Un exemple est illustré ci-dessous. Remplacez `REMOTE_NODE_CIDR` et `REMOTE_POD_CIDR` par les valeurs correspondant à votre réseau sur site.

Destination	Cible	Description
10.226.0.0/16	local	Trafic local vers les routes VPC au sein du VPC
REMOTE_NODE_CIDR	tgw-abcdef123456	CIDR du nœud sur site, acheminer le trafic vers le TGW

Destination	Cible	Description
REMODE_POD_CIDR	tgw-abcdef123456	CIDR du pod sur site, acheminer le trafic vers le TGW

Configuration du groupe de sécurité

Lorsque vous créez un cluster, Amazon EKS crée un groupe de sécurité nommé `eks-cluster-sg-<cluster-name>-<uniqueID>`. Vous ne pouvez pas modifier les règles entrantes de ce groupe de sécurité de cluster, mais vous pouvez restreindre les règles sortantes. Vous devez ajouter un groupe de sécurité supplémentaire à votre cluster pour permettre au kubelet et, éventuellement, aux webhooks s'exécutant sur vos nœuds hybrides de contacter le plan de contrôle Amazon EKS. Les règles entrantes requises pour ce groupe de sécurité supplémentaire sont indiquées ci-dessous. Remplacez `REMOTE_NODE_CIDR` et `REMOTE_POD_CIDR` par les valeurs correspondant à votre réseau sur site.

Nom	ID de règle du groupe de sécurité	Versions d'adresses IP	Type	Protocole	Plage de ports	Source
Nœud entrant sur site	sgr-abcdef123456	IPv4	HTTPS	TCP	443	REMOTE_NODE_CIDR
Envoi du pod sur site	sgr-abcdef654321	IPv4	HTTPS	TCP	443	REMOTE_POD_CIDR

Infrastructure

Vous devez disposer de serveurs de matériel nu ou de machines virtuelles pouvant être utilisés comme nœuds hybrides. Les nœuds hybrides sont indépendants de l'infrastructure sous-jacente et prennent en charge les architectures x86 et ARM. Les nœuds hybrides Amazon EKS suivent une approche « apportez votre propre infrastructure », dans laquelle vous êtes responsable de l'approvisionnement et de la gestion des serveurs de matériel nu ou des machines virtuelles que

vous utilisez pour les nœuds hybrides. Bien qu'il n'y ait pas d'exigence minimale stricte en matière de ressources, nous vous recommandons d'utiliser des hôtes disposant d'au moins 1 vCPU et 1 Go de RAM pour les nœuds hybrides.

Système d'exploitation

Bottlerocket, Amazon Linux 2023 (AL2023), Ubuntu et RHEL sont validés en permanence pour être utilisés comme système d'exploitation des nœuds hybrides. Bottlerocket est pris en charge par AWS uniquement dans les environnements VMware vSphere. AL2023 n'est pas couvert par les plans d'assistance AWS lorsqu'il est exécuté en dehors d'Amazon EC2. AL2023 ne peut être utilisé que dans des environnements virtualisés sur site. Pour plus d'informations, consultez le Guide de l'utilisateur Amazon Linux 2023. AWS prend en charge l'intégration des nœuds hybrides avec les systèmes d'exploitation Ubuntu et RHEL, mais ne fournit pas de support pour le système d'exploitation lui-même.

Vous êtes responsable de la mise à disposition et de la gestion du système d'exploitation. Lorsque vous testez des nœuds hybrides pour la première fois, il est plus simple d'exécuter la CLI des nœuds hybrides Amazon EKS (`nodeadm`) sur un hôte déjà provisionné. Pour les déploiements en production, nous vous recommandons d'inclure `nodeadm` dans vos images de système d'exploitation de référence une configuration permettant leur exécution en tant que service systemd afin de joindre automatiquement les hôtes aux clusters Amazon EKS au démarrage de l'hôte.

Fournisseur d'informations d'identification IAM sur site

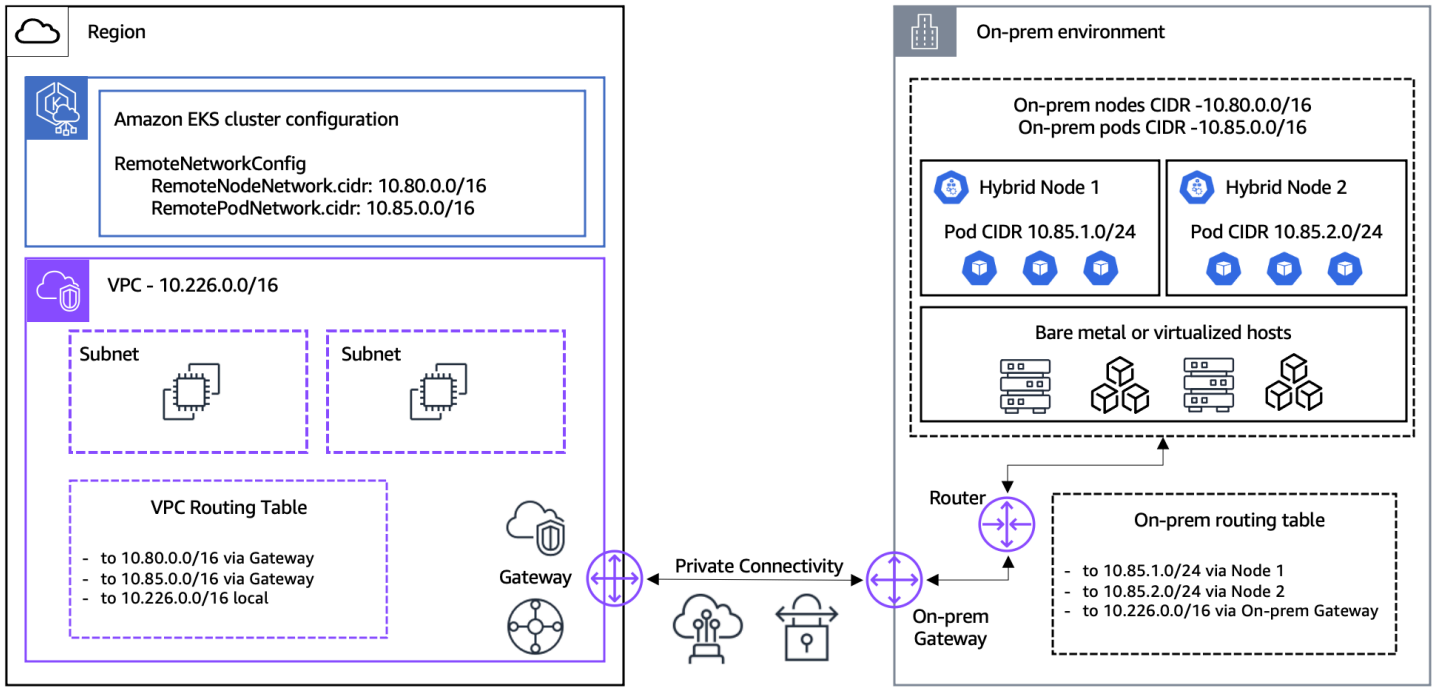
Les nœuds hybrides Amazon EKS utilisent des informations d'identification IAM temporaires fournies par les activations hybrides AWS SSM ou Rôles Anywhere IAM AWS pour s'authentifier auprès du cluster Amazon EKS. Vous devez utiliser soit les activations hybrides AWS SSM, soit Rôles Anywhere IAM AWS avec l'interface CLI des nœuds hybrides Amazon EKS (`nodeadm`). Nous vous recommandons d'utiliser les activations hybrides SSM AWS si vous ne disposez pas d'une infrastructure à clé publique (PKI) avec une autorité de certification (CA) et des certificats pour vos environnements sur site. Si vous disposez déjà d'une infrastructure PKI et de certificats sur site, utilisez Rôles Anywhere IAM AWS.

Comme pour the section called "Rôle IAM de nœud" avec les nœuds exécutés sur Amazon EC2, vous allez créer un rôle IAM pour les nœuds hybrides avec les autorisations nécessaires pour joindre les nœuds hybrides aux clusters Amazon EKS. Si vous utilisez Rôles Anywhere IAM AWS, configurez une politique de confiance qui autorise Rôles Anywhere IAM AWS à assumer le rôle IAM des nœuds hybrides et configurez votre profil Rôles Anywhere IAM AWS avec le rôle IAM des nœuds hybrides comme rôle assumable. Si vous utilisez SSM AWS, configurez une stratégie de confiance

qui autorise SSM AWS à assumer le rôle IAM des nœuds hybrides et créez l'activation hybride avec le rôle IAM des nœuds hybrides. Consultez the section called "Préparation des informations d'identification" pour découvrir comment créer le rôle IAM des nœuds hybrides avec les autorisations requises.

Préparer la mise en réseau pour les nœuds hybrides

Cette rubrique fournit une vue d'ensemble de la configuration réseau que vous devez avoir configurée avant de créer votre cluster Amazon EKS et d'y attacher des nœuds hybrides. Ce guide part du principe que vous avez satisfait aux exigences requises pour la connectivité réseau hybride à l'aide d'AWS Site-to-Site un VPN, de AWS Direct Connect ou de votre propre solution VPN.



Configuration réseau sur site

Configuration réseau minimale requise

Pour une expérience optimale, nous vous recommandons de disposer d'une connectivité réseau fiable d'au moins 100 Mbits/s et d'une latence aller-retour maximale de 200 ms pour la connexion des nœuds hybrides à la AWS région. Il s'agit d'une recommandation générale qui s'applique à la plupart des cas d'utilisation, mais qui ne constitue pas une exigence stricte. Les exigences en matière de bande passante et de latence peuvent varier en fonction du nombre de nœuds hybrides et des caractéristiques de votre charge de travail, telles que la taille de l'image de l'application, l'élasticité de l'application, les configurations de surveillance et de journalisation, ainsi que les dépendances

des applications en matière d'accès aux données stockées dans d'autres AWS services. Nous vous recommandons d'effectuer des tests avec vos propres applications et environnements avant le déploiement en production afin de vérifier que votre configuration réseau répond aux exigences de vos charges de travail.

Nœud et module sur site CIDRs

Identifiez le nœud et le pod CIDRs que vous utiliserez pour vos nœuds hybrides et les charges de travail qui y sont exécutées. Le CIDR du nœud est attribué à partir de votre réseau local et le CIDR du pod est attribué à partir de votre interface réseau de conteneur (CNI) si vous utilisez un réseau superposé pour votre CNI. Vous transmettez votre nœud CIDRs et votre pod locaux CIDRs en tant qu'entrées lorsque vous créez votre cluster EKS avec les `RemotePodNetwork` champs `RemoteNodeNetwork` et. Votre nœud local CIDRs doit être routable sur votre réseau local. Consultez la section suivante pour obtenir des informations sur la routabilité CIDR du pod sur site.

Les blocs CIDR des nœuds et pods sur site doivent répondre aux exigences suivantes :

1. Se situer dans l'une des plages RFC-1918 `IPv4` suivantes : `10.0.0.0/8`, `172.16.0.0/12`, ou `192.168.0.0/16`.
2. Ne pas se chevaucher entre eux, le CIDR VPC pour votre cluster EKS ou votre CIDR de service Kubernetes `IPv4`.

Routage du réseau de pods sur site

Lorsque vous utilisez des nœuds hybrides EKS, nous vous recommandons généralement de rendre votre module sur site CIDRs routable sur votre réseau local afin de permettre une communication et des fonctionnalités complètes entre les environnements cloud et sur site.

Réseaux de pods routables

Si vous êtes en mesure de rendre votre réseau de pods routable sur votre réseau local, suivez les instructions ci-dessous.

1. Configurez le champ `RemotePodNetwork` pour votre cluster EKS avec votre CIDR de pod sur site, vos tables de routage VPC avec votre CIDR de pod sur site et votre groupe de sécurité de cluster EKS avec votre CIDR de pod sur site.
2. Il existe plusieurs techniques que vous pouvez utiliser pour rendre votre pod CIDR sur site routable sur votre réseau sur site, notamment le protocole de passerelle frontière (BGP), les routes statiques ou d'autres solutions de routage personnalisées. Le BGP est la solution recommandée

car elle est plus évolutive et plus facile à gérer que les solutions alternatives qui nécessitent une configuration de routage personnalisée ou manuelle. AWS prend en charge les fonctionnalités BGP de Cilium et Calico pour les modules publicitaires CIDRs, voir the section called "Configurer CNI" et the section called "CIDR de pods distants routables" pour plus d'informations.

3. Les webhooks peuvent s'exécuter sur des nœuds hybrides, car le plan de contrôle EKS est capable de communiquer avec les adresses IP des pods attribuées aux webhooks.
4. Les charges de travail exécutées sur les nœuds cloud peuvent communiquer directement avec les charges de travail exécutées sur les nœuds hybrides du même cluster EKS.
5. D'autres AWS services, tels que les équilibreurs de charge des AWS applications et Amazon Managed Service for Prometheus, sont capables de communiquer avec les charges de travail exécutées sur des nœuds hybrides afin d'équilibrer le trafic réseau et de récupérer les métriques des pods.

Réseaux de pods non routables

Si vous ne parvenez pas à rendre vos réseaux de pods routables sur votre réseau local, suivez les instructions ci-dessous.

1. Les webhooks ne peuvent pas s'exécuter sur des nœuds hybrides, car ils nécessitent une connectivité entre le plan de contrôle EKS et les adresses IP des pods attribuées aux webhooks. Dans ce cas, nous vous recommandons d'exécuter les webhooks sur les nœuds cloud du même cluster EKS que vos nœuds hybrides. Pour plus d'informations, consultez the section called "Configuration de webhooks".
2. Les charges de travail exécutées sur des nœuds cloud ne peuvent pas communiquer directement avec les charges de travail exécutées sur des nœuds hybrides lorsque vous utilisez VPC CNI pour les nœuds cloud et Cilium ou Calico pour les nœuds hybrides.
3. Utilisez la distribution du trafic de service pour maintenir le trafic local dans la zone d'où il provient. Pour plus d'informations sur la distribution du trafic de service, voir the section called "Configuration de la distribution du trafic de service".
4. Configurez votre CNI pour utiliser le masquage de sortie ou la traduction d'adresses réseau (NAT) pour le trafic des pods lorsqu'il quitte vos hôtes sur site. Cette fonctionnalité est activée par défaut dans Cilium. Calico a besoin que `natOutgoing` soit défini sur `true`.
5. D'autres AWS services, tels que les équilibreurs de charge d' AWS application et Amazon Managed Service for Prometheus, ne sont pas en mesure de communiquer avec les charges de travail exécutées sur des nœuds hybrides.

Accès requis pendant l'installation et la mise à niveau des nœuds hybrides

Vous devez avoir accès aux domaines suivants pendant le processus d'installation où vous installez les dépendances des nœuds hybrides sur vos hôtes. Ce processus peut être effectué une seule fois lors de la création des images de votre système d'exploitation ou sur chaque hôte au moment de l'exécution. Cela inclut l'installation initiale et la mise à niveau de la version Kubernetes de vos nœuds hybrides.

Certains paquets sont installés à l'aide du gestionnaire de paquets par défaut du système d'exploitation. Pour AL2 023 et RHEL, la `yum` commande est utilisée pour installer `containerd`, `ca-certificates`, `iptables` et. `amazon-ssm-agent` Pour Ubuntu, `apt` est utilisé pour installer `containerd`, `ca-certificates`, et `iptables`, et `snap` sert à installer `amazon-ssm-agent`.

Composant	URL	Protocole	Port
Artefacts du nœud EKS (S3)	https://hybrid-assets.eks.amazonaws.com	HTTPS	443
Points de terminaison du service EKS	https://eks.*region*.amazonaws.com	HTTPS	443
Points de terminaison du service ECR	https://api.ecr.*region*.amazonaws.com	HTTPS	443
Points de terminaison EKS ECR	Voir the section called "Affichage des registres d'images Amazon" pour les points terminaux régionaux.	HTTPS	443
Point de terminaison binaire SSM [1]	https://amazon-ssm - *region* .s3.*region*.amazonaws.com	HTTPS	443

Composant	URL	Protocole	Port
Point de terminaison du service SSM [1]	https://ssm.*region*.amazonaws.com	HTTPS	443
Point de terminaison binaire IAM Anywhere [2]	https://rolesanywhere.amazonaws.com	HTTPS	443
Point de terminaison du service IAM Anywhere [2]	https://rolesanywhere.*region*.amazonaws.com	HTTPS	443
Points de terminaison du gestionnaire de paquets du système d'exploitation	Les points de terminaison du référentiel de paquets sont spécifiques au système d'exploitation et peuvent varier selon la région géographique.	HTTPS	443

Note

[1] L'accès aux points de terminaison AWS SSM n'est requis que si vous utilisez des activations hybrides AWS SSM pour votre fournisseur d'informations d'identification IAM sur site.

[2] L'accès aux points de terminaison AWS IAM n'est requis que si vous utilisez AWS IAM Roles Anywhere comme fournisseur d'informations d'identification IAM sur site.

Accès requis pour les opérations de cluster en cours

L'accès réseau suivant pour votre pare-feu local est nécessaire pour le fonctionnement continu du cluster.

> **Important**
>
> En fonction du CNI que vous avez choisi, vous devez configurer des règles d'accès réseau supplémentaires pour les ports CNI. Pour plus d'informations, consultez la documentation Cilium et la documentation Calico.

Type	Protocole	Direction	Port	Source	Destination	Usage
HTTPS	TCP	Sortant	443	CIDR des nœuds distants	Cluster EKS IPs [1]	kubelet vers le serveur API Kubernetes
HTTPS	TCP	Sortant	443	CIDR de pod distant(s)	Cluster EKS IPs [1]	Pod vers serveur API Kubernetes
HTTPS	TCP	Sortant	443	CIDR des nœuds distants	Point de terminaison du service SSM	Actualisation des informations d'identification des activations hybrides SSM et pulsations SSM toutes les 5 minutes
HTTPS	TCP	Sortant	443	CIDR des nœuds distants	Point de terminaison du	Actualisation des informati

Type	Protocole	Direction	Port	Source	Destination	Usage
					service IAM Anywhere	ons d'identification Rôles Anywhere IAM
HTTPS	TCP	Sortant	443	CIDR de pod distant(s)	Point de terminaison régional STS	Pod vers point de terminaison STS, requis uniquement pour IRSA
HTTPS	TCP	Sortant	443	CIDR des nœuds distants	Point de terminaison du service d'authentification Amazon EKS	Nœud vers le point de terminaison d'authentification Amazon EKS, requis uniquement pour l'identité du pod Amazon EKS
HTTPS	TCP	Entrant	10250	Cluster EKS IPs [1]	CIDR des nœuds distants	Serveur API Kubernetes vers kubelet

Type	Protocole	Direction	Port	Source	Destination	Usage
HTTPS	TCP	Entrant	Ports Webhook	Cluster EKS IPs [1]	CIDR de pod distant(s)	Serveur API Kubernetes vers webhooks
HTTPS	TCP, UDP	Entrant, sortant	53	CIDR de pod distant(s)	CIDR de pod distant(s)	Pod vers CoreDNS. Si vous exécutez au moins un réplica de CoreDNS dans le cloud, vous devez autoriser le trafic DNS vers le VPC où CoreDNS est exécuté.
Défini par l'utilisateur	Défini par l'utilisateur	Entrant, sortant	Ports d'applications	CIDR de pod distant(s)	CIDR de pod distant(s)	Pod à pod

Note

[1] Celui IPs du cluster EKS. Consultez la section suivante sur les interfaces réseau Elastic Amazon EKS.

Interfaces réseau Amazon EKS

Amazon EKS associe des interfaces réseau aux sous-réseaux du VPC que vous transmettez lors de la création du cluster afin de permettre la communication entre le plan de contrôle EKS et votre VPC. Les interfaces réseau créées par Amazon EKS se trouvent après la création du cluster dans la EC2 console Amazon ou à l'aide de la AWS CLI. Les interfaces réseau d'origine sont supprimées et de nouvelles interfaces réseau sont créées lorsque des modifications sont appliquées à votre cluster EKS, telles que les mises à niveau de version Kubernetes. Vous pouvez restreindre la plage d'adresses IP des interfaces réseau Amazon EKS en utilisant des tailles de sous-réseaux limitées pour les sous-réseaux que vous transmettez lors de la création du cluster, ce qui facilite la configuration de votre pare-feu sur site afin de permettre la inbound/outbound connectivité à cet ensemble connu et restreint de. IPs Pour contrôler les sous-réseaux dans lesquels les interfaces réseau sont créées, vous pouvez limiter le nombre de sous-réseaux que vous spécifiez lors de la création d'un cluster ou mettre à jour les sous-réseaux après avoir créé le cluster.

Les interfaces réseau fournies par Amazon EKS ont une description au format `Amazon EKS` *`your-cluster-name`* . Consultez l'exemple ci-dessous pour une commande AWS CLI que vous pouvez utiliser pour trouver les adresses IP des interfaces réseau approvisionnées par Amazon EKS. Remplacez par l'ID du VPC que vous avez transmis lors de la création du cluster par `VPC_ID`.

```
aws ec2 describe-network-interfaces \
--query 'NetworkInterfaces[?(VpcId == VPC_ID && contains(Description,Amazon
 EKS))].PrivateIpAddress'
```

AWS Configuration du VPC et du sous-réseau

Les exigences existantes en matière de VPC et de sous-réseau pour Amazon EKS s'appliquent aux clusters avec des nœuds hybrides. En outre, votre VPC CIDR ne peut pas chevaucher votre nœud et votre pod locaux. CIDRs Vous devez configurer les itinéraires dans votre table de routage VPC pour votre nœud local et éventuellement pour votre pod. CIDRs Ces routes doivent être configurées pour acheminer le trafic vers la passerelle que vous utilisez pour votre connectivité réseau hybride, qui est généralement une passerelle privée virtuelle (VGW) ou une passerelle de transit (TGW). Si vous utilisez TGW ou VGW pour connecter votre VPC à votre environnement sur site, vous devez créer une connexion TGW ou VGW pour votre VPC. Votre VPC doit prend en charge le nom d'hôte DNS et la résolution DNS.

Les étapes suivantes utilisent la AWS CLI. Vous pouvez également créer ces ressources dans AWS Management Console ou avec d'autres interfaces telles que AWS CloudFormation AWS CDK ou Terraform.

Étape 1 : créer un VPC

1. Exécutez la commande suivante pour créer un VPC. Remplacer `VPC_CIDR` par une plage CIDR RFC-1918 (privée) `IPv4` ou non RFC-1918 (publique) (par exemple `10.0.0.0/16`). Remarque : la résolution DNS, qui est une exigence EKS, est activée par défaut pour le VPC.

```
aws ec2 create-vpc --cidr-block VPC_CIDR
```

2. Activez les noms d'hôte DNS pour votre VPC. Remarque : la résolution DNS est activée par défaut pour le VPC. Remplacez `VPC_ID` par l'ID du VPC que vous avez créé à l'étape précédente.

```
aws ec2 modify-vpc-attribute --vpc-id VPC_ID --enable-dns-hostnames
```

Étape 2 : créer des sous-réseaux

Créez au moins 2 sous-réseaux. Amazon EKS utilise ces sous-réseaux pour les interfaces réseau du cluster. Pour plus d'informations, consultez la section Exigences et considérations relatives aux sous-réseaux.

1. Vous pouvez trouver les zones de disponibilité d'une AWS région à l'aide de la commande suivante. Remplacez `us-west-2` par votre région.

```
aws ec2 describe-availability-zones \
     --query 'AvailabilityZones[?(RegionName == us-west-2)].ZoneName'
```

2. Créez un sous-réseau. Remplacez `VPC_ID` par l'ID du VPC. Remplacez `SUBNET_CIDR` par le bloc CIDR de votre sous-réseau (par exemple 10.0.1.0/24). Remplacez `AZ` par la zone de disponibilité dans laquelle le sous-réseau sera créé (par exemple us-west-2a). Les sous-réseaux que vous créez doivent se trouver dans au moins deux zones de disponibilité différentes.

```
aws ec2 create-subnet \
    --vpc-id VPC_ID \
    --cidr-block SUBNET_CIDR \
    --availability-zone AZ
```

(Facultatif) Étape 3 : associer un VPC à la passerelle privée virtuelle Amazon VPC Transit Gateway (TGW) ou AWS Direct Connect (VGW)

Si vous utilisez un TGW ou un VGW, connectez votre VPC au TGW ou au VGW. Pour plus d'informations, consultez la section Pièces jointes Amazon VPC dans les passerelles de transit Amazon VPC ou Associations de passerelles privées virtuelles Direct Connect AWS.

Passerelle de transit

Exécutez la commande suivante pour connecter une passerelle Transit Gateway. Remplacez `VPC_ID` par l'ID du VPC. Remplacez `SUBNET_ID1` et `SUBNET_ID2` par IDs les sous-réseaux que vous avez créés à l'étape précédente. Remplacez `TGW_ID` par l'ID de votre TGW.

```
aws ec2 create-transit-gateway-vpc-attachment \
    --vpc-id VPC_ID \
    --subnet-ids SUBNET_ID1 SUBNET_ID2 \
    --transit-gateway-id TGW_ID
```

Passerelle privée virtuelle

Exécutez la commande suivante pour connecter une passerelle Transit Gateway. Remplacez `VPN_ID` par l'ID de votre VGW. Remplacez `VPC_ID` par l'ID du VPC.

```
aws ec2 attach-vpn-gateway \
    --vpn-gateway-id VPN_ID \
    --vpc-id VPC_ID
```

(Facultatif) Étape 4 : créer une table de routage

Vous pouvez modifier la table de routage principale pour le VPC ou créer une table de routage personnalisée. Les étapes suivantes créent une table de routage personnalisée avec les itinéraires vers le nœud et le pod CIDRs locaux. Pour plus d'informations, consultez Tables de routage de sous-réseau. Remplacez `VPC_ID` par l'ID du VPC.

```
aws ec2 create-route-table --vpc-id VPC_ID
```

Étape 5 : créer des routes pour les nœuds et les pods sur site

Créez des routes dans la table de routage pour chacun de vos nœuds distants sur site. Vous pouvez modifier la table de routage principale pour le VPC ou utiliser la table de routage personnalisée que vous avez créée à l'étape précédente.

Les exemples ci-dessous montrent comment créer des itinéraires pour votre nœud et votre pod CIDRs locaux. Dans les exemples, une passerelle de transit (TGW) est utilisée pour connecter le VPC à l'environnement sur site. Si vous disposez de plusieurs nœuds et pods sur site CIDRs, répétez les étapes pour chaque CIDR.

- Si vous utilisez une passerelle Internet ou une passerelle privée virtuelle (VGW), remplacez `--transit-gateway-id` par `--gateway-id`.
- Remplacez `RT_ID` par l'ID de la table de routage que vous avez créée à l'étape précédente.
- Remplacez `REMOTE_NODE_CIDR` par la plage CIDR que vous utiliserez pour vos nœuds hybrides.
- Remplacez `REMOTE_POD_CIDR` par la plage CIDR que vous utiliserez pour les pods s'exécutant sur des nœuds hybrides. La plage CIDR du pod correspond à la configuration CNI (Container Networking Interface), qui utilise le plus souvent un réseau superposé sur site. Pour de plus amples informations, veuillez consulter the section called "Configurer CNI".
- Remplacez `TGW_ID` par l'ID de votre TGW.

Réseau de nœuds distants

```
aws ec2 create-route \
    --route-table-id RT_ID \
    --destination-cidr-block REMOTE_NODE_CIDR \
    --transit-gateway-id TGW_ID
```

Réseau de pods distants

```
aws ec2 create-route \
    --route-table-id RT_ID \
    --destination-cidr-block REMOTE_POD_CIDR \
    --transit-gateway-id TGW_ID
```

(Facultatif) Étape 6 : associer les sous-réseaux à la table de routage

Si vous avez créé une table de routage personnalisée à l'étape précédente, associez chacun des sous-réseaux que vous avez créés à l'étape précédente à votre table de routage personnalisée. Si vous modifiez la table de routage principale du VPC, les sous-réseaux sont automatiquement associés à la table de routage principale du VPC et vous pouvez ignorer cette étape.

Exécutez la commande suivante pour chacun des sous-réseaux que vous avez créés lors des étapes précédentes. Remplacer `RT_ID` par la table de routage que vous avez créée à l'étape précédente. Remplacer `SUBNET_ID` par l'ID d'un sous-réseau.

```
aws ec2 associate-route-table --route-table-id RT_ID --subnet-id SUBNET_ID
```

Configuration du groupe de sécurité du cluster

L'accès suivant pour votre groupe de sécurité de cluster EKS est nécessaire pour le fonctionnement continu du cluster. Amazon EKS crée automatiquement les règles de groupe de sécurité entrantes requises pour les nœuds hybrides lorsque vous créez ou mettez à jour votre cluster avec des réseaux de nœuds distants et de pods configurés. Étant donné que les groupes de sécurité autorisent par défaut tout le trafic sortant, Amazon EKS ne modifie pas automatiquement les règles sortantes du groupe de sécurité du cluster pour les nœuds hybrides. Si vous souhaitez personnaliser le groupe de sécurité du cluster, vous pouvez limiter le trafic aux règles indiquées dans le tableau suivant.

Type	Protocole	Direction	Port	Source	Destination	Usage
HTTPS	TCP	Entrant	443	CIDR des nœuds distants	N/A	Kubelet vers le serveur API Kubernetes
HTTPS	TCP	Entrant	443	CIDR de pod distant(s)	N/A	Pods nécessitant un accès au serveur API K8s lorsque

Type	Protocole	Direction	Port	Source	Destination	Usage
						le CNI n'utilise pas NAT pour le trafic des pods.
HTTPS	TCP	Sortant	10250	N/A	CIDR des nœuds distants	Serveur API Kubernetes vers Kubelet
HTTPS	TCP	Sortant	Ports Webhook	N/A	CIDR de pod distant(s)	Serveur API Kubernetes vers webhook (si vous exécutez des webhooks sur des nœuds hybrides)

Important

Limites des règles relatives aux groupes de EC2 sécurité : les groupes de sécurité Amazon ont un maximum de 60 règles entrantes par défaut. Les règles entrantes du groupe de sécurité peuvent ne pas s'appliquer si le groupe de sécurité de votre cluster approche cette limite. Dans ce cas, il peut être nécessaire d'ajouter manuellement les règles entrantes manquantes.

Responsabilité du nettoyage CIDR : si vous supprimez des réseaux de nœuds distants ou de pods des clusters EKS, EKS ne supprime pas automatiquement les règles de groupe de

sécurité correspondantes. Vous êtes responsable de la suppression manuelle des réseaux de nœuds distants ou de pods inutilisés de vos règles de groupe de sécurité.

Pour plus d'informations sur le groupe de sécurité de cluster créé par Amazon EKS, consultez the section called "Exigences de groupe de sécurité".

(Facultatif) Configuration manuelle du groupe de sécurité

Si vous devez créer des groupes de sécurité supplémentaires ou modifier les règles créées automatiquement, vous pouvez utiliser les commandes suivantes comme référence. Par défaut, la commande ci-dessous crée un groupe de sécurité qui autorise tous les accès sortants. Vous pouvez restreindre l'accès sortant afin qu'il n'inclue que les règles ci-dessus. Si vous envisagez de limiter les règles sortantes, nous vous recommandons de tester minutieusement toutes vos applications et la connectivité des pods avant d'appliquer vos règles modifiées à un cluster de production.

- Dans la première commande, remplacez `SG_NAME` par le nom de votre groupe de sécurité.
- Dans la première commande, remplacez `VPC_ID` par l'ID du VPC que vous avez créé à l'étape précédente.
- Dans la deuxième commande, remplacez `SG_ID` par l'ID du groupe de sécurité que vous avez créé dans la première commande.
- Dans la deuxième commande, remplacez `REMOTE_NODE_CIDR` et `REMOTE_POD_CIDR` par les valeurs correspondant à vos nœuds hybrides et à votre réseau local.

```
aws ec2 create-security-group \
    --group-name SG_NAME \
    --description "security group for hybrid nodes" \
    --vpc-id VPC_ID
```

```
aws ec2 authorize-security-group-ingress \
    --group-id SG_ID \
    --ip-permissions '[{"IpProtocol": "tcp", "FromPort": 443, "ToPort": 443,
 "IpRanges": [{"CidrIp": "REMOTE_NODE_CIDR"}, {"CidrIp": "REMOTE_POD_CIDR"}]}]'
```

Préparer le système d'exploitation pour les nœuds hybrides

Bottlerocket, Amazon Linux 2023 (AL2023), Ubuntu et RHEL sont validés en permanence pour être utilisés comme système d'exploitation de nœuds pour les nœuds hybrides. Bottlerocket est uniquement pris en charge dans les environnements AWS vSphere VMware . AL2Le 023 n'est pas couvert par les plans de AWS Support lorsqu'il est exécuté en dehors d'Amazon EC2. AL2Le 023 ne peut être utilisé que dans des environnements virtualisés sur site. Consultez le guide de l'utilisateur Amazon Linux 2023 pour plus d'informations. AWS prend en charge l'intégration des nœuds hybrides avec les systèmes d'exploitation Ubuntu et RHEL, mais ne fournit pas de support pour le système d'exploitation lui-même.

Vous êtes responsable de la mise à disposition et de la gestion du système d'exploitation. Lorsque vous testez des nœuds hybrides pour la première fois, il est plus simple d'exécuter la CLI des nœuds hybrides Amazon EKS (`nodeadm`) sur un hôte déjà provisionné. Pour les déploiements en production, nous vous recommandons d'inclure `nodeadm` dans vos images de système d'exploitation une configuration permettant leur exécution en tant que service systemd afin de joindre automatiquement les hôtes aux clusters Amazon EKS au démarrage de l'hôte. Si vous utilisez Bottlerocket comme système d'exploitation de nœud sur vSphere, vous n'avez pas besoin d'utiliser `nodeadm`, car Bottlerocket contient déjà les dépendances requises pour les nœuds hybrides et se connectera automatiquement au cluster que vous configurez au démarrage de l'hôte.

Compatibilité des versions

Le tableau ci-dessous présente les versions de système d'exploitation compatibles et validées pour être utilisées comme système d'exploitation de nœud pour les nœuds hybrides. Si vous utilisez d'autres variantes ou versions de système d'exploitation qui ne sont pas incluses dans ce tableau, la compatibilité des nœuds hybrides avec la variante ou la version de votre système d'exploitation n'est pas couverte par le AWS Support. Les nœuds hybrides sont indépendants de l'infrastructure sous-jacente et prennent en charge les architectures x86 et ARM.

Système d'exploitation	Versions
Amazon Linux	Amazon Linux 2023 (AL2023)
Flacon Rocket	Variantes v1.37.0 et supérieures exécutant Kubernetes v1.28 et VMware versions ultérieures
Ubuntu	Ubuntu 20.04, Ubuntu 22.04, Ubuntu 24.04

Système d'exploitation	Versions
Utilisation de Red Hat Enterprise Linux	RHEL 8, RHEL 9

Considérations relatives au système d'exploitation

Général

- La CLI des nœuds hybrides Amazon EKS (`nodeadm`) peut être utilisée pour simplifier l'installation et la configuration des composants et des dépendances des nœuds hybrides. Vous pouvez exécuter le processus `nodeadm install` pendant la création des pipelines d'image de votre système d'exploitation ou au moment de l'exécution sur chaque hôte sur site. Pour plus d'informations sur les composants installés `nodeadm`, consultez le the section called "Nœuds hybrides nodeadm".
- Si vous utilisez un proxy dans votre environnement sur site pour accéder à Internet, une configuration supplémentaire du système d'exploitation est nécessaire pour les processus d'installation et de mise à niveau afin de configurer votre gestionnaire de paquets pour qu'il utilise le proxy. Pour obtenir des instructions, consultez the section called "Configurer le proxy".

Flacon Rocket

- Les étapes et les outils nécessaires pour connecter un nœud Bottlerocket diffèrent de ceux utilisés pour les autres systèmes d'exploitation et sont décrits séparément dans the section called "Connectez des nœuds hybrides avec Bottlerocket", plutôt que dans the section called "Connecter les nœuds hybrides".
- Les étapes pour Bottlerocket n'utilisent pas la CLI des nœuds hybrides, `nodeadm`.
- Seules les VMware variantes de la version v1.37.0 et supérieures de Bottlerocket sont prises en charge avec les nœuds hybrides EKS. VMware des variantes de Bottlerocket sont disponibles pour les versions v1.28 et supérieures de Kubernetes. Les autres variantes de Bottlerocket ne sont pas prises en charge en tant que système d'exploitation des nœuds hybrides. REMARQUE : les VMware variantes de Bottlerocket ne sont disponibles que pour l'architecture x86_64.

Containerd

- Containerd est l'exécution de conteneur Kubernetes standard et est une dépendance pour les nœuds hybrides, ainsi que pour tous les types de calcul de nœuds Amazon EKS. La CLI des

nœuds hybrides Amazon EKS (`nodeadm`) tente d'installer containerd pendant le processus `nodeadm install`. Vous pouvez configurer l'installation de containerd lors de l'exécution `nodeadm install` à l'aide de l'option de ligne de commande `--containerd-source`. Les options valides sont `none`, `distro` et `docker`. Si vous utilisez RHEL, `distro` n'est pas une option valide. Vous pouvez soit configurer `nodeadm` pour installer la version de containerd à partir des référentiels Docker, soit installer containerd manuellement. Lorsque vous utilisez AL2 023 ou Ubuntu, installez `nodeadm` par défaut containerd à partir de la distribution du système d'exploitation. Si vous ne souhaitez pas que nodeadm installe containerd, utilisez l'option `--containerd-source none`.

Ubuntu

- Si vous utilisez Ubuntu 24.04, vous devrez peut-être mettre à jour votre version de containerd ou modifier AppArmor votre configuration pour adopter un correctif permettant aux pods de s'arrêter correctement, voir Ubuntu #2065423. Un redémarrage est nécessaire pour appliquer les modifications au AppArmor profil. La dernière version d'Ubuntu 24.04 dispose d'une version mise à jour de containerd dans son gestionnaire de paquets avec le correctif (containerd version 1.7.19+).

ARM

- Si vous utilisez du matériel ARM, un processeur compatible ARMv8 .2 avec l'extension de cryptographie (ARMv8.2+crypto) est requis pour exécuter les versions 1.31 et supérieures du module complémentaire EKS kube-proxy. Tous les systèmes Raspberry Pi antérieurs au Raspberry Pi 5, ainsi que les processeurs basés sur Cortex-A72, ne répondent pas à cette exigence. Comme solution de contournement, vous pouvez continuer à utiliser la version 1.30 du module complémentaire EKS kube-proxy jusqu'à la fin de son support étendu en juillet 2026 (voir le calendrier des versions Kubernetes), ou utiliser une image kube-proxy personnalisée provenant de l'amont.
- Le message d'erreur suivant dans le journal kube-proxy indique cette incompatibilité :

```
Fatal glibc error: This version of Amazon Linux requires a newer ARM64 processor
 compliant with at least ARM architecture 8.2-a with Cryptographic extensions. On EC2
 this is Graviton 2 or later.
```

Création d'images du système d'exploitation

Amazon EKS fournit des exemples de modèles Packer que vous pouvez utiliser pour créer des images de système d'exploitation qui incluent `nodeadm` et configurent Packer pour qu'il s'exécute au démarrage de l'hôte. Ce processus est recommandé pour éviter de tirer individuellement les dépendances des nœuds hybrides sur chaque hôte et pour automatiser le processus de démarrage des nœuds hybrides. Vous pouvez utiliser les modèles Packer fournis à titre d'exemple avec une image ISO Ubuntu 22.04, Ubuntu 24.04, RHEL 8 ou RHEL 9 et générer des images aux formats suivants : OVA, Qcow2 ou raw.

Conditions préalables

Avant d'utiliser les modèles Packer fournis à titre d'exemple, vous devez avoir installé les éléments suivants sur la machine à partir de laquelle vous exécutez Packer.

- Packer version 1.11.0 ou supérieure. Pour obtenir des instructions sur l'installation de Packer, consultez la section Installer Packer dans la documentation Packer.
- En cas de compilation OVAs, VMware vSphere plugin 1.4.0 ou version ultérieure
- Si vous créez `Qcow2` ou des images brutes, utilisez le plug-in QEMU version 1.x.

Définir les variables d'environnement

Avant d'exécuter la compilation Packer, définissez les variables d'environnement suivantes sur la machine à partir de laquelle vous exécutez Packer.

Général

Les variables d'environnement suivantes doivent être définies pour créer des images avec tous les systèmes d'exploitation et formats de sortie.

Variable d'environnement	Type	Description
PKR_SSH_PASSWORD	String	Packer utilise les variables `ssh_username` et `ssh_password` pour se connecter via SSH à la machine créée lors du provisionnement. Cela doit correspondre aux mots de

Variable d'environnement	Type	Description
		passe utilisés pour créer l'utilisateur initial dans les fichiers kickstart ou user-data du système d'exploitation concerné. La valeur par défaut est définie comme « builder » ou « ubuntu » selon le système d'exploitation. Lorsque vous définissez votre mot de passe, veillez à le modifier dans le fichier `ks.cfg` ou `user-data` correspondant afin qu'il corresponde.
ISO_URL	String	URL de l'ISO à utiliser. Peut être un lien Web pour télécharger depuis un serveur ou un chemin absolu vers un fichier local
ISO_CHECKSUM	String	Somme de contrôle associée pour l'ISO fournie.
CREDENTIAL_PROVIDER	String	Fournisseur d'informations d'identification pour les nœuds hybrides. Les valeurs valides sont `ssm` (par défaut) pour les activations hybrides SSM et `iam` pour Rôles Anywhere IAM

Variable d'environnement	Type	Description
K8S_VERSION	String	Version Kubernetes pour les nœuds hybrides (par exemple `1.31`). Pour connaître les versions Kubernetes prises en charge, consultez la page Versions Amazon EKS prises en charge.
NODEADM_ARCH	String	Architecture pour `nodeadm install`. Sélectionnez `amd` ou `arm`.

RHEL

Si vous utilisez RHEL, les variables d'environnement suivantes doivent être définies.

Variable d'environnement	Type	Description
RH_USERNAME	String	Nom d'utilisateur du gestionnaire d'abonnement RHEL
RH_PASSWORD	String	Mot de passe du gestionnaire d'abonnements RHEL
RHEL_VERSION	String	Version Rhel iso utilisée. Les valeurs valides sont 8 ou 9.

Ubuntu

Aucune variable d'environnement spécifique à Ubuntu n'est requise.

vSphere

Si vous créez un VMware vSphere OVA, les variables d'environnement suivantes doivent être définies.

Variable d'environnement	Type	Description
VSPHERE_SERVER	String	Adresse du serveur vSphere
VSPHERE_USER	String	Nom d'utilisateur vSphere
VSPHERE_PASSWORD	String	Mot de passe vSphere
VSPHERE_DATACENTER	String	Nom du centre de données vSphere
VSPHERE_CLUSTER	String	Nom du cluster vSphere
VSPHERE_DATASTORE	String	Nom de l'entrepôt de données vSphere
VSPHERE_NETWORK	String	Nom du réseau vSphere
VSPHERE_OUTPUT_FOLDER	String	Dossier de sortie vSphere pour les modèles

QEMU

Variable d'environnement	Type	Description
PACKER_OUTPUT_FORMAT	String	Format de sortie pour le générateur QEMU. Les valeurs valides sont `qcow2` et `raw`.

Valider le modèle

Avant d'exécuter votre build, validez votre modèle à l'aide de la commande suivante après avoir défini vos variables d'environnement. Remplacez `template.pkr.hcl` si vous utilisez un nom différent pour votre modèle.

```
packer validate template.pkr.hcl
```

Créer des images

Créez vos images à l'aide des commandes suivantes et utilisez l'indicateur `-only` pour spécifier la cible et le système d'exploitation de vos images. Remplacez `template.pkr.hcl` si vous utilisez un autre nom pour votre modèle.

vSphere OVAs

> **Note**
>
> Si vous utilisez RHEL avec vSphere, vous devez convertir les fichiers Kickstart en une image OEMDRV et la transmettre sous forme d'ISO pour démarrer à partir de celle-ci. Pour plus d'informations, consultez le fichier Packer Readme dans le référentiel EKS Hybrid Nodes GitHub .

Ubuntu 22.04 OVA

```
packer build -only=general-build.vsphere-iso.ubuntu22 template.pkr.hcl
```

Ubuntu 24.04 OVA

```
packer build -only=general-build.vsphere-iso.ubuntu24 template.pkr.hcl
```

RHEL 8 OVA

```
packer build -only=general-build.vsphere-iso.rhel8 template.pkr.hcl
```

RHEL 9 OVA

```
packer build -only=general-build.vsphere-iso.rhel9 template.pkr.hcl
```

QEMU

> **Note**
>
> Si vous créez une image pour un processeur hôte spécifique qui ne correspond pas à votre hôte de compilation, consultez la documentation QEMU pour trouver le nom qui correspond

à votre processeur hôte et utilisez l'indicateur `-cpu` avec le nom du processeur hôte lorsque vous exécutez les commandes suivantes.

Ubuntu 22.04 Qcow2 / Raw

```
packer build -only=general-build.qemu.ubuntu22 template.pkr.hcl
```

Ubuntu 24.04 Qcow2 / Raw

```
packer build -only=general-build.qemu.ubuntu24 template.pkr.hcl
```

RHEL 8 Qcow2 / Raw

```
packer build -only=general-build.qemu.rhel8 template.pkr.hcl
```

RHEL 9 Qcow2 / Raw

```
packer build -only=general-build.qemu.rhel9 template.pkr.hcl
```

Transmettre la configuration nodeadm via les données utilisateur

Vous pouvez transmettre la configuration pour `nodeadm` dans vos données utilisateur via cloud-init afin de configurer et de connecter automatiquement les nœuds hybrides à votre cluster EKS au démarrage de l'hôte. Vous trouverez ci-dessous un exemple de la manière d'y parvenir lorsque vous utilisez VMware vSphere comme infrastructure pour vos nœuds hybrides.

1. Installez la `govc` CLI en suivant les instructions du fichier readme govc on. GitHub
2. Après avoir exécuté la compilation Packer dans la section précédente et provisionné votre modèle, vous pouvez cloner votre modèle pour créer plusieurs nœuds différents à l'aide de la commande suivante. Vous devez cloner le modèle pour chaque nouvelle machine virtuelle que vous créez et qui sera utilisée pour les nœuds hybrides. Remplacez les variables dans la commande ci-dessous par les valeurs correspondant à votre environnement. La `VM_NAME` commande ci-dessous est utilisée comme vôtre `NODE_NAME` lorsque vous injectez les noms de votre VMs via votre `metadata.yaml` fichier.

```
govc vm.clone -vm "/PATH/TO/TEMPLATE" -ds="YOUR_DATASTORE" \
```

```
    -on=false -template=false -folder=/FOLDER/TO/SAVE/VM "VM_NAME"
```

3. Après avoir cloné le modèle pour chacun de vos nouveaux modèles VMs, créez un `userdata.yaml` et `metadata.yaml` pour votre VMs. VMs Vous pouvez les partager `userdata.yaml` `metadata.yaml` et vous les remplirez par machine virtuelle en suivant les étapes ci-dessous. La configuration `nodeadm` est créée et définie dans la section `write_files` de votre `userdata.yaml`. L'exemple ci-dessous utilise les activations hybrides AWS SSM comme fournisseur d'informations d'identification sur site pour les nœuds hybrides. Pour plus d'informations sur la configuration `nodeadm`, consultez le the section called "Nœuds hybrides nodeadm".

   userdata.yaml :

```
#cloud-config
users:
  - name: # username for login. Use 'builder' for RHEL or 'ubuntu' for Ubuntu.
    passwd: # password to login. Default is 'builder' for RHEL.
    groups: [adm, cdrom, dip, plugdev, lxd, sudo]
    lock-passwd: false
    sudo: ALL=(ALL) NOPASSWD:ALL
    shell: /bin/bash

write_files:
  - path: /usr/local/bin/nodeConfig.yaml
    permissions: '0644'
    content: |
      apiVersion: node.eks.aws/v1alpha1
      kind: NodeConfig
      spec:
          cluster:
              name: # Cluster Name
              region: # AWS region
          hybrid:
              ssm:
                  activationCode: # Your ssm activation code
                  activationId: # Your ssm activation id

runcmd:
  - /usr/local/bin/nodeadm init -c file:///usr/local/bin/nodeConfig.yaml >> /var/log/
nodeadm-init.log 2>&1
```

   métadonnées.yaml :

Créez un `metadata.yaml` pour votre environnement. Conservez le format variable `"$NODE_NAME"` dans le fichier, car il sera rempli avec des valeurs lors d'une étape ultérieure.

```
instance-id: "$NODE_NAME"
local-hostname: "$NODE_NAME"
network:
  version: 2
  ethernets:
    nics:
      match:
        name: ens*
      dhcp4: yes
```

4. Ajoutez les fichiers `userdata.yaml` et `metadata.yaml` sous forme de chaînes `gzip+base64` à l'aide des commandes suivantes. Les commandes suivantes doivent être exécutées pour chacune des commandes VMs que vous créez. Remplacez `VM_NAME` par le nom de la machine virtuelle que vous mettez à jour.

```
export NODE_NAME="VM_NAME"
export USER_DATA=$(gzip -c9 <userdata.yaml | base64)

govc vm.change -dc="YOUR_DATASTORE" -vm "$NODE_NAME" -e
 guestinfo.userdata="${USER_DATA}"
govc vm.change -dc="YOUR_DATASTORE" -vm "$NODE_NAME" -e
 guestinfo.userdata.encoding=gzip+base64

envsubst '$NODE_NAME' < metadata.yaml > metadata.yaml.tmp
export METADATA=$(gzip -c9 <metadata.yaml.tmp | base64)

govc vm.change -dc="YOUR_DATASTORE" -vm "$NODE_NAME" -e
 guestinfo.metadata="${METADATA}"
govc vm.change -dc="YOUR_DATASTORE" -vm "$NODE_NAME" -e
 guestinfo.metadata.encoding=gzip+base64
```

5. Allumez votre nouveau VMs, qui devrait se connecter automatiquement au cluster EKS que vous avez configuré.

```
govc vm.power -on "${NODE_NAME}"
```

Préparer les informations d'identification pour les nœuds hybrides

Les nœuds hybrides Amazon EKS utilisent des informations d'identification IAM temporaires fournies par des activations hybrides AWS SSM ou par AWS IAM Roles Anywhere pour s'authentifier auprès du cluster Amazon EKS. Vous devez utiliser des activations hybrides AWS SSM ou des rôles AWS IAM Anywhere avec la CLI Amazon EKS Hybrid Nodes (). `nodeadm` Vous ne devez pas utiliser à la fois les activations hybrides AWS SSM et les rôles AWS IAM Anywhere. Nous vous recommandons d'utiliser les activations hybrides AWS SSM si vous ne possédez pas d'infrastructure à clé publique (PKI) existante dotée d'une autorité de certification (CA) et de certificats pour vos environnements sur site. Si vous disposez d'une infrastructure PKI et de certificats sur site, utilisez AWS IAM Roles Anywhere.

Rôle IAM des nœuds hybrides

Avant de pouvoir connecter des nœuds hybrides à votre cluster Amazon EKS, vous devez créer un rôle IAM qui sera utilisé avec les activations hybrides AWS SSM ou AWS IAM Roles Anywhere pour les informations d'identification de vos nœuds hybrides. Après la création du cluster, vous utiliserez ce rôle avec une entrée d'accès Amazon EKS ou `aws-auth` ConfigMap une entrée pour associer le rôle IAM au contrôle d'accès basé sur les rôles (RBAC) de Kubernetes. Pour plus d'informations sur l'association du rôle IAM des nœuds hybrides avec Kubernetes RBAC, consultez the section called "Préparation de l'accès aux clusters".

Le rôle IAM des nœuds hybrides doit disposer des autorisations suivantes.

- Autorisations `nodeadm` permettant d'utiliser l'`eks:DescribeCluster`action pour recueillir des informations sur le cluster auquel vous souhaitez connecter des nœuds hybrides. Si vous n'activez pas l'`eks:DescribeCluster`action, vous devez transmettre votre point de terminaison d'API Kubernetes, votre bundle CA de cluster et votre IPv4 CIDR de service dans la configuration du nœud que vous transmettez à la commande. `nodeadm init`
- Autorisations `nodeadm` permettant d'utiliser l'`eks:ListAccessEntries`action pour répertorier les entrées d'accès du cluster auquel vous souhaitez connecter des nœuds hybrides. Si vous n'activez pas l'`eks:ListAccessEntries`action, vous devez passer le `--skip cluster-access-validation` drapeau lorsque vous exécutez la `nodeadm init` commande.
- Autorisations permettant au kubelet d'utiliser des images de conteneur issues d'Amazon Elastic Container Registry (Amazon ECR), conformément à la politique d'Amazon. EC2 ContainerRegistryPullOnly

- Si vous utilisez AWS SSM, autorisations `nodeadm init` pour utiliser les activations hybrides AWS SSM telles que définies dans la politique. https://docs.aws.amazon.com/aws-managed-policy/latest/reference/AmazonSSMManagedInstanceCore.html
- Si vous utilisez AWS SSM, autorisations d'utilisation de l'`ssm:DeregisterManagedInstance`action et de l'`ssm:DescribeInstanceInformation`action pour `nodeadm uninstall` désenregistrer les instances.
- (Facultatif) Autorisations permettant à l'agent d'identité du pod Amazon EKS d'utiliser l'action `eks-auth:AssumeRoleForPodIdentity` pour récupérer les informations d'identification pour les pods.

Configuration des activations hybrides AWS SSM

Avant de configurer les activations hybrides AWS SSM, vous devez avoir créé et configuré un rôle IAM de nœuds hybrides. Pour de plus amples informations, veuillez consulter the section called "Création du rôle IAM des nœuds hybrides". Suivez les instructions de la section Créer une activation hybride pour enregistrer des nœuds auprès de Systems Manager dans le guide de l'utilisateur de AWS Systems Manager afin de créer une activation hybride AWS SSM pour vos nœuds hybrides. Le code d'activation et l'identifiant que vous recevez sont utilisés avec `nodeadm` lorsque vous enregistrez vos hôtes en tant que nœuds hybrides avec votre cluster Amazon EKS. Vous pourrez revenir à cette étape ultérieurement après avoir créé et préparé vos clusters Amazon EKS pour les nœuds hybrides.

> **Important**
>
> Systems Manager renvoie immédiatement le code d'activation et l'ID à la console ou la fenêtre de commande, selon la méthode de création de l'activation. Copiez ces informations et stockez-les en lieu sûr. Si vous quittez la console ou fermez la fenêtre de commande, vous risquez de perdre ces informations. Si vous les perdez, vous devrez recréer une activation.

Par défaut, les activations hybrides AWS SSM sont actives pendant 24 heures. Vous pouvez également spécifier un `--expiration-date` lorsque vous créez votre activation hybride, par exemple `2024-08-01T00:00:00`. Lorsque vous utilisez AWS SSM comme fournisseur d'informations d'identification, le nom du nœud de vos nœuds hybrides n'est pas configurable et est généré automatiquement par SSM. AWS Vous pouvez consulter et gérer les instances gérées par AWS SSM dans la console AWS Systems Manager sous Fleet Manager. Vous pouvez enregistrer

jusqu'à 1 000 nœuds hybrides standard par compte et par AWS région sans frais supplémentaires. Toutefois, pour enregistrer plus de 1 000 nœuds hybrides, vous avez besoin d'activer le niveau d'instances avancées. L'utilisation du niveau Advanced Instances entraîne des frais qui ne sont pas inclus dans la tarification des nœuds hybrides Amazon EKS. Pour plus d'informations, consultez la section Tarifs de AWS Systems Manager.

Consultez l'exemple ci-dessous pour savoir comment créer une activation hybride AWS SSM avec votre rôle IAM Hybrid Nodes. Lorsque vous utilisez des activations hybrides AWS SSM pour les informations d'identification de vos nœuds hybrides, les noms de vos nœuds hybrides auront le même format `mi-012345678abcdefgh` et les informations d'identification temporaires fournies par AWS SSM sont valides pendant 1 heure. Vous ne pouvez pas modifier le nom du nœud ou la durée des informations d'identification lorsque vous utilisez AWS SSM comme fournisseur d'informations d'identification. Les informations d'identification temporaires sont automatiquement modifiées par AWS SSM et la rotation n'a aucun impact sur l'état de vos nœuds ou applications.

Nous vous recommandons d'utiliser une activation hybride AWS SSM par cluster EKS pour définir l'`ssm:DeregisterManagedInstance`autorisation AWS SSM du rôle IAM des nœuds hybrides afin de ne pouvoir désenregistrer que les instances associées à votre activation hybride SSM. AWS Dans l'exemple de cette page, une balise avec l'ARN du cluster EKS est utilisée, qui peut être utilisée pour mapper votre activation hybride AWS SSM au cluster EKS. Vous pouvez également utiliser votre balise et votre méthode préférées pour délimiter les autorisations AWS SSM en fonction de vos limites et exigences en matière d'autorisation. L'`REGISTRATION_LIMIT`option de la commande ci-dessous est un entier utilisé pour limiter le nombre de machines pouvant utiliser l'activation hybride AWS SSM (par exemple`10`)

```
aws ssm create-activation \
     --region AWS_REGION \
     --default-instance-name eks-hybrid-nodes \
     --description "Activation for EKS hybrid nodes" \
     --iam-role AmazonEKSHybridNodesRole \
     --tags Key=EKSClusterARN,Value=arn:aws: eks:AWS_REGION:AWS_ACCOUNT_ID:cluster/
CLUSTER_NAME \
     --registration-limit REGISTRATION_LIMIT
```

Consultez les instructions de la section Créer une activation hybride pour enregistrer des nœuds auprès de Systems Manager pour plus d'informations sur les paramètres de configuration disponibles pour les activations hybrides AWS SSM.

Configurer des rôles AWS IAM n'importe où

Suivez les instructions de la section Démarrage avec Rôles Anywhere IAM du guide de l'utilisateur Rôles Anywhere IAM pour configurer l'ancre d'approbation et le profil que vous utiliserez pour les informations d'identification IAM temporaires de votre rôle IAM des nœuds hybrides. Lorsque vous créez votre profil, vous pouvez le créer sans ajouter de rôles. Vous pouvez créer ce profil, revenir à ces étapes pour créer votre rôle IAM des nœuds hybrides, puis ajouter votre rôle à votre profil une fois celui-ci créé. Vous pouvez également AWS CloudFormation suivre les étapes décrites plus loin sur cette page pour terminer la configuration d'IAM Roles Anywhere pour les nœuds hybrides.

Lorsque vous ajoutez le rôle IAM Hybrid Nodes à votre profil, sélectionnez Accepter le nom de session du rôle personnalisé dans le panneau Nom de session du rôle personnalisé au bas de la page Modifier le profil de la console AWS IAM Roles Anywhere. Cela correspond au champ acceptRoleSessionNom de l'`CreateProfile`API. Cela vous permet de fournir un nom de nœud personnalisé pour vos nœuds hybrides dans la configuration que vous transmettez à `nodeadm` au cours du processus d'amorçage. Il est nécessaire de passer un nom de nœud personnalisé pendant le processus `nodeadm init`. Vous pouvez mettre à jour votre profil pour accepter un nom de session personnalisé après avoir créé votre profil.

Vous pouvez configurer la durée de validité des informations d'identification avec AWS IAM Roles Anywhere via le champ DurationSeconds de votre profil IAM Roles Anywhere. AWS La durée par défaut est de 1 heure avec un maximum de 12 heures. Le `MaxSessionDuration` paramètre de votre rôle IAM Hybrid Nodes doit être supérieur à celui `durationSeconds` de votre profil AWS IAM Roles Anywhere. Pour plus d'informations`MaxSessionDuration`, consultez la documentation de UpdateRole l'API.

Les certificats et clés par machine que vous générez à partir de votre autorité de certification (CA) doivent être placés dans le répertoire `/etc/iam/pki` de chaque nœud hybride avec les noms de fichier `server.pem` correspondant au certificat et `server.key` pour la clé.

Création du rôle IAM des nœuds hybrides

Pour exécuter les étapes décrites dans cette section, le principal IAM utilisant la AWS console ou la AWS CLI doit disposer des autorisations suivantes.

- `iam:CreatePolicy`
- `iam:CreateRole`
- `iam:AttachRolePolicy`
- Si vous utilisez AWS IAM Roles Anywhere

- `rolesanywhere:CreateTrustAnchor`
- `rolesanywhere:CreateProfile`
- `iam:PassRole`

AWS CloudFormation

Installez et configurez la AWS CLI, si ce n'est pas déjà fait. Voir Installation ou mise à jour vers la dernière version de la AWS CLI.

Étapes pour les AWS activations hybrides SSM

La CloudFormation pile crée le rôle IAM des nœuds hybrides avec les autorisations décrites ci-dessus. Le CloudFormation modèle ne crée pas l'activation hybride AWS SSM.

1. Téléchargez le CloudFormation modèle AWS SSM pour les nœuds hybrides :

```
curl -OL 'https://raw.githubusercontent.com/aws/eks-hybrid/refs/heads/main/example/
hybrid-ssm-cfn.yaml'
```

2. Créez un `cfn-ssm-parameters.json` avec les options suivantes :
   a. Remplacez `ROLE_NAME` par le nom de votre rôle IAM des nœuds hybrides. Par défaut, le CloudFormation modèle utilise `AmazonEKSHybridNodesRole` comme nom du rôle qu'il crée si vous ne spécifiez aucun nom.
   b. `TAG_KEY`Remplacez-la par la clé de balise de ressource AWS SSM que vous avez utilisée lors de la création de votre AWS activation hybride SSM. La combinaison de la clé de balise et de la valeur de balise est utilisée `ssm:DeregisterManagedInstance` à condition que le rôle IAM des nœuds hybrides ne soit autorisé qu'à désenregistrer les instances gérées par AWS SSM associées à votre AWS activation hybride SSM. Dans le CloudFormation modèle, la `TAG_KEY` valeur par défaut est. `EKSClusterARN`
   c. `TAG_VALUE`Remplacez-la par la valeur de balise de ressource AWS SSM que vous avez utilisée lors de la création de votre AWS activation hybride SSM. La combinaison de la clé de balise et de la valeur de balise est utilisée `ssm:DeregisterManagedInstance` à condition que le rôle IAM des nœuds hybrides ne soit autorisé qu'à désenregistrer les instances gérées par AWS SSM associées à votre AWS activation hybride SSM. Si vous utilisez la valeur par défaut `TAG_KEY` ou `EKSClusterARN`, transmettez l'ARN de votre cluster EKS en tant que `TAG_VALUE`. Les clusters EKS ARNs ont le format `arn:aws:eks:AWS_REGION:AWS_ACCOUNT_ID:cluster/CLUSTER_NAME`.

```
{
  "Parameters": {
    "RoleName": "ROLE_NAME",
    "SSMDeregisterConditionTagKey": "TAG_KEY",
    "SSMDeregisterConditionTagValue": "TAG_VALUE"
  }
}
```

3. Déployez la CloudFormation pile. Remplacez `STACK_NAME` par le nom de la CloudFormation pile.

```
aws cloudformation deploy \
    --stack-name STACK_NAME \
    --template-file hybrid-ssm-cfn.yaml \
    --parameter-overrides file://cfn-ssm-parameters.json \
    --capabilities CAPABILITY_NAMED_IAM
```

Étapes à suivre pour jouer AWS des rôles partout dans le monde

La CloudFormation pile crée l'ancre de confiance AWS IAM Roles Anywhere auprès de l'autorité de certification (CA) que vous configurez, crée le profil AWS IAM Roles Anywhere et crée le rôle IAM Hybrid Nodes avec les autorisations décrites précédemment.

1. Pour configurer une autorité de certification (CA)
   a. Pour utiliser une ressource d'autorité de certification AWS privée, ouvrez la console AWS Private Certificate Authority. Suivez les instructions du Guide de l'utilisateur de l'autorité de certification privée AWS.
   b. Pour utiliser une autorité de certification externe, suivez les instructions fournies par celle-ci. Vous fournirez le corps du certificat lors d'une étape ultérieure.
   c. Les certificats émis par le public CAs ne peuvent pas être utilisés comme points d'ancrage de confiance.
2. Téléchargez le CloudFormation modèle AWS IAM Roles Anywhere pour les nœuds hybrides

```
curl -OL 'https://raw.githubusercontent.com/aws/eks-hybrid/refs/heads/main/example/
hybrid-ira-cfn.yaml'
```

3. Créez un `cfn-iamra-parameters.json` avec les options suivantes :

a. Remplacez `ROLE_NAME` par le nom de votre rôle IAM des nœuds hybrides. Par défaut, le CloudFormation modèle utilise `AmazonEKSHybridNodesRole` comme nom du rôle qu'il crée si vous ne spécifiez aucun nom.

b. Remplacez `CERT_ATTRIBUTE` par l'attribut de certificat par machine qui identifie de manière unique votre hôte. L'attribut de certificat que vous utilisez doit correspondre au nodeName que vous utilisez pour la configuration `nodeadm` lorsque vous connectez des nœuds hybrides à votre cluster. Pour de plus amples informations, veuillez consulter the section called "Nœuds hybrides nodeadm". Par défaut, le CloudFormation modèle utilise `${aws:PrincipalTag/x509Subject/CN}` as le`CERT_ATTRIBUTE`, qui correspond au champ CN de vos certificats par machine. Vous pouvez également faire passer `$(aws:PrincipalTag/x509SAN/Name/CN}` comme votre `CERT_ATTRIBUTE`.

c. Remplacez `CA_CERT_BODY` par le corps du certificat de votre autorité de certification sans sauts de ligne. Le `CA_CERT_BODY` doit être au format PEM (Privacy Enhanced Mail). Si vous disposez d'un certificat CA au format PEM, supprimez les sauts de ligne et les lignes BEGIN CERTIFICATE et END CERTIFICATE avant d'insérer le corps du certificat CA dans votre fichier `cfn-iamra-parameters.json`.

```
{
  "Parameters": {
    "RoleName": "ROLE_NAME",
    "CertAttributeTrustPolicy": "CERT_ATTRIBUTE",
    "CABundleCert": "CA_CERT_BODY"
  }
}
```

4. Déployez le CloudFormation modèle. Remplacez `STACK_NAME` par le nom de la CloudFormation pile.

```
aws cloudformation deploy \
    --stack-name STACK_NAME \
    --template-file hybrid-ira-cfn.yaml \
    --parameter-overrides file://cfn-iamra-parameters.json
    --capabilities CAPABILITY_NAMED_IAM
```

AWS CLI

Installez et configurez la AWS CLI, si ce n'est pas déjà fait. Voir Installation ou mise à jour vers la dernière version de la AWS CLI.

Créer une politique de cluster EKS Describe

1. Créez un fichier nommé `eks-describe-cluster-policy.json` avec le contenu suivant :

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "eks:DescribeCluster"
            ],
            "Resource": "*"
        }
    ]
}
```

2. Créez la politique à l'aide de la commande suivante :

```
aws iam create-policy \
    --policy-name EKSDescribeClusterPolicy \
    --policy-document file://eks-describe-cluster-policy.json
```

Étapes pour les AWS activations hybrides SSM

1. Créez un fichier nommé `eks-hybrid-ssm-policy.json` avec les contenus suivants. La politique autorise deux actions `ssm:DescribeInstanceInformation` et `ssm:DeregisterManagedInstance`. La politique restreint l'`ssm:DeregisterManagedInstance`autorisation aux instances gérées AWS SSM associées à votre activation hybride AWS SSM en fonction de la balise de ressource que vous spécifiez dans votre politique de confiance.
   a. Remplacez `AWS_REGION` par la AWS région pour votre activation hybride AWS SSM.
   b. Remplacez `AWS_ACCOUNT_ID` par votre identifiant de AWS compte.
   c. `TAG_KEY`Remplacez-la par la clé de balise de ressource AWS SSM que vous avez utilisée lors de la création de votre AWS activation hybride SSM. La combinaison de la clé de balise et de la valeur de balise est utilisée `ssm:DeregisterManagedInstance` à condition que le rôle IAM des nœuds hybrides ne soit autorisé qu'à désenregistrer les instances gérées par AWS SSM

associées à votre AWS activation hybride SSM. Dans le CloudFormation modèle, la `TAG_KEY` valeur par défaut est. `EKSClusterARN`

d. `TAG_VALUE`Remplacez-la par la valeur de balise de ressource AWS SSM que vous avez utilisée lors de la création de votre AWS activation hybride SSM. La combinaison de la clé de balise et de la valeur de balise est utilisée `ssm:DeregisterManagedInstance` à condition que le rôle IAM des nœuds hybrides ne soit autorisé qu'à désenregistrer les instances gérées par AWS SSM associées à votre AWS activation hybride SSM. Si vous utilisez la valeur par défaut `TAG_KEY` ou `EKSClusterARN`, transmettez l'ARN de votre cluster EKS en tant que `TAG_VALUE`. Les clusters EKS ARNs ont le format `arn:aws:eks:AWS_REGION:AWS_ACCOUNT_ID:cluster/CLUSTER_NAME`.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "ssm:DescribeInstanceInformation",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "ssm:DeregisterManagedInstance",
            "Resource": "arn:aws:ssm:us-east-1:123456789012:managed-instance/*",
            "Condition": {
                "StringEquals": {
                    "ssm:resourceTag/TAG_KEY": "TAG_VALUE"
                }
            }
        }
    ]
}
```

2. Créez la politique à l'aide de la commande suivante

```
aws iam create-policy \
    --policy-name EKSHybridSSMPolicy \
    --policy-document file://eks-hybrid-ssm-policy.json
```

3. Créez un fichier nommé `eks-hybrid-ssm-trust.json`. `AWS_REGION`Remplacez-le par la AWS région de votre activation hybride AWS SSM et `AWS_ACCOUNT_ID` par votre identifiant de AWS compte.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"",
         "Effect":"Allow",
         "Principal":{
            "Service":"ssm.amazonaws.com"
         },
         "Action":"sts:AssumeRole",
         "Condition":{
            "StringEquals":{
               "aws:SourceAccount":"123456789012"
            },
            "ArnEquals":{
               "aws:SourceArn":"arn:aws:ssm:us-east-1:123456789012:*"
            }
         }
      }
   ]
}
```

4. Créez le rôle à l'aide de la commande suivante.

```
aws iam create-role \
    --role-name AmazonEKSHybridNodesRole \
    --assume-role-policy-document file://eks-hybrid-ssm-trust.json
```

5. Joignez le `EKSDescribeClusterPolicy` et le `EKSHybridSSMPolicy` que vous avez créés lors des étapes précédentes. Remplacez `AWS_ACCOUNT_ID` par votre identifiant de AWS compte.

```
aws iam attach-role-policy \
    --role-name AmazonEKSHybridNodesRole \
    --policy-arn arn:aws: iam::AWS_ACCOUNT_ID:policy/EKSDescribeClusterPolicy
```

```
aws iam attach-role-policy \
    --role-name AmazonEKSHybridNodesRole \
    --policy-arn arn:aws: iam::AWS_ACCOUNT_ID:policy/EKSHybridSSMPolicy
```

6. Joignez les politiques `AmazonEC2ContainerRegistryPullOnly` et les politiques `AmazonSSMManagedInstanceCore` AWS gérées.

```
aws iam attach-role-policy \
    --role-name AmazonEKSHybridNodesRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEC2ContainerRegistryPullOnly
```

```
aws iam attach-role-policy \
    --role-name AmazonEKSHybridNodesRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonSSMManagedInstanceCore
```

Étapes à suivre pour jouer AWS des rôles partout dans le monde

Pour utiliser AWS IAM Roles Anywhere, vous devez configurer votre ancre de confiance AWS IAM Roles Anywhere avant de créer le rôle IAM Hybrid Nodes. Pour obtenir des instructions, consultez the section called "Configurer des rôles AWS IAM n'importe où".

1. Créez un fichier nommé `eks-hybrid-iamra-trust.json`. Remplacez `TRUST_ANCHOR ARN` par l'ARN de l'ancre d'approbation que vous avez créée au cours des étapes the section called "Configurer des rôles AWS IAM n'importe où". La condition énoncée dans cette politique de confiance limite la capacité d' AWS IAM Roles Anywhere à assumer le rôle Hybrid Nodes IAM pour échanger des informations d'identification IAM temporaires uniquement lorsque le nom de session du rôle correspond au CN figurant dans le certificat x509 installé sur vos nœuds hybrides. Vous pouvez également utiliser d'autres attributs de certificat pour identifier de manière unique votre nœud. L'attribut de certificat que vous utilisez dans la stratégie de confiance doit correspondre au `nodeName` que vous avez défini dans votre configuration `nodeadm`. Pour de plus amples informations, veuillez consulter the section called "Nœuds hybrides nodeadm".

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "rolesanywhere.amazonaws.com"
            },
            "Action": [
                "sts:TagSession",
                "sts:SetSourceIdentity"
            ],
            "Condition": {
```

```
                "StringEquals": {
                    "aws:PrincipalTag/x509Subject/CN": "${aws:PrincipalTag/
x509Subject/CN}"
                },
                "ArnEquals": {
                    "aws:SourceArn": "arn:aws:rolesanywhere:us-
east-1:123456789012:trust-anchor/TA_ID"
                }
            }
        },
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "rolesanywhere.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "sts:RoleSessionName": "${aws:PrincipalTag/x509Subject/CN}",
                    "aws:PrincipalTag/x509Subject/CN": "${aws:PrincipalTag/
x509Subject/CN}"
                },
                "ArnEquals": {
                    "aws:SourceArn": "arn:aws:rolesanywhere:us-
east-1:123456789012:trust-anchor/TA_ID"
                }
            }
        }
    ]
}
```

2. Créez le rôle à l'aide de la commande suivante.

```
aws iam create-role \
    --role-name AmazonEKSHybridNodesRole \
    --assume-role-policy-document file://eks-hybrid-iamra-trust.json
```

3. Attachez le `EKSDescribeClusterPolicy` que vous avez créé lors des étapes précédentes. Remplacez `AWS_ACCOUNT_ID` par votre identifiant de AWS compte.

```
aws iam attach-role-policy \
    --role-name AmazonEKSHybridNodesRole \
    --policy-arn arn:aws: iam::AWS_ACCOUNT_ID:policy/EKSDescribeClusterPolicy
```

4. Joindre la politique `AmazonEC2ContainerRegistryPullOnly` AWS gérée

```
aws iam attach-role-policy \
    --role-name AmazonEKSHybridNodesRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEC2ContainerRegistryPullOnly
```

AWS Management Console

Créer une politique de cluster EKS Describe

1. Ouvrez la console Amazon IAM.
2. Dans le volet de navigation de gauche, choisissez Politiques.
3. Sur la page Politiques, choisissez Créer une politique.
4. Sur la page Spécifier les autorisations, dans le panneau Sélectionner un service, choisissez EKS.
   a. Filtrez les actions pour DescribeClusteret sélectionnez l'action DescribeClusterLire.
   b. Choisissez Suivant.
5. Sur la page Réviser et créer
   a. Saisissez un nom pour votre politique, par exemple `EKSDescribeClusterPolicy`.
   b. Choisissez Create Policy (Créer une politique).

Étapes pour les AWS activations hybrides SSM

1. Ouvrez la console Amazon IAM.
2. Dans le volet de navigation de gauche, choisissez Politiques.
3. Sur la page Politiques, choisissez Créer une politique.
4. Sur la page Spécifier les autorisations, dans le menu de navigation en haut à droite de l'éditeur de stratégie, sélectionnez JSON. Collez l'extrait suivant. Remplacez `AWS_REGION` par la AWS région de votre activation hybride AWS SSM et remplacez `AWS_ACCOUNT_ID` par votre identifiant de AWS compte. Remplacez `TAG_KEY` et `TAG_VALUE` par la clé de balise de ressource AWS SSM que vous avez utilisée lors de la création de votre activation hybride AWS SSM.

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

```
            "Action": "ssm:DescribeInstanceInformation",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "ssm:DeregisterManagedInstance",
            "Resource": "arn:aws:ssm:us-east-1:123456789012:managed-instance/*",
            "Condition": {
                "StringEquals": {
                    "ssm:resourceTag/TAG_KEY": "TAG_VALUE"
                }
            }
        }
    ]
}
```

a. Choisissez Suivant.

5. Sur la page Réviser et créer
   a. Saisissez un nom pour votre politique, par exemple `EKSHybridSSMPolicy`.
   b. Choisissez Create Policy (Créer une politique).
6. Dans le volet de navigation de gauche, choisissez Rôles.
7. Sur la page Rôles, choisissez Créer un rôle.
8. Sur la page Select trusted entity (Sélectionner une entité de confiance), procédez comme suit :
   a. Dans la section Type d'entité approuvée, sélectionnez Stratégie d'approbation personnalisée. Collez ce qui suit dans l'éditeur de stratégie de confiance personnalisée. `AWS_REGION`Remplacez-le par la AWS région de votre activation hybride AWS SSM et `AWS_ACCOUNT_ID` par votre identifiant de AWS compte.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"",
         "Effect":"Allow",
         "Principal":{
            "Service":"ssm.amazonaws.com"
         },
         "Action":"sts:AssumeRole",
         "Condition":{
            "StringEquals":{
```

```
                "aws:SourceAccount":"123456789012"
            },
            "ArnEquals":{
                "aws:SourceArn":"arn:aws:ssm:us-east-1:123456789012:*"
            }
         }
      }
   ]
}
```

b. Choisissez Suivant.

9. Sur la page Ajouter des autorisations, associez une stratégie personnalisée ou procédez comme suit :

   a. Dans la zone Stratégies de filtre, saisissez `EKSDescribeClusterPolicy`, ou le nom de la stratégie que vous avez créée ci-dessus. Cochez la case située à gauche du nom de votre police dans les résultats de recherche.

   b. Dans la zone Stratégies de filtre, saisissez `EKSHybridSSMPolicy`, ou le nom de la stratégie que vous avez créée ci-dessus. Cochez la case située à gauche du nom de votre police dans les résultats de recherche.

   c. Dans la zone Filter policies (Politiques de filtre), saisissez `AmazonEC2ContainerRegistryPullOnly`. Cochez la case à gauche de `AmazonEC2ContainerRegistryPullOnly` dans les résultats de recherche.

   d. Dans la zone Filter policies (Politiques de filtre), saisissez `AmazonSSMManagedInstanceCore`. Cochez la case à gauche de `AmazonSSMManagedInstanceCore` dans les résultats de recherche.

   e. Choisissez Suivant.

10. Sur la page Name, review, and create (Nommer, vérifier et créer), procédez comme suit :

    a. Pour Role name (Nom de rôle), saisissez un nom unique pour votre rôle, par exemple, `AmazonEKSHybridNodesRole`.

    b. Pour Description, remplacez le texte actuel par un texte descriptif tel que `Amazon EKS - Hybrid Nodes role`.

    c. Choisissez Créer un rôle.

Étapes à suivre pour jouer AWS des rôles partout dans le monde

Pour utiliser AWS IAM Roles Anywhere, vous devez configurer votre ancre de confiance AWS IAM Roles Anywhere avant de créer le rôle IAM Hybrid Nodes. Pour obtenir des instructions, consultez the section called "Configurer des rôles AWS IAM n'importe où".

1. Ouvrez la console Amazon IAM.
2. Dans le volet de navigation de gauche, choisissez Rôles.
3. Sur la page Rôles, choisissez Créer un rôle.
4. Sur la page Select trusted entity (Sélectionner une entité de confiance), procédez comme suit :
   a. Dans la section Type d'entité approuvée, sélectionnez Stratégie d'approbation personnalisée. Collez ce qui suit dans l'éditeur de stratégie de confiance personnalisée. Remplacez `TRUST_ANCHOR ARN` par l'ARN de l'ancre d'approbation que vous avez créée au cours des étapes the section called "Configurer des rôles AWS IAM n'importe où". La condition énoncée dans cette politique de confiance limite la capacité d' AWS IAM Roles Anywhere à assumer le rôle Hybrid Nodes IAM pour échanger des informations d'identification IAM temporaires uniquement lorsque le nom de session du rôle correspond au CN figurant dans le certificat x509 installé sur vos nœuds hybrides. Vous pouvez également utiliser d'autres attributs de certificat pour identifier de manière unique votre nœud. L'attribut de certificat que vous utilisez dans la stratégie de confiance doit correspondre au nom de nœud que vous avez défini dans votre configuration nodeadm. Pour de plus amples informations, veuillez consulter the section called "Nœuds hybrides nodeadm".

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "rolesanywhere.amazonaws.com"
            },
            "Action": [
                "sts:TagSession",
                "sts:SetSourceIdentity"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:PrincipalTag/x509Subject/CN": "${aws:PrincipalTag/
x509Subject/CN}"
                },
                "ArnEquals": {
```

```
                    "aws:SourceArn": "arn:aws:rolesanywhere:us-
east-1:123456789012:trust-anchor/TA_ID"
                }
            }
        },
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "rolesanywhere.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "sts:RoleSessionName": "${aws:PrincipalTag/x509Subject/CN}",
                    "aws:PrincipalTag/x509Subject/CN": "${aws:PrincipalTag/
x509Subject/CN}"
                },
                "ArnEquals": {
                    "aws:SourceArn": "arn:aws:rolesanywhere:us-
east-1:123456789012:trust-anchor/TA_ID"
                }
            }
        }
    ]
}
```

b. Choisissez Suivant.

5. Sur la page Ajouter des autorisations, associez une stratégie personnalisée ou procédez comme suit :

   a. Dans la zone Stratégies de filtre, saisissez `EKSDescribeClusterPolicy`, ou le nom de la stratégie que vous avez créée ci-dessus. Cochez la case située à gauche du nom de votre police dans les résultats de recherche.

   b. Dans la zone Filter policies (Politiques de filtre), saisissez `AmazonEC2ContainerRegistryPullOnly`. Cochez la case à gauche de `AmazonEC2ContainerRegistryPullOnly` dans les résultats de recherche.

   c. Choisissez Suivant.

6. Sur la page Name, review, and create (Nommer, vérifier et créer), procédez comme suit :

   a. Pour Role name (Nom de rôle), saisissez un nom unique pour votre rôle, par exemple, `AmazonEKSHybridNodesRole`.

b. Pour Description, remplacez le texte actuel par un texte descriptif tel que `Amazon EKS - Hybrid Nodes role`.

c. Choisissez Créer un rôle.

Créer un cluster Amazon EKS avec des nœuds hybrides

Cette rubrique fournit une vue d'ensemble des options disponibles et décrit les éléments à prendre en compte lors de la création d'un cluster Amazon EKS hybride compatible avec les nœuds. Les nœuds hybrides EKS prennent en charge la même version de Kubernetes que les clusters Amazon EKS avec nœuds cloud, y compris la prise en charge standard et étendue.

Si vous ne prévoyez pas d'utiliser les nœuds hybrides EKS, consultez la documentation principale sur la création de clusters Amazon EKS à l'adresse the section called "Créer un cluster".

Conditions préalables

- Le the section called "Prérequis" terminé. Avant de créer votre cluster compatible avec les nœuds hybrides, vous devez CIDRs identifier votre nœud local et éventuellement votre pod, créer votre VPC et vos sous-réseaux conformément aux exigences d'EKS et aux exigences relatives aux nœuds hybrides, ainsi que votre groupe de sécurité avec des règles d'entrée pour votre espace sur site et éventuellement pour votre espace. CIDRs Pour plus d'informations sur ces prérequis, consultez the section called "Préparez votre réseau".
- La dernière version de l'interface de ligne de AWS commande (AWS CLI) installée et configurée sur votre appareil. Pour vérifier votre version actuelle, utilisez `aws --version`. Les gestionnaires de packages tels que yum, apt-get ou Homebrew pour macOS ont souvent plusieurs versions de retard sur la dernière version de la CLI. AWS Pour installer la dernière version, reportez-vous aux sections Installation ou mise à jour de la dernière version de la AWS CLI et Configuration des paramètres de la AWS CLI dans le Guide de l'utilisateur de l'interface de ligne de AWS commande.
- Un principal IAM disposant des autorisations nécessaires pour créer des rôles IAM et associer des politiques, ainsi que pour créer et décrire des clusters EKS

Considérations

- Votre cluster doit utiliser soit `API` ou `API_AND_CONFIG_MAP` pour le mode d'authentification du cluster.
- Votre cluster doit utiliser une famille IPv4 d'adresses.

- Votre cluster doit utiliser une connectivité de point de terminaison de cluster publique ou privée. Votre cluster ne peut pas utiliser la connectivité de point de terminaison de cluster « public et privé », car le point de terminaison du serveur d'API Amazon EKS Kubernetes sera transmis au public IPs pour les nœuds hybrides exécutés en dehors de votre VPC.
- L'authentification OIDC est prise en charge pour les clusters EKS avec des nœuds hybrides.
- Vous pouvez ajouter, modifier ou supprimer la configuration des nœuds hybrides d'un cluster existant. Pour de plus amples informations, veuillez consulter the section called "Cluster existant".

Étape 1 : créer un rôle IAM de cluster

Si vous possédez déjà un rôle IAM dans le cluster, ou si vous allez créer votre cluster avec `eksctl` ou AWS CloudFormation, vous pouvez ignorer cette étape. Par défaut, `eksctl` le AWS CloudFormation modèle crée le rôle IAM du cluster pour vous.

1. Exécutez la commande suivante pour créer un fichier JSON de politique d'approbation IAM.

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

2. Créez le rôle IAM de cluster Amazon EKS. Si nécessaire, préfacez eks-cluster-role-trust - policy.json avec le chemin sur l'ordinateur où vous avez écrit le fichier à l'étape précédente. La commande associe la politique d'approbation que vous avez créée lors de l'étape précédente à ce rôle. Pour créer un rôle IAM, le principal IAM qui crée le rôle doit se voir attribuer l'action (autorisation) IAM `iam:CreateRole`.

```
aws iam create-role \
    --role-name myAmazonEKSClusterRole \
    --assume-role-policy-document file://"eks-cluster-role-trust-policy.json"
```

3. Vous pouvez attribuer la politique gérée par Amazon EKS ou créer votre propre politique personnalisée. Pour connaître les autorisations minimales que vous devez utiliser dans votre politique personnalisée, consultez the section called "Rôle IAM de nœud". Attachez la politique gérée par Amazon EKS appelée `AmazonEKSClusterPolicy` au rôle. Pour attacher une politique IAM à un principal IAM, le principal qui attache la politique doit se voir attribuer l'une des actions (autorisations) IAM `iam:AttachUserPolicy` ou `iam:AttachRolePolicy`.

```
aws iam attach-role-policy \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSClusterPolicy \
    --role-name myAmazonEKSClusterRole
```

Étape 2 : créer un cluster compatible avec les nœuds hybrides

Vous pouvez créer un cluster à l'aide de :

- eksctl
- AWS CloudFormation
- AWS INTERFACE DE LIGNE DE COMMANDE (CLI)
- AWS Management Console

Création d'un cluster compatible avec les nœuds hybrides – eksctl

Vous devez installer la dernière version de l'outil en ligne de commande `eksctl`. Pour installer ou mettre à jour `eksctl`, veuillez consulter Installation dans la documentation de `eksctl`.

1. Créez `cluster-config.yaml` pour définir un cluster Amazon EKS IPv4 compatible avec les nœuds hybrides. Effectuez les remplacements suivants dans votre fichier `cluster-config.yaml`. Pour une liste complète des paramètres, consultez la documentation eksctl.
   a. Remplacez `CLUSTER_NAME` par un nom pour votre cluster. Un nom ne peut contenir que des caractères alphanumériques (sensibles à la casse) et des traits d'union. Il doit commencer par un caractère alphanumérique et ne peut pas dépasser 100 caractères. Le nom doit être unique dans la AWS région et le AWS compte dans lesquels vous créez le cluster.
   b. `AWS_REGION`Remplacez-le par la AWS région dans laquelle vous souhaitez créer votre cluster.
   c. Remplacez la version `K8S_VERSION` par n'importe quelle version prise en charge par Amazon EKS.

d. Remplacez `CREDS_PROVIDER` par `ssm` ou `ira` en fonction du fournisseur d'informations d'identification que vous avez configuré dans les étapes pour the section called "Préparation des informations d'identification".

e. `CA_BUNDLE_CERT`Remplacez-le si votre fournisseur d'informations d'identification est défini sur`ira`, ce qui utilise AWS IAM Roles Anywhere comme fournisseur d'informations d'identification. CA_BUNDLE_CERT est l'autorité de certification (CA) et dépend de votre choix de CA. Le certificat doit être au format PEM (Privacy Enhanced Mail).

f. Remplacez `GATEWAY_ID` par l'ID de votre passerelle privée virtuelle ou passerelle de transit à associer à votre VPC.

g. Remplacez `REMOTE_NODE_CIDRS` par le CIDR du nœud sur site pour vos nœuds hybrides.

h. Remplacez `REMOTE_POD_CIDRS` par le CIDR du pod sur site pour les charges de travail exécutées sur des nœuds hybrides ou supprimez la ligne de votre configuration si vous n'exécutez pas de webhooks sur des nœuds hybrides. Vous devez configurer votre `REMOTE_POD_CIDRS` si votre CNI n'utilise pas la traduction d'adresses réseau (NAT) ou le masquage pour les adresses IP des pods lorsque le trafic des pods quitte vos hôtes sur site. Vous devez configurer `REMOTE_POD_CIDRS` si vous exécutez des webhooks sur des nœuds hybrides. Pour plus d'informations, consultez the section called "Configuration de webhooks".

i. Les blocs CIDR de vos nœuds et pods sur site doivent répondre aux exigences suivantes :

   i. Être dans l'une des plages de la IPv4 RFC-1918 : `10.0.0.0/8`172.16.0.0/12, ou. `192.168.0.0/16`

   ii. Pas de chevauchement entre eux, `VPC  CIDR` pour votre cluster ou pour votre service Kubernetes CIDR IPv4

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: CLUSTER_NAME
  region: AWS_REGION
  version: "K8S_VERSION"

remoteNetworkConfig:
  iam:
    provider: CREDS_PROVIDER # default SSM, can also be set to IRA
    # caBundleCert: CA_BUNDLE_CERT
  vpcGatewayID: GATEWAY_ID
  remoteNodeNetworks:
```

```
  - cidrs: ["REMOTE_NODE_CIDRS"]
  remotePodNetworks:
  - cidrs: ["REMOTE_POD_CIDRS"]
```

2. Exécutez la commande suivante :

```
eksctl create cluster -f cluster-config.yaml
```

L'approvisionnement de cluster dure plusieurs minutes. Pendant la création du cluster, plusieurs lignes de sortie apparaissent. La dernière ligne de sortie est similaire à celle de l'exemple suivant.

```
[#]  EKS cluster "CLUSTER_NAME" in "REGION" region is ready
```

3. Poursuivez avec the section called "Étape 3 : mettre à jour kubeconfig".

Créez un cluster compatible avec les nœuds hybrides - AWS CloudFormation

La CloudFormation pile crée le rôle IAM du cluster EKS et un cluster EKS avec le `RemoteNodeNetwork` et `RemotePodNetwork` que vous spécifiez. Modifiez le CloudFormation modèle si vous devez personnaliser les paramètres de votre cluster EKS qui ne sont pas exposés dans le CloudFormation modèle.

1. Téléchargez le CloudFormation modèle.

```
curl -OL 'https://raw.githubusercontent.com/aws/eks-hybrid/refs/heads/main/example/
hybrid-eks-cfn.yaml'
```

2. Créez un `cfn-eks-parameters.json` et spécifiez votre configuration pour chaque valeur.
   a. `CLUSTER_NAME` : nom du cluster EKS à créer
   b. `CLUSTER_ROLE_NAME` : nom du rôle IAM du cluster EKS à créer. La valeur par défaut du modèle est « EKSCluster Rôle ».
   c. `SUBNET1_ID` : l'ID du premier sous-réseau que vous avez créé lors des étapes préalables
   d. `SUBNET2_ID` : l'ID du deuxième sous-réseau que vous avez créé dans les étapes préalables.
   e. `SG_ID` : l'ID du groupe de sécurité que vous avez créé lors des étapes préalables.
   f. `REMOTE_NODE_CIDRS` : le CIDR du nœud sur site pour vos nœuds hybrides
   g. `REMOTE_POD_CIDRS` : le CIDR du pod sur site pour les charges de travail exécutées sur des nœuds hybrides. Vous devez configurer votre `REMOTE_POD_CIDRS` si votre CNI n'utilise pas

la traduction d'adresses réseau (NAT) ou le masquage pour les adresses IP des pods lorsque le trafic des pods quitte vos hôtes sur site. Vous devez configurer `REMOTE_POD_CIDRS` si vous exécutez des webhooks sur des nœuds hybrides. Pour plus d'informations, consultez the section called "Configuration de webhooks".

h. Les blocs CIDR de vos nœuds et pods sur site doivent répondre aux exigences suivantes :

   i. Être dans l'une des plages de la IPv4 RFC-1918 : `10.0.0.0/8172.16.0.0/12`, ou. `192.168.0.0/16`

   ii. Ne vous chevauchez pas, ni avec le CIDR `VPC  CIDR` de votre cluster, ni avec le CIDR de votre service Kubernetes. IPv4

i. `CLUSTER_AUTH` : le mode d'authentification du cluster pour votre cluster. Les valeurs valides sont `API` et `API_AND_CONFIG_MAP`. La valeur par défaut dans le modèle est `API_AND_CONFIG_MAP`.

j. `CLUSTER_ENDPOINT` : la connectivité du point de terminaison de votre cluster. Les valeurs valides sont « Public » et « Privé ». La valeur par défaut dans le modèle est Privé, ce qui signifie que vous ne pourrez vous connecter au point de terminaison de l'API Kubernetes qu'à partir de votre VPC.

k. `K8S_VERSION` : la version de Kubernetes à utiliser pour votre cluster. Consultez les versions prises en charge par Amazon EKS.

```
{
  "Parameters": {
    "ClusterName": "CLUSTER_NAME",
    "ClusterRoleName": "CLUSTER_ROLE_NAME",
    "SubnetId1": "SUBNET1_ID",
    "SubnetId2": "SUBNET2_ID",
    "SecurityGroupId" "SG_ID",
    "RemoteNodeCIDR": "REMOTE_NODE_CIDRS",
    "RemotePodCIDR": "REMOTE_POD_CIDRS",
    "ClusterAuthMode": "CLUSTER_AUTH",
    "ClusterEndpointConnectivity": "CLUSTER_ENDPOINT",
    "K8sVersion": "K8S_VERSION"
  }
 }
```

3. Déployez la CloudFormation pile. `STACK_NAME`Remplacez-le par le nom de la CloudFormation pile et `AWS_REGION` par AWS la région dans laquelle le cluster sera créé.

```
aws cloudformation deploy \
```

```
    --stack-name STACK_NAME \
    --region AWS_REGION \
    --template-file hybrid-eks-cfn.yaml \
    --parameter-overrides file://cfn-eks-parameters.json \
    --capabilities CAPABILITY_NAMED_IAM
```

L'approvisionnement de cluster dure plusieurs minutes. Vous pouvez vérifier l'état de votre pile à l'aide de la commande suivante. `STACK_NAME`Remplacez-le par le nom de la CloudFormation pile et `AWS_REGION` par AWS la région dans laquelle le cluster sera créé.

```
aws cloudformation describe-stacks \
    --stack-name STACK_NAME \
    --region AWS_REGION \
    --query 'Stacks[].StackStatus'
```

4. Poursuivez avec the section called "Étape 3 : mettre à jour kubeconfig".

Créer un cluster compatible avec les nœuds hybrides - CLI AWS

1. Exécutez la commande suivante pour créer un cluster EKS hybride compatible avec les nœuds. Avant d'exécuter la commande, remplacez ce qui suit par vos paramètres. Pour obtenir la liste complète des paramètres, consultez la documentation the section called "Créer un cluster".
   a. `CLUSTER_NAME` : nom du cluster EKS à créer
   b. `AWS_REGION`: AWS région dans laquelle le cluster sera créé.
   c. `K8S_VERSION` : la version de Kubernetes à utiliser pour votre cluster. Consultez les versions prises en charge par Amazon EKS.
   d. `ROLE_ARN` : le rôle Amazon EKS que vous avez configuré pour votre cluster. Pour plus d'informations, consultez la section Rôle IAM du cluster Amazon EKS.
   e. `SUBNET1_ID` : l'ID du premier sous-réseau que vous avez créé lors des étapes préalables
   f. `SUBNET2_ID` : l'ID du deuxième sous-réseau que vous avez créé dans les étapes préalables.
   g. `SG_ID` : l'ID du groupe de sécurité que vous avez créé lors des étapes préalables.
   h. Vous pouvez utiliser `API` et `API_AND_CONFIG_MAP` pour votre mode d'authentification d'accès au cluster. Dans la commande ci-dessous, le mode d'authentification d'accès au cluster est défini sur `API_AND_CONFIG_MAP`.
   i. Vous pouvez utiliser les paramètres `endpointPublicAccess` et `endpointPrivateAccess` pour activer ou désactiver l'accès public et privé au point de terminaison de serveur d'API

Kubernetes de votre cluster. Dans la commande ci-dessous, `endpointPublicAccess` est défini sur false et `endpointPrivateAccess` est défini sur true.

j. `REMOTE_NODE_CIDRS` : le CIDR du nœud sur site pour vos nœuds hybrides.

k. `REMOTE_POD_CIDRS` (facultatif) : le CIDR du pod sur site pour les charges de travail exécutées sur des nœuds hybrides.

l. Les blocs CIDR de vos nœuds et pods sur site doivent répondre aux exigences suivantes :

i. Être dans l'une des plages de la IPv4 RFC-1918 : `10.0.0.0/8172.16.0.0/12`, ou. `192.168.0.0/16`

ii. Ne vous chevauchez pas, que ce soit `VPC CIDR` pour votre cluster Amazon EKS ou pour le CIDR de votre service Kubernetes. IPv4

```
aws eks create-cluster \
    --name CLUSTER_NAME \
    --region AWS_REGION \
    --kubernetes-version K8S_VERSION \
    --role-arn ROLE_ARN \
    --resources-vpc-config
 subnetIds=SUBNET1_ID,SUBNET2_ID,securityGroupIds=SG_ID,endpointPrivateAccess=true,endpo
 \
    --access-config authenticationMode=API_AND_CONFIG_MAP \
    --remote-network-config '{"remoteNodeNetworks":[{"cidrs":
["REMOTE_NODE_CIDRS"]}],"remotePodNetworks":[{"cidrs":["REMOTE_POD_CIDRS"]}]}'
```

2. Le provisionnement du cluster prend quelques minutes. Vous pouvez vérifier le statut de votre cluster avec la commande suivante. Remplacez `CLUSTER_NAME` par le nom du cluster que vous créez et `AWS_REGION` par la AWS région dans laquelle le cluster est créé. Ne passez pas à l'étape suivante tant que le résultat renvoyé n'est pas `ACTIVE`.

```
aws eks describe-cluster \
    --name CLUSTER_NAME \
    --region AWS_REGION \
    --query "cluster.status"
```

3. Poursuivez avec the section called "Étape 3 : mettre à jour kubeconfig".

Créez un cluster compatible avec les nœuds hybrides - AWS Management Console

1. Ouvrez la console Amazon EKS à l'adresse Amazon EKS console.

2. Choisissez Add cluster (Ajouter un cluster), puis choisissez Create (Créer).
3. Sur la page Configurer le cluster, renseignez les champs suivants :
   a. Nom : nom de votre cluster. Le nom ne peut contenir que des caractères alphanumériques (sensibles à la casse), des tirets et des traits de soulignement. Il doit commencer par un caractère alphanumérique et ne peut pas dépasser 100 caractères. Le nom doit être unique dans la AWS région et le AWS compte dans lesquels vous créez le cluster.
   b. Rôle IAM du cluster : choisissez le rôle IAM du cluster Amazon EKS que vous avez créé pour permettre au plan de contrôle Kubernetes de gérer AWS les ressources en votre nom.
   c. Version Kubernetes : version de Kubernetes à utiliser pour votre cluster. Nous vous recommandons de sélectionner la dernière version, sauf si vous avez besoin d'une version antérieure.
   d. Politique de mise à niveau : choisissez Extended ou Standard.
      i. Étendu : Cette option prend en charge la version de Kubernetes pendant 26 mois après la date de sortie. La période de support prolongée a un coût horaire supplémentaire qui commence après la fin de la période de support standard. À la fin du support étendu, votre cluster sera automatiquement mis à niveau vers la version suivante.
      ii. Standard : Cette option prend en charge la version de Kubernetes pendant 14 mois après la date de sortie. Il n'y a aucun coût supplémentaire. À la fin du support standard, votre cluster sera automatiquement mis à niveau vers la version suivante.
   e. Accès au cluster : Choisissez d'autoriser ou de refuser l'accès à l'administrateur du cluster et sélectionnez un mode d'authentification. Les modes d'authentification suivants sont pris en charge pour les clusters compatibles avec les nœuds hybrides.
      i. API EKS : le cluster s'approvisionnera en principaux IAM authentifiés uniquement à partir de l'entrée d'accès EKS. APIs
      ii. API EKS et ConfigMap : Le cluster fournira des principes IAM authentifiés à la fois à partir de l'entrée APIs d'accès EKS et du. `aws-auth` ConfigMap
   f. Chiffrement des secrets : (facultatif) choisissez d'activer le chiffrement des secrets Kubernetes à l'aide d'une clé KMS. Vous pouvez également activer cette option une fois que vous avez créé votre cluster. Avant d'activer cette fonctionnalité, assurez-vous d'avoir pris connaissance des informations mentionnées dans the section called "Activer le chiffrement secret".
   g. Déplacement de zone ARC : Si cette option est activée, EKS enregistrera votre cluster auprès du déplacement de zone ARC afin de vous permettre d'utiliser le déplacement de zone pour détourner le trafic applicatif d'une zone de disponibilité.

h. Identifications◦: (facultatif) ajoutez des identifications à votre cluster. Pour de plus amples informations, veuillez consulter the section called “Balisage de vos ressources ”.

i. Lorsque vous avez terminé d'utiliser cette page, choisissez Suivant.

4. Sur la page Spécifier les réseaux sélectionnez des valeurs pour les champs suivants :

a. VPC : choisissez un VPC existant qui répond à the section called “Exigences requises pour le VPC et les sous-réseaux” et aux exigences des nœuds hybrides Amazon EKS. Avant de choisir un VPC, nous vous recommandons de vous familiariser avec toutes les exigences et considérations décrites dans la section Consulter les exigences réseau Amazon EKS pour les VPC, les sous-réseaux et les nœuds hybrides. Vous ne pouvez pas modifier le VPC que vous souhaitez utiliser après la création du cluster. Si aucun VPCs n'est répertorié, vous devez d'abord en créer un. Pour plus d'informations, consultez the section called “Création d'un VPC” ainsi que les exigences réseau des nœuds hybrides Amazon EKS.

b. Sous-réseaux : par défaut, tous les sous-réseaux disponibles dans le VPC spécifié dans le champ précédent sont présélectionnés. Vous devez en sélectionner au moins deux.

c. Groupes de sécurité : (facultatif) spécifiez un ou plusieurs groupes de sécurité créant des interfaces réseau auxquelles vous souhaitez associer Amazon EKS. Au moins l'un des groupes de sécurité que vous spécifiez doit disposer de règles entrantes pour votre nœud local et éventuellement pour votre pod. CIDRs Pour plus d'informations, consultez les exigences réseau des nœuds hybrides Amazon EKS. Que vous choisissiez ou non un groupe de sécurité, Amazon EKS crée un groupe de sécurité qui permet la communication entre votre cluster et votre VPC. Amazon EKS associe ce groupe de sécurité, et tout ce que vous choisissez, aux interfaces réseau qu'il crée. Pour plus d'informations sur le groupe de sécurité de cluster créé par Amazon EKS, consultez the section called “Exigences de groupe de sécurité” Vous pouvez modifier les règles du groupe de sécurité de cluster créé par Amazon EKS.

d. Choisissez la famille d'adresses IP du cluster : vous devez choisir IPv4 des clusters compatibles avec les nœuds hybrides.

e. (Facultatif) Choisissez Configurer la plage d'adresses IP du service Kubernetes et spécifiez une plage de services. IPv4

f. Choisissez Configurer les réseaux distants pour activer les nœuds hybrides et spécifiez votre nœud local et votre pod CIDRs pour les nœuds hybrides.

g. Vous devez configurer le CIDR de votre pod distant si votre CNI n'utilise pas la traduction d'adresses réseau (NAT) ou le masquage pour les adresses IP des pods lorsque le trafic des pods quitte vos hôtes sur site. Vous devez configurer le CIDR du pod distant si vous exécutez des webhooks sur des nœuds hybrides.

h. Les blocs CIDR de vos nœuds et pods sur site doivent répondre aux exigences suivantes :

i. Être dans l'une des plages de la IPv4 RFC-1918 : `10.0.0.0/8`172.16.0.0/12, ou. `192.168.0.0/16`

ii. Pas de chevauchement entre eux, `VPC CIDR` pour votre cluster ou pour votre service Kubernetes CIDR IPv4

i. Pour Accès au point de terminaison de cluster, sélectionnez une option. Une fois votre cluster créé, vous pouvez modifier cette option. Pour les clusters compatibles avec les nœuds hybrides, vous devez choisir entre Public et Privé. Avant de sélectionner une option autre que par défaut, assurez-vous de vous familiariser avec les options et leurs implications. Pour de plus amples informations, veuillez consulter the section called “Accès au point de terminaison de cluster”.

j. Lorsque vous avez terminé d'utiliser cette page, choisissez Suivant.

5. (Facultatif) Sur la page Configurer l'observabilité, choisissez les options de métriques et de journalisation du plan de contrôle à activer. Par défaut, chaque type de journal est désactivé.

a. Pour plus d'informations sur l'option Prometheus metrics, consultez the section called “Métriques Prometheus”.

b. Pour plus d'informations sur les options de journalisation de contrôle EKS, voir the section called “journaux de plan de contrôle ”.

c. Lorsque vous avez terminé d'utiliser cette page, choisissez Suivant.

6. Sur la page Select add-ons (Sélectionner les modules complémentaires), choisissez les modules complémentaires que vous souhaitez ajouter à votre cluster.

a. Vous pouvez choisir autant de modules complémentaires Amazon EKS et de modules complémentaires AWS Marketplace que vous le souhaitez. Les modules complémentaires Amazon EKS qui ne sont pas compatibles avec les nœuds hybrides sont signalés par la mention « Non compatible avec les nœuds hybrides » et sont soumis à une règle d'anti-affinité afin d'empêcher leur exécution sur les nœuds hybrides. Pour plus d'informations, consultez la section Configuration des modules complémentaires pour les nœuds hybrides. Si les modules complémentaires AWS Marketplace que vous souhaitez installer ne figurent pas dans la liste, vous pouvez rechercher les modules complémentaires AWS Marketplace disponibles en saisissant du texte dans le champ de recherche. Vous pouvez également effectuer une recherche par catégorie, fournisseur ou modèle de tarification, puis choisir les modules complémentaires dans les résultats de la recherche.

b. Certains modules complémentaires, tels que CoreDNS et kube-proxy, sont installés par défaut. Si vous désactivez l'un des modules complémentaires par défaut, cela peut affecter votre capacité à exécuter des applications Kubernetes.

c. Lorsque vous avez terminé cette page, sélectionnez `Next`.

7. Sur la page Configurer les paramètres de modules complémentaires sélectionnés, sélectionnez la version que vous voulez installer.

   a. Vous pourrez toujours effectuer une mise à jour vers une version ultérieure après la création du cluster. Vous pourrez mettre à jour la configuration de chaque module complémentaire après la création du cluster. Pour plus d'informations sur la configuration des modules complémentaires, consultez the section called "Mettre à jour un module complémentaire". Pour les versions des modules complémentaires compatibles avec les nœuds hybrides, voir the section called "Configurer les modules complémentaires".

   b. Lorsque vous avez terminé d'utiliser cette page, choisissez Suivant.

8. Sur la page Vérifier et créer, passez en revue les informations que vous avez saisies ou sélectionnées sur les pages précédentes. Si vous devez apporter des modifications, choisissez Modifier. Quand vous êtes satisfait, choisissez Créer. Le champ État affiche EN COURS DE CRÉATION pendant que le cluster est provisionné. L'approvisionnement de cluster dure plusieurs minutes.

9. Poursuivez avec the section called "Étape 3 : mettre à jour kubeconfig".

Étape 3 : mettre à jour kubeconfig

Si vous avez créé votre cluster à l'aide de `eksctl`, vous pouvez ignorer cette étape. Cela est dû au fait que `eksctl` a déjà terminé cette étape pour vous. Activez `kubectl` la communication avec votre cluster en ajoutant un nouveau contexte au fichier de configuration `kubectl`. Pour plus d'informations sur la création et la mise à jour du fichier, consultez the section called "Accès au cluster avec kubectl".

```
aws eks update-kubeconfig --name CLUSTER_NAME --region AWS_REGION
```

L'exemple qui suit illustre un résultat.

```
Added new context arn:aws: eks:AWS_REGION:111122223333:cluster/CLUSTER_NAME to /home/
username/.kube/config
```

Confirmez la communication avec votre cluster en exécutant la commande suivante.

```
kubectl get svc
```

L'exemple qui suit illustre un résultat.

```
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.100.0.1   <none>        443/TCP   28h
```

Étape 4 : configuration du cluster

Ensuite, consultez the section called "Préparation de l'accès aux clusters" pour activer l'accès pour que vos nœuds hybrides puissent rejoindre votre cluster.

Activation de nœuds hybrides sur un cluster Amazon EKS existant ou modification de la configuration

Cette rubrique fournit une vue d'ensemble des options disponibles et décrit les éléments à prendre en compte lorsque vous ajoutez, modifiez ou supprimez la configuration des nœuds hybrides pour un cluster Amazon EKS.

Pour permettre à un cluster Amazon EKS d'utiliser des nœuds hybrides, ajoutez les plages CIDR d'adresses IP de votre nœud sur site et, éventuellement, le réseau de pods dans la configuration `RemoteNetworkConfig`. EKS utilise cette liste de CIDR pour activer la connectivité entre le cluster et vos réseaux sur site. Pour obtenir la liste complète des options disponibles lors de la mise à jour de la configuration de votre cluster, consultez la section UpdateClusterConfig dans la référence API Amazon EKS.

Vous pouvez effectuer l'une des actions suivantes sur la configuration réseau des nœuds hybrides EKS dans un cluster :

- Ajoutez une configuration réseau à distance pour activer les nœuds hybrides EKS dans un cluster existant.
- Ajoutez, modifiez ou supprimez les réseaux de nœuds distants ou les réseaux de pods distants dans un cluster existant.
- Supprimez toutes les plages CIDR du réseau des nœuds distants pour désactiver les nœuds hybrides EKS dans un cluster existant.

Prérequis

- Avant d'activer votre cluster Amazon EKS pour les nœuds hybrides, assurez-vous que votre environnement répond aux exigences décrites à the section called "Prérequis", et détaillées à the section called "Préparez votre réseau", the section called "Préparer le système d'exploitation", et the section called "Préparation des informations d'identification".
- Votre cluster doit utiliser la famille d'adresses IPv4.
- Votre cluster doit utiliser soit `API` ou `API_AND_CONFIG_MAP` pour le mode d'authentification du cluster. La procédure de modification du mode d'authentification du cluster est décrite à l'adresse the section called "Mode d'authentification".
- Nous vous recommandons d'utiliser soit l'accès public, soit l'accès privé pour le point de terminaison du serveur API Amazon EKS Kubernetes, mais pas les deux. Si vous choisissez « Public et privé », le point de terminaison du serveur API Amazon EKS Kubernetes sera toujours résolu vers les adresses IP publiques des nœuds hybrides s'exécutant en dehors de votre VPC, ce qui peut empêcher vos nœuds hybrides de rejoindre le cluster. La procédure permettant de modifier l'accès réseau à votre cluster est décrite à l'adresse the section called "Accès au point de terminaison de cluster".
- La dernière version de l'interface de la ligne de commande AWS (AWS CLI) installée et configurée sur votre appareil. Pour vérifier votre version actuelle, utilisez `aws --version`. Les gestionnaires de paquets tels que yum, apt-get ou Homebrew pour macOS ont souvent plusieurs versions de retard par rapport à la dernière version de l'interface CLI AWS. Pour installer la dernière version, consultez les sections Installation ou mise à jour vers la dernière version de l'AWS CLI et Configuration des paramètres de l'AWS CLI du Guide de l'utilisateur de l'interface de la ligne de commande AWS.
- Un principal IAM disposant de l'autorisation d'appeler UpdateClusterConfig sur votre cluster Amazon EKS.
- Mettez à jour les modules complémentaires vers des versions compatibles avec les nœuds hybrides. Pour les versions des modules complémentaires compatibles avec les nœuds hybrides, voir the section called "Configurer les modules complémentaires".
- Si vous utilisez des modules complémentaires qui ne sont pas compatibles avec les nœuds hybrides, assurez-vous que le DaemonSet ou le Déploiment du module complémentaire dispose de la règle d'affinité suivante afin d'empêcher le déploiement sur les nœuds hybrides. Ajoutez la règle d'affinité suivante si elle n'est pas déjà présente.

```
affinity:
  nodeAffinity:
```

```
    requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
      - matchExpressions:
        - key: eks.amazonaws.com/compute-type
          operator: NotIn
          values:
          - hybrid
```

Considérations

L'objet `remoteNetworkConfig` JSON se comporte comme suit lors d'une mise à jour :

- Toute partie existante de la configuration que vous ne spécifiez pas reste inchangée. Si vous ne spécifiez ni `remoteNodeNetworks` ni `remotePodNetworks`, cette partie restera inchangée.
- Si vous modifiez les listes de CIDR `remoteNodeNetworks` ou `remotePodNetworks`, vous devez spécifier la liste complète des CIDR que vous souhaitez inclure dans votre configuration finale. Lorsque vous spécifiez une modification à apporter à la liste ou à la liste CIDR `remoteNodeNetworks` ou `remotePodNetworks`, EKS remplace la liste d'origine lors de la mise à jour.
- Les blocs CIDR de vos nœuds et pods sur site doivent répondre aux exigences suivantes :
  1. Se trouver dans l'une des plages IPv4 RFC-1918 suivantes : 10.0.0.0/8, 172.16.0.0/12 ou 192.168.0.0/16.
  2. Ne se chevauchent pas entre eux, tous les CIDR du VPC pour votre cluster Amazon EKS ou votre CIDR IPv4 Kubernetes.

Activation de nœuds hybrides sur un cluster existant

Vous pouvez activer les nœuds hybrides EKS dans un cluster existant en utilisant :

- AWS CloudFormation
- Interface de ligne de commande AWS
- AWS Management Console

Activer les nœuds hybrides EKS dans un cluster existant – AWS CloudFormation

1. Pour activer les nœuds hybrides EKS dans votre cluster, ajoutez les éléments `RemoteNodeNetwork` et (facultatif) `RemotePodNetwork` à votre modèle CloudFormation, puis

mettez à jour la pile. Notez que `RemoteNodeNetwork` est une liste comportant au maximum un élément `Cidrs` et que `Cidrs` est une liste de plusieurs plages CIDR IP.

```
RemoteNetworkConfig:
  RemoteNodeNetworks:
    - Cidrs: [RemoteNodeCIDR]
  RemotePodNetworks:
    - Cidrs: [RemotePodCIDR]
```

2. Passez au the section called "Préparation de l'accès aux clusters".

Activer les nœuds hybrides EKS dans un cluster existant – AWS CLI

1. Exécutez la commande suivante pour activer `RemoteNetworkConfig` pour les nœuds hybrides EKS pour votre cluster EKS. Avant d'exécuter la commande, remplacez ce qui suit par vos paramètres. Pour obtenir la liste complète des paramètres, consultez la section UpdateClusterConfig dans la référence API Amazon EKS.
   a. `CLUSTER_NAME` : nom du cluster EKS à mettre à jour.
   b. `AWS_REGION` : région AWS dans laquelle le cluster EKS est exécuté.
   c. `REMOTE_NODE_CIDRS` : le CIDR du nœud sur site pour vos nœuds hybrides.
   d. `REMOTE_POD_CIDRS` (facultatif) : le CIDR du pod sur site pour les charges de travail exécutées sur des nœuds hybrides.

```
aws eks update-cluster-config \
    --name CLUSTER_NAME \
    --region AWS_REGION \
    --remote-network-config '{"remoteNodeNetworks":[{"cidrs":
["REMOTE_NODE_CIDRS"]}],"remotePodNetworks":[{"cidrs":["REMOTE_POD_CIDRS"]}]}'
```

2. La mise à jour du cluster prend plusieurs minutes. Vous pouvez vérifier le statut de votre cluster avec la commande suivante. Remplacez `CLUSTER_NAME` par le nom du cluster que vous modifiez et `AWS_REGION` par la région AWS dans laquelle le cluster est exécuté. Ne passez pas à l'étape suivante tant que le résultat renvoyé n'est pas `ACTIVE`.

```
aws eks describe-cluster \
    --name CLUSTER_NAME \
    --region AWS_REGION \
    --query "cluster.status"
```

3. Passez au the section called "Préparation de l'accès aux clusters".

Activez les nœuds hybrides EKS dans un cluster existant - AWS Management Console

1. Ouvrez la console Amazon EKS sur la console Amazon EKS.
2. Choisissez le nom du cluster pour afficher les informations le concernant.
3. Sélectionnez l'onglet Réseau, puis Gestion.
4. Dans le menu déroulant, sélectionnez Réseaux distants.
5. Sélectionnez Configurer les réseaux distants pour activer les nœuds hybrides et spécifier votre nœud sur site et les CIDR des pods pour les nœuds hybrides.
6. Choisissez Save changes (Enregistrer les modifications) pour terminer. Attendez que le statut du cluster redevienne Actif.
7. Passez au the section called "Préparation de l'accès aux clusters".

Mettre à jour la configuration des nœuds hybrides dans un cluster existant

Vous pouvez modifier `remoteNetworkConfig` dans un cluster hybride existant à l'aide de l'une des méthodes suivantes :

- AWS CloudFormation
- Interface de ligne de commande AWS
- AWS Management Console

Mettre à jour la configuration hybride dans un cluster existant – AWS CloudFormation

1. Mettez à jour votre modèle CloudFormation avec les nouvelles valeurs CIDR du réseau.

```
RemoteNetworkConfig:
  RemoteNodeNetworks:
    - Cidrs: [NEW_REMOTE_NODE_CIDRS]
  RemotePodNetworks:
    - Cidrs: [NEW_REMOTE_POD_CIDRS]
```

> **Note**
>
> Lors de la mise à jour des listes CIDR `RemoteNodeNetworks` ou `RemotePodNetworks`, incluez tous les CIDR (nouveaux et existants). EKS remplace la liste complète lors des mises à jour. En omettant ces champs dans la demande de mise à jour, leurs configurations existantes sont conservées.

2. Mettez à jour votre pile CloudFormation avec le modèle modifié et attendez que la mise à jour de la pile soit terminée.

Mettre à jour la configuration hybride dans un cluster – AWS CLI

1. Pour modifier les CIDR du réseau distant, exécutez la commande suivante. Remplacez les valeurs par vos paramètres :

```
aws eks update-cluster-config
--name CLUSTER_NAME
--region AWS_REGION
--remote-network-config '{"remoteNodeNetworks":[{"cidrs":
["NEW_REMOTE_NODE_CIDRS"]}],"remotePodNetworks":[{"cidrs":
["NEW_REMOTE_POD_CIDRS"]}]}'
```

> **Note**
>
> Lors de la mise à jour des listes CIDR `remoteNodeNetworks` ou `remotePodNetworks`, incluez tous les CIDR (nouveaux et existants). EKS remplace la liste complète lors des mises à jour. En omettant ces champs dans la demande de mise à jour, leurs configurations existantes sont conservées.

2. Attendez que le statut du cluster revienne à ACTIF avant de continuer.

Mettre à jour la configuration hybride dans un cluster existant - AWS Management Console

1. Ouvrez la console Amazon EKS sur la console Amazon EKS.
2. Choisissez le nom du cluster pour afficher les informations le concernant.
3. Sélectionnez l'onglet Réseau, puis Gestion.

4. Dans le menu déroulant, sélectionnez Réseaux distants.
5. Mettez à jour les CIDR sous `Remote node networks` et `Remote pod networks - Optional` selon les besoins.
6. Choisissez Enregistrer les modifications et attendez que le statut du cluster redevienne actif.

Désactiver les nœuds hybrides dans un cluster existant

Vous pouvez désactiver les nœuds hybrides EKS dans un cluster existant en utilisant :

- AWS CloudFormation
- Interface de ligne de commande AWS
- AWS Management Console

Désactiver les nœuds hybrides EKS dans un cluster existant – AWS CloudFormation

1. Pour désactiver les nœuds hybrides EKS dans votre cluster, définissez `RemoteNodeNetworks` et `RemotePodNetworks` sur des tableaux vides dans votre modèle CloudFormation, puis mettez à jour la pile.

```
RemoteNetworkConfig:
  RemoteNodeNetworks: []
  RemotePodNetworks: []
```

Désactiver les nœuds hybrides EKS dans un cluster existant – AWS CLI

1. Exécutez la commande suivante pour supprimer `RemoteNetworkConfig` de votre cluster EKS. Avant d'exécuter la commande, remplacez ce qui suit par vos paramètres. Pour obtenir la liste complète des paramètres, consultez la section UpdateClusterConfig dans la référence API Amazon EKS.
   a. `CLUSTER_NAME` : nom du cluster EKS à mettre à jour.
   b. `AWS_REGION` : région AWS dans laquelle le cluster EKS est exécuté.

```
aws eks update-cluster-config \
    --name CLUSTER_NAME \
    --region AWS_REGION \
    --remote-network-config '{"remoteNodeNetworks":[],"remotePodNetworks":[]}'
```

2. La mise à jour du cluster prend plusieurs minutes. Vous pouvez vérifier le statut de votre cluster avec la commande suivante. Remplacez `CLUSTER_NAME` par le nom du cluster que vous modifiez et `AWS_REGION` par la région AWS dans laquelle le cluster est exécuté. Ne passez pas à l'étape suivante tant que le résultat renvoyé n'est pas `ACTIVE`.

```
aws eks describe-cluster \
    --name CLUSTER_NAME \
    --region AWS_REGION \
    --query "cluster.status"
```

Désactiver les nœuds hybrides EKS dans un cluster existant – AWS Management Console

1. Ouvrez la console Amazon EKS sur la console Amazon EKS.
2. Choisissez le nom du cluster pour afficher les informations le concernant.
3. Sélectionnez l'onglet Réseau, puis Gestion.
4. Dans le menu déroulant, sélectionnez Réseaux distants.
5. Sélectionnez Configurer les réseaux distants pour activer les nœuds hybrides et supprimer tous les CIDR sous `Remote node networks` et `Remote pod networks - Optional`.
6. Choisissez Save changes (Enregistrer les modifications) pour terminer. Attendez que le statut du cluster redevienne Actif.

Préparation de l'accès au cluster pour les nœuds hybrides

Avant de connecter des nœuds hybrides à votre cluster Amazon EKS, vous devez activer votre rôle IAM pour les nœuds hybrides avec les autorisations Kubernetes afin de rejoindre le cluster. Pour plus d'informations sur la création du rôle IAM de nœuds hybrides, consultez the section called "Préparation des informations d'identification". Amazon EKS propose deux méthodes pour associer les principaux IAM au contrôle d'accès basé sur les rôles (RBAC) de Kubernetes, aux entrées d'accès Amazon EKS et au. `aws-auth` ConfigMap Pour plus d'informations sur la gestion des accès Amazon EKS, consultez the section called "Accès à l'API Kubernetes".

Suivez les procédures ci-dessous pour associer votre rôle IAM des nœuds hybrides aux autorisations Kubernetes. Pour utiliser les entrées d'accès Amazon EKS, votre cluster doit avoir été créé avec les modes d'authentification `API` ou `API_AND_CONFIG_MAP`. Pour utiliser le `aws-auth` ConfigMap, votre cluster doit avoir été créé avec le mode `API_AND_CONFIG_MAP` d'authentification. Le mode

d'authentification CONFIG_MAP-uniquement n'est pas pris en charge pour les clusters Amazon EKS compatibles avec les nœuds hybrides.

Utilisation des entrées d'accès Amazon EKS pour le rôle IAM des nœuds hybrides

Il existe un type d'entrée d'accès Amazon EKS pour les nœuds hybrides nommé HYBRID_LINUX qui peut être utilisé avec un rôle IAM. Avec ce type d'entrée d'accès, le nom d'utilisateur est automatiquement défini sur system:node : {{SessionName}}. Pour plus d'informations sur la création d'entrées d'accès, voir the section called "Créer des entrées d'accès".

AWS CLI

1. La dernière version de la AWS CLI doit être installée et configurée sur votre appareil. Pour vérifier votre version actuelle, utilisez `aws --version`. Les gestionnaires de packages tels que yum, apt-get ou Homebrew pour macOS ont souvent plusieurs versions de retard sur la dernière version de la CLI. AWS Pour installer la dernière version, consultez la section Installation et configuration rapide avec aws configure dans le Guide de l'utilisateur de l'interface de ligne de AWS commande.
2. Créez votre entrée d'accès à l'aide de la commande suivante. Remplacez CLUSTER_NAME par le nom de votre cluster et HYBRID_NODES_ROLE_ARN par l'ARN du rôle que vous avez créé dans les étapes précédentes pour the section called "Préparation des informations d'identification".

```
aws eks create-access-entry --cluster-name CLUSTER_NAME \
    --principal-arn HYBRID_NODES_ROLE_ARN \
    --type HYBRID_LINUX
```

AWS Management Console

1. Ouvrez la console Amazon EKS à l'adresse Amazon EKS console.
2. Choisissez le nom de votre cluster compatible avec les nœuds hybrides.
3. Choisissez l'onglet Access.
4. Choisissez Créer une entrée d'accès.
5. Pour le principal IAM, sélectionnez le rôle IAM des nœuds hybrides que vous avez créé dans les étapes pour the section called "Préparation des informations d'identification".
6. Pour Type, sélectionnez Linux hybride.
7. (Facultatif) Pour Balises, attribuez des étiquettes à l'entrée d'accès. Par exemple, pour faciliter la recherche de toutes les ressources portant la même balise.

8. Choisissez Passer à la révision et à la création. Vous ne pouvez pas ajouter de stratégies à l'entrée d'accès Linux hybride ni modifier sa portée d'accès.
9. Vérifiez la configuration de votre entrée d'accès. Si quelque chose semble incorrect, choisissez Précédent pour revenir en arrière et corriger l'erreur. Si la configuration est correcte, choisissez Créer.

Utilisation d'aws-auth ConfigMap pour le rôle IAM des nœuds hybrides

Dans les étapes suivantes, vous allez créer ou mettre à jour le `aws-auth` ConfigMap avec l'ARN du rôle IAM des nœuds hybrides que vous avez créé dans les étapes pour the section called "Préparation des informations d'identification" lesquelles.

1. Vérifiez si vous en avez déjà un `aws-auth` ConfigMap pour votre cluster. Notez que si vous utilisez un fichier `kubeconfig` spécifique, utilisez le drapeau `--kubeconfig`.

```
kubectl describe configmap -n kube-system aws-auth
```

2. Si un s'affiche `aws-auth` ConfigMap, mettez-le à jour si nécessaire.
   a. Ouvrez le ConfigMap pour le modifier.

```
kubectl edit -n kube-system configmap/aws-auth
```

   b. Ajoutez une nouvelle entrée `mapRoles` si nécessaire. Remplacez `HYBRID_NODES_ROLE_ARN` par l'ARN de votre rôle IAM des nœuds hybrides. Remarque : `{{SessionName}}` le format de modèle à enregistrer dans le ConfigMap. Ne le remplacez pas par d'autres valeurs.

```
data:
  mapRoles: |
  - groups:
    - system:bootstrappers
    - system:nodes
    rolearn: HYBRID_NODES_ROLE_ARN
    username: system:node:{{SessionName}}
```

   c. Enregistrez le fichier et quittez votre éditeur de texte.
3. S'il n'en existe pas `aws-auth` ConfigMap pour votre cluster, créez-le à l'aide de la commande suivante. Remplacez `HYBRID_NODES_ROLE_ARN` par l'ARN de votre rôle IAM des nœuds hybrides. Notez qu'il `{{SessionName}}` s'agit du format de modèle correct à enregistrer dans le ConfigMap. Ne le remplacez pas par d'autres valeurs.

```
kubectl apply -f=/dev/stdin <<-EOF
apiVersion: v1
kind: ConfigMap
metadata:
  name: aws-auth
  namespace: kube-system
data:
  mapRoles: |
  - groups:
    - system:bootstrappers
    - system:nodes
    rolearn: HYBRID_NODES_ROLE_ARN
    username: system:node:{{SessionName}}
EOF
```

Exécution de charges de travail sur site sur des nœuds hybrides

Dans un cluster EKS avec des nœuds hybrides activés, vous pouvez exécuter des applications sur site et en périphérie sur votre propre infrastructure avec les mêmes clusters, fonctionnalités et outils Amazon EKS que ceux que vous utilisez dans le cloud AWS.

Les sections suivantes contiennent des instructions détaillées pour l'utilisation des nœuds hybrides.

Rubriques

- Connecter les nœuds hybrides
- Connectez des nœuds hybrides avec Bottlerocket
- Mettre à niveau les nœuds hybrides pour votre cluster
- Mises à jour de sécurité pour les nœuds hybrides
- Supprimer les nœuds hybrides

Connecter les nœuds hybrides

> **Note**
>
> Les étapes suivantes s'appliquent aux nœuds hybrides exécutant des systèmes d'exploitation compatibles, à l'exception de Bottlerocket. Pour connaître les étapes à suivre pour connecter

un nœud hybride qui exécute Bottlerocket, consultez the section called "Connectez des nœuds hybrides avec Bottlerocket".

Cette rubrique décrit comment connecter des nœuds hybrides à un cluster Amazon EKS. Une fois que vos nœuds hybrides ont rejoint le cluster, ils apparaissent avec le statut Not Ready (Non prêt) dans la console Amazon EKS et dans les outils compatibles avec Kubernetes, tels que kubectl. Une fois les étapes décrites sur cette page terminées, passez à la section the section called "Configurer CNI" pour préparer vos nœuds hybrides à exécuter des applications.

Prérequis

Avant de connecter des nœuds hybrides à votre cluster Amazon EKS, assurez-vous d'avoir suivi toutes les étapes préalables.

- Vous disposez d'une connectivité réseau entre votre environnement sur site et la région AWS hébergeant votre cluster Amazon EKS. Pour plus d'informations, consultez the section called "Préparez votre réseau".
- Vous disposez d'un système d'exploitation compatible pour les nœuds hybrides installés sur vos hôtes sur site. Pour plus d'informations, consultez the section called "Préparer le système d'exploitation".
- Vous avez créé votre rôle IAM des nœuds hybrides et configuré votre fournisseur d'informations d'identification sur site (activations hybrides AWS Systems Manager ou Rôles Anywhere IAM AWS). Pour plus d'informations, consultez the section called "Préparation des informations d'identification".
- Vous avez créé votre cluster Amazon EKS compatible avec les nœuds hybrides. Pour plus d'informations, consultez the section called "Créer un cluster".
- Vous avez associé votre rôle IAM des nœuds hybrides aux autorisations RBAC de Kubernetes. Pour plus d'informations, consultez the section called "Préparation de l'accès aux clusters".

Étape 1 : installer l'interface CLI des nœuds hybrides (**`nodeadm`**) sur chaque hôte sur site

Si vous incluez l'interface CLI des nœuds hybrides Amazon EKS (`nodeadm`) dans vos images de système d'exploitation pré-construites, vous pouvez ignorer cette étape. Pour plus d'informations sur la version hybride des nœuds de `nodeadm`, consultez the section called "Nœuds hybrides nodeadm".

La version hybride des nœuds de `nodeadm` est hébergée dans Amazon S3, avec Amazon CloudFront comme interface. Pour installer `nodeadm` sur chaque hôte sur site, vous pouvez exécuter la commande suivante à partir de vos hôtes sur site.

Pour les hôtes x86_64 :

```
curl -OL 'https://hybrid-assets.eks.amazonaws.com/releases/latest/bin/linux/amd64/nodeadm'
```

Pour les hôtes ARM

```
curl -OL 'https://hybrid-assets.eks.amazonaws.com/releases/latest/bin/linux/arm64/nodeadm'
```

Ajoutez les permissions d'exécution au fichier binaire téléchargé sur chaque hôte.

```
chmod +x nodeadm
```

Étape 2 : installer les dépendances des nœuds hybrides avec **`nodeadm`**

Si vous installez les dépendances des nœuds hybrides dans des images de système d'exploitation précompilées, vous pouvez ignorer cette étape. La commande `nodeadm install` peut être utilisée pour installer toutes les dépendances requises pour les nœuds hybrides. Les dépendances des nœuds hybrides incluent containerd, kubelet, kubectl et les composants AWS SSM ou Rôles Anywhere IAM AWS. Pour plus d'informations sur les composants et les emplacements des fichiers installés par `nodeadm install`, consultez the section called "Nœuds hybrides nodeadm". Pour plus d'informations sur les domaines qui doivent être autorisés dans votre pare-feu local pour le processus `nodeadm install`, consultez the section called "Préparez votre réseau".

Exécutez la commande ci-dessous pour installer les dépendances des nœuds hybrides sur votre hôte local. La commande ci-dessous doit être exécutée par un utilisateur disposant d'un accès sudo/root sur votre hôte.

> **Important**
>
> Les nœuds hybrides CLI (`nodeadm`) doivent être exécutés avec un utilisateur disposant d'un accès sudo/root sur votre hôte.

- Remplacez `K8S_VERSION` par la version mineure Kubernetes de votre cluster Amazon EKS, par exemple `1.31`. Consultez les versions prises en charge par Amazon EKS pour obtenir la liste des versions Kubernetes prises en charge.
- Remplacez `CREDS_PROVIDER` par le fournisseur d'informations d'identification local que vous utilisez. Les valeurs valides sont `ssm` pour AWS SSM et `iam-ra` pour Rôles Anywhere IAM AWS.

```
nodeadm install K8S_VERSION --credential-provider CREDS_PROVIDER
```

Étape 3 : connecter les nœuds hybrides à votre cluster

Avant de connecter vos nœuds hybrides à votre cluster, assurez-vous d'avoir autorisé l'accès requis dans votre pare-feu local et dans le groupe de sécurité de votre cluster pour la communication entre le plan de contrôle Amazon EKS et les nœuds hybrides. La plupart des problèmes rencontrés à cette étape sont liés à la configuration du pare-feu, à la configuration du groupe de sécurité ou à la configuration du rôle IAM des nœuds hybrides.

> **Important**
>
> Les nœuds hybrides CLI (`nodeadm`) doivent être exécutés avec un utilisateur disposant d'un accès sudo/root sur votre hôte.

1. Créez un fichier `nodeConfig.yaml` sur chaque hôte avec les valeurs pour votre déploiement. Pour une description complète des paramètres de configuration disponibles, voir the section called "Nœuds hybrides nodeadm". Si votre rôle IAM des nœuds hybrides ne dispose pas des autorisations nécessaires pour l'action `eks:DescribeCluster`, vous devez transmettre votre point de terminaison de l'API Kubernetes, votre offre groupée CA et votre CIDR IPv4 du service Kubernetes dans la section cluster de votre `nodeConfig.yaml`.
   a. Utilisez l'exemple `nodeConfig.yaml` ci-dessous si vous utilisez des activations hybrides SSM AWS pour votre fournisseur d'informations d'identification sur site.
      i. Remplacez `CLUSTER_NAME` par le nom de votre cluster.
      ii. Remplacez `AWS_REGION` par la région AWS hébergeant votre cluster. Par exemple, `us-west-2`.

iii. Remplacez `ACTIVATION_CODE` par le code d'activation que vous avez reçu lors de la création de votre activation hybride SSM AWS. Pour plus d'informations, consultez the section called "Préparation des informations d'identification".

iv. Remplacez `ACTIVATION_ID` par l'ID d'activation que vous avez reçu lors de la création de votre activation hybride SSM AWS. Vous pouvez récupérer ces informations à partir de la console AWS Systems Manager ou à partir de la commande AWS CLI `aws ssm describe-activations`.

```
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
    name: CLUSTER_NAME
    region: AWS_REGION
  hybrid:
    ssm:
      activationCode: ACTIVATION_CODE
      activationId: ACTIVATION_ID
```

b. Utilisez l'exemple `nodeConfig.yaml` ci-dessous si vous utilisez Rôles Anywhere IAM AWS pour votre fournisseur d'informations d'identification sur site.

i. Remplacez `CLUSTER_NAME` par le nom de votre cluster.

ii. Remplacez `AWS_REGION` par la région AWS hébergeant votre cluster. Par exemple, `us-west-2`.

iii. Remplacez `NODE_NAME` par le nom de votre nœud. Le nom du nœud doit correspondre au nom commun (CN) du certificat sur l'hôte si vous avez configuré la stratégie de confiance de votre rôle IAM des nœuds hybrides avec la condition de ressource `"sts:RoleSessionName": "${aws:PrincipalTag/x509Subject/CN}"`. Le `nodeName` que vous utilisez ne doit pas dépasser 64 caractères.

iv. Remplacez `TRUST_ANCHOR_ARN` par l'ARN de l'ancre d'approbation que vous avez configurée dans les étapes de la section Préparer les informations d'identification pour les nœuds hybrides.

v. Remplacez `PROFILE_ARN` par l'ARN de l'ancre d'approbation que vous avez configurée dans les étapes pour the section called "Préparation des informations d'identification".

vi. Remplacez `ROLE_ARN` par l'ARN de votre rôle IAM des nœuds hybrides.

vii. Remplacez `CERTIFICATE_PATH` par le chemin d'accès sur le disque vers votre certificat de nœud. Si vous ne le précisez pas, la valeur par défaut est `/etc/iam/pki/server.pem`.

viiiRemplacez KEY_PATH par le chemin d'accès sur le disque vers la clé privée de votre certificat. Si vous ne le précisez pas, la valeur par défaut est `/etc/iam/pki/server.key`.

```
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
    name: CLUSTER_NAME
    region: AWS_REGION
  hybrid:
    iamRolesAnywhere:
      nodeName: NODE_NAME
      trustAnchorArn: TRUST_ANCHOR_ARN
      profileArn: PROFILE_ARN
      roleArn: ROLE_ARN
      certificatePath: CERTIFICATE_PATH
      privateKeyPath: KEY_PATH
```

2. Exécutez la commande `nodeadm init` avec votre `nodeConfig.yaml` pour connecter vos nœuds hybrides à votre cluster Amazon EKS.

```
nodeadm init -c file://nodeConfig.yaml
```

Si la commande ci-dessus s'exécute correctement, votre nœud hybride a rejoint votre cluster Amazon EKS. Vous pouvez vérifier cela dans la console Amazon EKS en accédant à l'onglet Calcul de votre cluster (assurez-vous que le principal IAM dispose des autorisations nécessaires pour afficher) ou avec `kubectl get nodes`.

> **Important**
>
> Vos nœuds auront le statut `Not Ready`, ce qui est normal et dû à l'absence d'un CNI fonctionnant sur vos nœuds hybrides. Si vos nœuds ne se sont pas joints au cluster, consultez the section called "Résolution des problèmes".

Étape 4 : configurer un CNI pour les nœuds hybrides

Pour que vos nœuds hybrides soient prêts à exécuter des applications, poursuivez avec les étapes décrites à la section the section called "Configurer CNI".

Connectez des nœuds hybrides avec Bottlerocket

Cette rubrique décrit comment connecter des nœuds hybrides exécutant Bottlerocket à un cluster Amazon EKS. Bottlerocket est une distribution Linux open source sponsorisée et soutenue par. AWS Bottlerocket est spécialement conçu pour héberger des charges de travail conteneurisées. Avec Bottlerocket, vous pouvez améliorer la disponibilité des déploiements conteneurisés et réduire les coûts opérationnels en automatisant les mises à jour de votre infrastructure de conteneurs. Bottlerocket ne comprend que les logiciels essentiels à l'exécution des conteneurs, ce qui améliore l'utilisation des ressources, réduit les menaces de sécurité et diminue les frais généraux liés à la gestion.

Seules les VMware variantes de la version v1.37.0 et supérieures de Bottlerocket sont prises en charge avec les nœuds hybrides EKS. VMware des variantes de Bottlerocket sont disponibles pour les versions v1.28 et supérieures de Kubernetes. Les images du système d'exploitation de ces variantes incluent le kubelet, le containerd aws-iam-authenticator et d'autres logiciels requis pour les nœuds hybrides EKS. Vous pouvez configurer ces composants à l'aide d'un fichier de paramètres Bottlerocket qui comprend des données utilisateur encodées en base64 pour les conteneurs de démarrage et d'administration Bottlerocket. La configuration de ces paramètres permet à Bottlerocket d'utiliser votre fournisseur d'informations d'identification de nœuds hybrides pour authentifier les nœuds hybrides sur votre cluster. Une fois que vos nœuds hybrides ont rejoint le cluster, leur statut `Not Ready` s'affiche dans la console Amazon EKS et dans les outils compatibles avec Kubernetes, tels que `kubectl`. Une fois les étapes décrites sur cette page terminées, passez à la section the section called "Configurer CNI" pour préparer vos nœuds hybrides à exécuter des applications.

Conditions préalables

Avant de connecter des nœuds hybrides à votre cluster Amazon EKS, assurez-vous d'avoir suivi toutes les étapes préalables.

- Vous disposez d'une connectivité réseau entre votre environnement sur site et la AWS région hébergeant votre cluster Amazon EKS. Pour plus d'informations, consultez the section called "Préparez votre réseau".
- Vous avez créé votre rôle Hybrid Nodes IAM et configuré votre fournisseur d'informations d'identification sur site (activations hybrides de AWS Systems Manager ou AWS IAM Roles Anywhere). Pour plus d'informations, consultez the section called "Préparation des informations d'identification".
- Vous avez créé votre cluster Amazon EKS compatible avec les nœuds hybrides. Pour plus d'informations, consultez the section called "Créer un cluster".

- Vous avez associé votre rôle IAM des nœuds hybrides aux autorisations RBAC de Kubernetes. Pour plus d'informations, consultez the section called "Préparation de l'accès aux clusters".

Étape 1 : créer le fichier TOML des paramètres Bottlerocket

Pour configurer Bottlerocket pour les nœuds hybrides, vous devez créer un fichier `settings.toml` contenant la configuration nécessaire. Le contenu du fichier TOML varie en fonction du fournisseur d'informations d'identification que vous utilisez (SSM ou Rôles Anywhere IAM). Ce fichier sera transmis en tant que données utilisateur lors de la mise à disposition de l'instance Bottlerocket.

> **Note**
>
> Les fichiers TOML fournis ci-dessous ne représentent que les paramètres minimaux requis pour initialiser une VMWare machine Bottlerocket en tant que nœud sur un cluster EKS. Bottlerocket fournit une large gamme de paramètres pour répondre à différents cas d'utilisation. Pour d'autres options de configuration au-delà de l'initialisation du nœud hybride, veuillez consulter la documentation de Bottlerocket pour obtenir la liste complète de tous les paramètres documentés pour la version de Bottlerocket que vous utilisez (par exemple, voici tous les paramètres disponibles pour Bottlerocket 1.51.x).

SSM

Si vous utilisez AWS Systems Manager comme fournisseur d'informations d'identification, créez un `settings.toml` fichier avec le contenu suivant :

```
[settings.kubernetes]
cluster-name = "<cluster-name>"
api-server = "<api-server-endpoint>"
cluster-certificate = "<cluster-certificate-authority>"
hostname-override = "<hostname>"
provider-id = "eks-hybrid:///<region>/<cluster-name>/<hostname>"
authentication-mode = "aws"
cloud-provider = ""
server-tls-bootstrap = true

[settings.network]
hostname = "<hostname>"

[settings.aws]
```

```
region = "<region>"

[settings.kubernetes.credential-providers.ecr-credential-provider]
enabled = true
cache-duration = "12h"
image-patterns = [
    "*.dkr.ecr.*.amazonaws.com",
    "*.dkr.ecr.*.amazonaws.com.cn",
    "*.dkr.ecr.*.amazonaws.eu",
    "*.dkr.ecr-fips.*.amazonaws.com",
    "*.dkr.ecr-fips.*.amazonaws.eu",
    "public.ecr.aws"
]

[settings.kubernetes.node-labels]
"eks.amazonaws.com/compute-type" = "hybrid"
"eks.amazonaws.com/hybrid-credential-provider" = "ssm"

[settings.host-containers.admin]
enabled = true
user-data = "<base64-encoded-admin-container-userdata>"

[settings.bootstrap-containers.eks-hybrid-setup]
mode = "always"
user-data = "<base64-encoded-bootstrap-container-userdata>"

[settings.host-containers.control]
enabled = true
```

Remplacez les espaces réservés par les valeurs suivantes :

- `<cluster-name>` : le nom de votre cluster Amazon EKS.
- `<api-server-endpoint>` : point de terminaison du serveur d'API de votre cluster.
- `<cluster-certificate-authority>` : Ensemble CA codé en base64 de votre cluster.
- `<region>`: La AWS région hébergeant votre cluster, par exemple « us-east-1 ».
- `<hostname>` : le nom d'hôte de l'instance Bottlerocket, qui sera également configuré comme nom de nœud. Il peut s'agir de n'importe quelle valeur unique de votre choix, mais elle doit respecter les conventions de nommage des objets Kubernetes. De plus, le nom d'hôte que vous utilisez ne peut pas dépasser 64 caractères. REMARQUE : lorsque vous utilisez le fournisseur SSM, ce nom d'hôte et ce nom de nœud seront remplacés par l'ID de l'instance gérée (par exemple, ID `mi-*`) une fois que l'instance aura été enregistrée auprès de SSM.

- `<base64-encoded-admin-container-userdata>` : Contenu codé en base64 de la configuration du conteneur d'administration Bottlerocket. L'activation du conteneur d'administration vous permet de vous connecter à votre instance Bottlerocket avec SSH pour explorer le système et le déboguer. Bien que ce paramètre ne soit pas obligatoire, nous vous recommandons de l'activer afin de faciliter le dépannage. Pour plus d'informations sur l'authentification avec le conteneur d'administration, consultez la documentation relative au conteneur d'administration Bottlerocket. Le conteneur d'administration accepte les entrées utilisateur et clé SSH au format JSON, par exemple :

```
{
  "user": "<ssh-user>",
  "ssh": {
    "authorized-keys": [
      "<ssh-authorized-key>"
    ]
  }
}
```

- `<base64-encoded-bootstrap-container-userdata>` : Contenu encodé en base64 de la configuration du conteneur d'amorçage Bottlerocket. Pour plus d'informations sur sa configuration, consultez la documentation relative au conteneur Bottlerocket bootstrap. Le conteneur bootstrap est chargé d'enregistrer l'instance en tant qu'instance gérée par AWS SSM et de la joindre en tant que nœud Kubernetes sur votre cluster Amazon EKS. Les données utilisateur transmises au conteneur Bootstrap prennent la forme d'une invocation de commande qui accepte en entrée le code d'activation hybride SSM et l'ID que vous avez créés précédemment :

```
eks-hybrid-ssm-setup --activation-id=<activation-id> --activation-code=<activation-
code> --region=<region>
```

Rôles Anywhere IAM

Si vous utilisez AWS IAM Roles Anywhere comme fournisseur d'informations d'identification, créez un `settings.toml` fichier avec le contenu suivant :

```
[settings.kubernetes]
cluster-name = "<cluster-name>"
api-server = "<api-server-endpoint>"
cluster-certificate = "<cluster-certificate-authority>"
```

```
hostname-override = "<hostname>"
provider-id = "eks-hybrid:///<region>/<cluster-name>/<hostname>"
authentication-mode = "aws"
cloud-provider = ""
server-tls-bootstrap = true

[settings.network]
hostname = "<hostname>"

[settings.aws]
region = "<region>"
config = "<base64-encoded-aws-config-file>"

[settings.kubernetes.credential-providers.ecr-credential-provider]
enabled = true
cache-duration = "12h"
image-patterns = [
    "*.dkr.ecr.*.amazonaws.com",
    "*.dkr.ecr.*.amazonaws.com.cn",
    "*.dkr.ecr.*.amazonaws.eu",
    "*.dkr.ecr-fips.*.amazonaws.com",
    "*.dkr.ecr-fips.*.amazonaws.eu",
    "public.ecr.aws"
]

[settings.kubernetes.node-labels]
"eks.amazonaws.com/compute-type" = "hybrid"
"eks.amazonaws.com/hybrid-credential-provider" = "iam-ra"

[settings.host-containers.admin]
enabled = true
user-data = "<base64-encoded-admin-container-userdata>"

[settings.bootstrap-containers.eks-hybrid-setup]
mode = "always"
user-data = "<base64-encoded-bootstrap-container-userdata>"
```

Remplacez les espaces réservés par les valeurs suivantes :

- `<cluster-name>` : le nom de votre cluster Amazon EKS.
- `<api-server-endpoint>` : point de terminaison du serveur d'API de votre cluster.
- `<cluster-certificate-authority>` : Ensemble CA codé en base64 de votre cluster.

- `<region>`: La AWS région hébergeant votre cluster, par exemple « us-east-1 »
- `<hostname>` : le nom d'hôte de l'instance Bottlerocket, qui sera également configuré comme nom de nœud. Il peut s'agir de n'importe quelle valeur unique de votre choix, mais elle doit respecter les conventions de nommage des objets Kubernetes. De plus, le nom d'hôte que vous utilisez ne peut pas dépasser 64 caractères. REMARQUE : lorsque vous utilisez le fournisseur IAM-RA, le nom du nœud doit correspondre au nom commun (CN) du certificat sur l'hôte si vous avez configuré la stratégie de confiance de votre rôle IAM des nœuds hybrides avec la condition de ressource `"sts:RoleSessionName": "${aws:PrincipalTag/x509Subject/CN}"`.
- `<base64-encoded-aws-config-file>`: le contenu codé en base64 de votre AWS fichier de configuration. Le contenu du fichier doit être le suivant :

```
[default]
credential_process = aws_signing_helper credential-process --certificate /root/.aws/
node.crt --private-key /root/.aws/node.key --profile-arn <profile-arn> --role-arn
 <role-arn> --trust-anchor-arn <trust-anchor-arn> --role-session-name <role-session-
name>
```

- `<base64-encoded-admin-container-userdata>` : Contenu codé en base64 de la configuration du conteneur d'administration Bottlerocket. L'activation du conteneur d'administration vous permet de vous connecter à votre instance Bottlerocket avec SSH pour explorer le système et le déboguer. Bien que ce paramètre ne soit pas obligatoire, nous vous recommandons de l'activer afin de faciliter le dépannage. Pour plus d'informations sur l'authentification avec le conteneur d'administration, consultez la documentation relative au conteneur d'administration Bottlerocket. Le conteneur d'administration accepte les entrées utilisateur et clé SSH au format JSON, par exemple :

```
{
  "user": "<ssh-user>",
  "ssh": {
    "authorized-keys": [
      "<ssh-authorized-key>"
    ]
  }
}
```

- `<base64-encoded-bootstrap-container-userdata>` : Contenu encodé en base64 de la configuration du conteneur d'amorçage Bottlerocket. Pour plus d'informations sur sa configuration, consultez la documentation relative au conteneur Bottlerocket bootstrap. Le conteneur Bootstrap est chargé de créer les fichiers de certificat hôte et de clé privée de certificat Rôles Anywhere IAM sur l'instance. Ceux-ci seront ensuite utilisés par le `aws_signing_helper` pour obtenir des informations d'identification temporaires afin de s'authentifier auprès de votre cluster Amazon EKS. Les données utilisateur transmises au conteneur bootstrap prennent la forme d'une invocation de commande qui accepte en entrée le contenu du certificat et de la clé privée que vous avez créés précédemment :

```
eks-hybrid-iam-ra-setup --certificate=<certificate> --key=<private-key>
```

Étape 2 : fournir les données utilisateur à la machine virtuelle Bottlerocket vSphere

Une fois le fichier TOML créé, transmettez-le en tant que données utilisateur lors de la création de la machine virtuelle vSphere. N'oubliez pas que les données utilisateur doivent être configurées avant la première mise sous tension de la machine virtuelle. Vous devrez donc le fournir lors de la création de l'instance. Si vous souhaitez créer la VM à l'avance, celle-ci devra être dans un état « poweredOff » jusqu'à ce que vous configuriez les données utilisateur correspondantes. Par exemple, si vous utilisez l'interface CLI `govc` :

Création d'une machine virtuelle pour la première fois

```
govc vm.create \
  -on=true \
  -c=2 \
  -m=4096 \
  -net.adapter=<network-adapter> \
  -net=<network-name> \
  -e guestinfo.userdata.encoding="base64" \
  -e guestinfo.userdata="$(base64 -w0 settings.toml)" \
  -template=<template-name> \
  <vm-name>
```

Mise à jour des données utilisateur pour une machine virtuelle existante

```
govc vm.create \
    -on=false \
```

```
    -c=2 \
    -m=4096 \
    -net.adapter=<network-adapter> \
    -net=<network-name> \
    -template=<template-name> \
    <vm-name>

govc vm.change
    -vm <vm-name> \
    -e guestinfo.userdata="$(base64 -w0 settings.toml)" \
    -e guestinfo.userdata.encoding="base64"

govc vm.power -on <vm-name>
```

Dans les sections ci-dessus, l'option `-e guestinfo.userdata.encoding="base64"` spécifie que les données utilisateur sont encodées en base64. Cette option `-e guestinfo.userdata` transmet le contenu du fichier `settings.toml` encodé en base64 sous forme de données utilisateur à l'instance Bottlerocket. Remplacez les espaces réservés par vos valeurs spécifiques, telles que le modèle OVA Bottlerocket et les détails réseau.

Étape 3 : vérifier la connexion du nœud hybride

Une fois l'instance Bottlerocket démarrée, elle tentera de rejoindre votre cluster Amazon EKS. Vous pouvez vérifier la connexion dans la console Amazon EKS en accédant à l'onglet Calcul de votre cluster ou en exécutant la commande suivante :

```
kubectl get nodes
```

> **Important**
>
> Vos nœuds auront le statut `Not Ready`, ce qui est normal et dû à l'absence d'un CNI fonctionnant sur vos nœuds hybrides. Si vos nœuds ne se sont pas joints au cluster, consultez the section called "Résolution des problèmes".

Étape 4 : configurer un CNI pour les nœuds hybrides

Pour que vos nœuds hybrides soient prêts à exécuter des applications, poursuivez avec les étapes décrites à la section the section called "Configurer CNI".

Mettre à niveau les nœuds hybrides pour votre cluster

Les instructions relatives à la mise à niveau des nœuds hybrides sont similaires à celles des nœuds Amazon EKS autogérés qui s'exécutent dans Amazon EC2. Nous vous recommandons de créer de nouveaux nœuds hybrides sur votre version Kubernetes cible, de migrer en douceur vos applications existantes vers les nœuds hybrides de la nouvelle version Kubernetes, puis de supprimer les nœuds hybrides de l'ancienne version Kubernetes de votre cluster. Avant de lancer une mise à niveau, veillez à consulter les meilleures pratiques Amazon EKS en matière de mise à niveau. Les nœuds hybrides Amazon EKS prennent en charge la même version de Kubernetes que les clusters Amazon EKS avec nœuds cloud, y compris la prise en charge standard et étendue.

Les nœuds hybrides Amazon EKS suivent la même politique de distorsion des versions pour les nœuds que pour Kubernetes en amont. Les nœuds hybrides Amazon EKS ne peuvent pas être d'une version plus récente que le plan de contrôle Amazon EKS, et les nœuds hybrides peuvent être jusqu'à trois versions mineures Kubernetes plus anciennes que la version mineure du plan de contrôle Amazon EKS.

Si vous ne disposez pas de capacité suffisante pour créer de nouveaux nœuds hybrides sur votre version Kubernetes cible dans le cadre d'une stratégie de migration par basculement, vous pouvez également utiliser l'interface CLI nœuds hybrides Amazon EKS (`nodeadm`) pour mettre à niveau la version Kubernetes de vos nœuds hybrides sur place.

> **Important**
>
> Si vous mettez à niveau vos nœuds hybrides sur place avec `nodeadm`, il y a une durée d'indisponibilité pour le nœud pendant le processus où l'ancienne version des composants Kubernetes est arrêtée et où les composants de la nouvelle version Kubernetes sont installés et démarrés.

Prérequis

Avant de procéder à la mise à niveau, assurez-vous d'avoir rempli les conditions préalables suivantes.

- La version Kubernetes cible pour la mise à niveau de vos nœuds hybrides doit être égale ou inférieure à la version du plan de contrôle Amazon EKS.
- Si vous suivez une stratégie de migration par basculement, les nouveaux nœuds hybrides que vous installez sur votre version Kubernetes cible doivent répondre aux exigences the section called

"Prérequis". Cela inclut le fait d'avoir des adresses IP dans le CIDR du réseau de nœuds distants que vous avez transmis lors de la création du cluster Amazon EKS.

- Pour les migrations par basculement et les mises à niveau sur site, les nœuds hybrides doivent avoir accès aux domaines requis pour extraire les nouvelles versions des dépendances des nœuds hybrides.
- Vous devez avoir installé kubectl sur votre machine locale ou l'instance que vous utilisez pour interagir avec votre point de terminaison API Amazon EKS Kubernetes.
- La version de votre CNI doit prendre en charge la version de Kubernetes vers laquelle vous effectuez la mise à niveau. Si ce n'est pas le cas, mettez à niveau votre version CNI avant de mettre à niveau vos nœuds hybrides. Pour plus d'informations, consultez the section called "Configurer CNI".

Mises à niveau liées à la migration par transition (bleu-vert)

Les mises à niveau par migration cutover désignent le processus consistant à créer de nouveaux nœuds hybrides sur de nouveaux hôtes avec votre version Kubernetes cible, à migrer en douceur vos applications existantes vers les nouveaux nœuds hybrides sur votre version Kubernetes cible, et à supprimer les nœuds hybrides sur l'ancienne version Kubernetes de votre cluster. Cette stratégie est également appelée migration bleu-vert.

1. Connectez vos nouveaux hôtes en tant que nœuds hybrides en suivant les étapes the section called "Connecter les nœuds hybrides". Lorsque vous exécutez la commande `nodeadm install`, utilisez votre version cible de Kubernetes.
2. Activez la communication entre les nouveaux nœuds hybrides sur la version Kubernetes cible et vos nœuds hybrides sur l'ancienne version Kubernetes. Cette configuration permet aux pods de communiquer entre eux pendant que vous migrez votre charge de travail vers les nœuds hybrides sur la version Kubernetes cible.
3. Vérifiez que vos nœuds hybrides sur votre version Kubernetes cible ont bien rejoint votre cluster et ont le statut Ready.
4. Utilisez la commande suivante pour marquer chacun des nœuds que vous souhaitez supprimer comme non planifiable. Cela permet d'éviter que de nouveaux pods ne soient planifiés ou replanifiés sur les nœuds que vous remplacez. Pour plus d'informations, consultez le cordon kubelet dans la documentation Kubernetes. Remplacez `NODE_NAME` par le nom des nœuds hybrides de l'ancienne version de Kubernetes.

```
kubectl cordon NODE_NAME
```

Vous pouvez identifier et isoler tous les nœuds d'une version particulière de Kubernetes (dans ce cas, `1.28`) à l'aide de l'extrait de code suivant.

```
K8S_VERSION=1.28
for node in $(kubectl get nodes -o json | jq --arg K8S_VERSION
 "$K8S_VERSION" -r '.items[] | select(.status.nodeInfo.kubeletVersion |
 match("\($K8S_VERSION)")).metadata.name')
do
    echo "Cordoning $node"
    kubectl cordon $node
done
```

5. Si votre déploiement actuel exécute moins de deux réplicas CoreDNS sur vos nœuds hybrides, augmentez horizontalement la capacité du déploiement à au moins deux réplicas. Nous vous recommandons d'exécuter au moins deux réplicas CoreDNS sur des nœuds hybrides afin d'assurer la résilience pendant les opérations normales.

```
kubectl scale deployments/coredns --replicas=2 -n kube-system
```

6. Videz chacun des nœuds hybrides de l'ancienne version de Kubernetes que vous souhaitez supprimer de votre cluster à l'aide de la commande suivante. Pour plus d'informations sur la mise hors tension des nœuds, consultez la section Mettre hors tension un nœud en toute sécurité dans la documentation Kubernetes. Remplacez `NODE_NAME` par le nom des nœuds hybrides de l'ancienne version de Kubernetes.

```
kubectl drain NODE_NAME --ignore-daemonsets --delete-emptydir-data
```

Vous pouvez identifier et vider tous les nœuds d'une version particulière de Kubernetes (dans ce cas, `1.28`) à l'aide de l'extrait de code suivant.

```
K8S_VERSION=1.28
for node in $(kubectl get nodes -o json | jq --arg K8S_VERSION
 "$K8S_VERSION" -r '.items[] | select(.status.nodeInfo.kubeletVersion |
 match("\($K8S_VERSION)")).metadata.name')
do
    echo "Draining $node"
    kubectl drain $node --ignore-daemonsets --delete-emptydir-data
done
```

7. Vous pouvez utiliser `nodeadm` pour arrêter et supprimer les artefacts des nœuds hybrides de l'hôte. Vous devez exécuter `nodeadm` avec un utilisateur disposant des privilèges root/sudo. Par défaut, `nodeadm uninstall` ne continuera pas s'il reste des pods sur le nœud. Pour plus d'informations, consultez the section called "Nœuds hybrides nodeadm".

```
nodeadm uninstall
```

8. Une fois les artefacts des nœuds hybrides arrêtés et désinstallés, supprimez la ressource du nœud de votre cluster.

```
kubectl delete node node-name
```

   Vous pouvez identifier et supprimer tous les nœuds d'une version particulière de Kubernetes (dans ce cas, `1.28`) à l'aide de l'extrait de code suivant.

```
K8S_VERSION=1.28
for node in $(kubectl get nodes -o json | jq --arg K8S_VERSION
 "$K8S_VERSION" -r '.items[] | select(.status.nodeInfo.kubeletVersion |
 match("\($K8S_VERSION)")).metadata.name')
do
    echo "Deleting $node"
    kubectl delete node $node
done
```

9. Selon votre choix de CNI, il se peut que des artefacts subsistent sur vos nœuds hybrides après avoir exécuté les étapes ci-dessus. Pour plus d'informations, consultez the section called "Configurer CNI".

Mises à niveau sur place

Le processus de mise à niveau sur place consiste à utiliser `nodeadm upgrade` pour mettre à niveau la version Kubernetes pour les nœuds hybrides sans utiliser de nouveaux hôtes physiques ou virtuels ni de stratégie de migration par basculement. Le processus `nodeadm upgrade` arrête les anciens composants Kubernetes existants qui s'exécutent sur le nœud hybride, désinstalle les anciens composants Kubernetes existants, installe les nouveaux composants Kubernetes cibles et démarre les nouveaux composants Kubernetes cibles. Il est fortement recommandé de mettre à niveau un nœud à la fois afin de minimiser l'impact sur les applications exécutées sur les nœuds hybrides. La durée de ce processus dépend de la bande passante et de la latence de votre réseau.

1. Utilisez la commande suivante pour marquer le nœud que vous mettez à niveau comme non planifiable. Cela permet d'éviter que de nouveaux pods ne soient planifiés ou replanifiés sur le nœud que vous mettez à niveau. Pour plus d'informations, consultez kubectl cordon dans la documentation Kubernetes. Remplacer `NODE_NAME` par le nom du nœud hybride que vous mettez à niveau

```
kubectl cordon NODE_NAME
```

2. Videz le nœud que vous mettez à niveau à l'aide de la commande suivante. Pour plus d'informations sur la mise hors tension des nœuds, consultez la section Mettre hors tension un nœud en toute sécurité dans la documentation Kubernetes. Remplacez `NODE_NAME` par le nom du nœud hybride que vous mettez à niveau.

```
kubectl drain NODE_NAME --ignore-daemonsets --delete-emptydir-data
```

3. Exécutez `nodeadm upgrade` sur le nœud hybride que vous mettez à niveau. Vous devez exécuter `nodeadm` avec un utilisateur disposant des privilèges root/sudo. Le nom du nœud est conservé lors de la mise à niveau pour les fournisseurs d'informations d'identification AWS SSM et Rôles Anywhere IAM AWS. Vous ne pouvez pas changer de fournisseur d'informations d'identification pendant le processus de mise à niveau. Consultez the section called "Nœuds hybrides nodeadm" pour les valeurs de configuration de `nodeConfig.yaml`. Remplacez `K8S_VERSION` par la version cible de Kubernetes vers laquelle vous effectuez la mise à niveau.

```
nodeadm upgrade K8S_VERSION -c file://nodeConfig.yaml
```

4. Pour autoriser la planification des pods sur le nœud après la mise à niveau, tapez ce qui suit. Remplacez `NODE_NAME` par le nom du nœud.

```
kubectl uncordon NODE_NAME
```

5. Vérifiez le statut de vos nœuds hybrides et attendez que vos nœuds s'arrêtent et redémarrent sur la nouvelle version de Kubernetes avec le statut Ready.

```
kubectl get nodes -o wide -w
```

Mises à jour de sécurité pour les nœuds hybrides

Cette rubrique décrit la procédure à suivre pour appliquer des correctifs de sécurité à des packages et dépendances spécifiques s'exécutant sur vos nœuds hybrides. Nous vous recommandons de mettre régulièrement à jour vos nœuds hybrides afin de recevoir les CVE et les correctifs de sécurité.

Pour connaître les étapes à suivre pour mettre à niveau la version de Kubernetes, consultez the section called "Mettre à niveau les nœuds hybrides".

Un exemple de logiciel pouvant nécessiter un correctif de sécurité est `containerd`.

`Containerd`

`containerd` est l'exécution de conteneur Kubernetes standard et la dépendance principale pour les nœuds hybrides EKS, utilisée pour gérer le cycle de vie des conteneurs, y compris le téléchargement d'images et la gestion de l'exécution des conteneurs. Sur un nœud hybride, vous pouvez installer `containerd` via l'interface CLI nodeadm ou manuellement. En fonction du système d'exploitation de votre nœud, `nodeadm` installera `containerd` à partir du paquet distribué par le système d'exploitation ou du paquet Docker.

Lorsqu'un CVE dans `containerd` a été publié, vous disposez des options suivantes pour passer à la version corrigée de `containerd` sur vos nœuds hybrides.

Étape 1 : vérifier si le correctif a été publié dans les gestionnaires de paquets

Vous pouvez vérifier si le correctif CVE `containerd` a été publié pour chaque gestionnaire de paquets OS en consultant les bulletins de sécurité correspondants :

- Amazon Linux 2023
- RHEL
- Ubuntu 20.04
- Ubuntu 22.04
- Ubuntu 24.04

Si vous utilisez le dépôt Docker comme source de `containerd`, vous pouvez consulter les annonces de sécurité Docker pour vérifier la disponibilité de la version corrigée dans le dépôt Docker.

Étape 2 : choisir la méthode d'installation du correctif

Il existe trois méthodes pour corriger et installer les mises à niveau de sécurité sur place sur les nœuds. La méthode que vous pouvez utiliser dépend de la disponibilité du correctif dans le gestionnaire de paquets du système d'exploitation :

1. Installez les correctifs avec `nodeadm upgrade` qui sont publiés dans les gestionnaires de paquets, voir Étape 2a.
2. Installez les correctifs directement à l'aide des gestionnaires de paquets, voir Étape 2b.
3. Installez des correctifs personnalisés qui ne sont pas publiés dans les gestionnaires de paquets. Veuillez noter qu'il existe des considérations particulières pour les correctifs personnalisés pour `containerd`, Étape 2c.

Étape 2a : application d'un correctif avec **`nodeadm upgrade`**

Après avoir vérifié que le correctif CVE `containerd` a été publié dans les référentiels OS ou Docker (Apt ou RPM), vous pouvez utiliser la commande `nodeadm upgrade` pour passer à la dernière version de `containerd`. Comme il ne s'agit pas d'une mise à niveau de la version Kubernetes, vous devez indiquer votre version Kubernetes actuelle dans la commande de mise à niveau `nodeadm`.

```
nodeadm upgrade K8S_VERSION --config-source file:///root/nodeConfig.yaml
```

Étape 2b : application de correctifs à l'aide des gestionnaires de paquets du système d'exploitation

Vous pouvez également effectuer la mise à jour via le gestionnaire de paquets correspondant et l'utiliser pour mettre à niveau le paquet `containerd` comme suit.

Amazon Linux 2023

```
sudo yum update -y
sudo yum install -y containerd
```

RHEL

```
sudo yum install -y yum-utils
sudo yum-config-manager --add-repo https://download.docker.com/linux/rhel/docker-
ce.repo
sudo yum update -y
sudo yum install -y containerd
```

Ubuntu

```
sudo mkdir -p /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/
docker.asc
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc]
 https://download.docker.com/linux/ubuntu \
  $(. /etc/os-release && echo "${UBUNTU_CODENAME:-$VERSION_CODENAME}") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt update -y
sudo apt install -y --only-upgrade containerd.io
```

Étape 2c : correctif CVE **`Containerd`** non publié dans les gestionnaires de paquets

Si la version corrigée `containerd` n'est disponible que par d'autres moyens que le gestionnaire de paquets, par exemple dans les versions GitHub, vous pouvez installer `containerd` à partir du site officiel GitHub.

1. Si la machine a déjà rejoint le cluster en tant que nœud hybride, vous devez alors exécuter la commande `nodeadm uninstall`.
2. Installez les binaires officiels `containerd`. Vous pouvez suivre les étapes d'installation officielles sur GitHub.
3. Exécutez la commande `nodeadm install` avec l'argument `--containerd-source` défini sur `none`, ce qui ignorera l'installation `containerd` via `nodeadm`. Vous pouvez utiliser la valeur de `none` dans la source `containerd` pour tout système d'exploitation exécuté par le nœud.

```
nodeadm install K8S_VERSION --credential-provider CREDS_PROVIDER --containerd-source
 none
```

Supprimer les nœuds hybrides

Cette rubrique décrit comment supprimer les nœuds hybrides de votre cluster Amazon EKS. Vous devez supprimer vos nœuds hybrides avec l'outil compatible Kubernetes de votre choix, tel que kubectl. Les frais liés aux nœuds hybrides cessent lorsque l'objet nœud est supprimé du cluster Amazon EKS. Pour plus d'informations sur la tarification des nœuds hybrides, consultez la section Tarification d'Amazon EKS.

> **Important**
>
> La suppression de nœuds perturbe les charges de travail exécutées sur le nœud. Avant de supprimer des nœuds hybrides, nous vous recommandons de vider d'abord le nœud afin de déplacer les pods vers un autre nœud actif. Pour plus d'informations sur la mise hors tension des nœuds, consultez la section Mettre hors tension un nœud en toute sécurité dans la documentation Kubernetes.

Exécutez les étapes kubectl ci-dessous à partir de votre machine locale ou de l'instance que vous utilisez pour interagir avec le point de terminaison API Kubernetes du cluster Amazon EKS. Si vous utilisez un fichier spécifique `kubeconfig`, utilisez l'indicateur `--kubeconfig`.

Étape 1 : répertorier vos nœuds

```
kubectl get nodes
```

Étape 2 : vider votre nœud

Pour plus d'informations sur cette commande `kubectl drain`, consultez la section kubectl drain dans la documentation Kubernetes.

```
kubectl drain --ignore-daemonsets <node-name>
```

Étape 3 : arrêter et désinstaller les artefacts des nœuds hybrides

Vous pouvez utiliser la CLI des nœuds hybrides Amazon EKS (`nodeadm`) pour arrêter et supprimer les artefacts des nœuds hybrides de l'hôte. Vous devez exécuter `nodeadm` avec un utilisateur disposant des privilèges root/sudo. Par défaut, `nodeadm uninstall` ne continuera pas s'il reste des pods sur le nœud. Si vous utilisez Systems Manager (SSM) AWS comme fournisseur d'informations d'identification, la commande `nodeadm uninstall` désenregistre l'hôte en tant qu'instance AWS gérée par SSM. Pour de plus amples informations, consultez the section called "Nœuds hybrides nodeadm".

```
nodeadm uninstall
```

Étape 4 : supprimer votre nœud du cluster

Une fois les artefacts des nœuds hybrides arrêtés et désinstallés, supprimez la ressource du nœud de votre cluster.

```
kubectl delete node <node-name>
```

Étape 5 : vérifier s'il reste des artefacts

Selon votre choix de CNI, il se peut que des artefacts subsistent sur vos nœuds hybrides après avoir exécuté les étapes ci-dessus. Pour plus d'informations, consultez the section called "Configurer CNI".

Configuration de la mise en réseau des applications, des modules complémentaires et des webhooks pour les nœuds hybrides

Après avoir créé un cluster EKS pour les nœuds hybrides, configurez des fonctionnalités supplémentaires pour la mise en réseau des applications (CNI, BGP, Ingress, équilibrage de charge, politiques réseau), les modules complémentaires, les webhooks et les paramètres de proxy. Pour obtenir la liste complète des modules complémentaires EKS et communautaires compatibles avec les nœuds hybrides, consultez the section called "Configurer les modules complémentaires".

Informations sur les clusters EKS EKS inclut des vérifications permettant de détecter les erreurs de configuration dans vos nœuds hybrides qui pourraient nuire au bon fonctionnement de votre cluster ou de vos charges de travail. Pour plus d'informations sur les informations relatives aux clusters, consultez the section called "Informations sur le cluster".

La liste suivante répertorie les fonctionnalités courantes et les modules complémentaires que vous pouvez utiliser avec les nœuds hybrides :

- Interface réseau de conteneur (CNI) : AWS prend en charge Cilium en tant que CNI pour les nœuds hybrides. Pour de plus amples informations, consultez the section called "Configurer CNI". Notez que le CNI VPC AWS ne peut pas être utilisé avec des nœuds hybrides.
- CoreDNS et **`kube-proxy`** : CoreDNS et `kube-proxy` sont installés automatiquement lorsque des nœuds hybrides rejoignent le cluster EKS. Ces modules complémentaires peuvent être gérés en tant que modules complémentaires EKS après la création du cluster.
- Ingress et équilibrage de charge : vous pouvez utiliser l'AWS Load Balancer Controller et le Application Load Balancer (ALB) ou Network Load Balancer (NLB) avec le type de cible `ip` pour les charges de travail s'exécutant sur des nœuds hybrides. AWS prend en charge les fonctionnalités intégrées d'Ingress, de passerelle et d'équilibrage de charge Kubernetes Service de

Cilium pour les charges de travail s'exécutant sur des nœuds hybrides. Pour plus d'informations, consultez the section called "Configurer Ingress" et the section called "Configurer les services LoadBalancer".

- Métriques : vous pouvez utiliser les scrapers sans agent du service géré Amazon pour Prometheus (AMP), AWS Distro for Open Telemetry (ADOT) et l'agent d'observabilité Amazon CloudWatch avec des nœuds hybrides. Pour utiliser les scrapers AMP sans agent pour les métriques de pod sur les nœuds hybrides, vos pods doivent être accessibles depuis le VPC que vous utilisez pour le cluster EKS.
- Journaux : Vous pouvez activer la journalisation du plan de contrôle EKS pour les clusters compatibles avec les nœuds hybrides. Vous pouvez utiliser le module complémentaire EKS ADOT et le module complémentaire EKS de l'agent d'observabilité Amazon CloudWatch pour la journalisation hybride des nœuds et des pods.
- Identités de pod et IRSA : Vous pouvez utiliser les identités de pod EKS et les rôles IAM pour les comptes de service (IRSA) avec des applications s'exécutant sur des nœuds hybrides afin de permettre un accès granulaire à vos pods s'exécutant sur des nœuds hybrides avec d'autres services AWS.
- Webhooks : si vous utilisez des webhooks, consultez the section called "Configuration de webhooks" pour obtenir les considérations et les étapes à suivre pour exécuter éventuellement des webhooks sur des nœuds cloud si vous ne pouvez pas rendre vos réseaux de pods sur site routables.
- Proxy : si vous utilisez un serveur proxy dans votre environnement sur site pour le trafic sortant de votre centre de données ou de votre environnement périphérique, vous pouvez configurer vos nœuds hybrides et votre cluster pour utiliser votre serveur proxy. Pour de plus amples informations, consultez the section called "Configurer le proxy".

Rubriques

- Configurer CNI pour les nœuds hybrides
- Configurer les modules complémentaires pour les nœuds hybrides
- Configurer les webhooks pour les nœuds hybrides
- Configurer le proxy pour les nœuds hybrides
- Configurer Cilium BGP pour les nœuds hybrides
- Configurer Kubernetes Ingress pour les nœuds hybrides
- Configurer les services de type LoadBalancer pour les nœuds hybrides
- Configurer les politiques réseau Kubernetes pour les nœuds hybrides

Configurer CNI pour les nœuds hybrides

Cilium est l'interface réseau de conteneurs (CNI) prise en AWS charge par les nœuds hybrides Amazon EKS. Vous devez installer un CNI pour que les nœuds hybrides soient prêts à prendre en charge les charges de travail. Les nœuds hybrides apparaissent avec le statut `Not Ready` jusqu'à ce qu'un CNI soit en cours d'exécution. Vous pouvez gérer le CNI à l'aide des outils de votre choix, tels que Helm. Les instructions fournies sur cette page concernent la gestion du cycle de vie de Cilium (installation, mise à niveau, suppression). Consultez the section called "Présentation de Cilium Ingress et Cilium Gateway", the section called "Type de service LoadBalancer", et the section called "Configurer les politiques réseau" pour savoir comment configurer Cilium pour l'ingress, l'équilibrage de charge et les politiques réseau.

Cilium n'est pas pris en charge AWS lorsqu'il est exécuté sur des nœuds dans AWS le cloud. Le CNI VPC Amazon n'est pas compatible avec les nœuds hybrides et le CNI VPC est configuré avec une anti-affinité pour l'étiquette `eks.amazonaws.com/compute-type: hybrid`.

La documentation Calico qui figurait auparavant sur cette page a été déplacée vers le référentiel d'exemples hybrides EKS.

Compatibilité des versions

Les versions Cilium `v1.17.x` et B `v1.18.x` sont prises en charge pour les nœuds hybrides EKS pour chaque version de Kubernetes prise en charge dans Amazon EKS.

> **Note**
>
> Configuration requise pour le noyau Cilium v1.18.3 : En raison de l'exigence du noyau (noyau Linux >= 5.10), Cilium v1.18.3 n'est pas pris en charge sur :

- Ubuntu 20.04
- Red Hat Enterprise Linux (RHEL) 8

Pour la configuration système requise, consultez la section Configuration requise pour Cilium.

Consultez la prise en charge des versions Kubernetes pour connaître les versions Kubernetes prises en charge par Amazon EKS. Les nœuds hybrides EKS prennent en charge la même version de Kubernetes que les clusters Amazon EKS avec nœuds cloud.

Fonctionnalités prises en charge

AWS gère des versions de Cilium pour les nœuds hybrides EKS basées sur le projet open source Cilium. Pour bénéficier de l'assistance de AWS Cilium, vous devez utiliser les versions de Cilium AWS maintenues et les versions de Cilium prises en charge.

AWS fournit un support technique pour les configurations par défaut des fonctionnalités suivantes de Cilium à utiliser avec les nœuds hybrides EKS. Si vous envisagez d'utiliser des fonctionnalités hors du champ d' AWS assistance, il est recommandé d'obtenir un support commercial alternatif pour Cilium ou de disposer de l'expertise interne nécessaire pour résoudre les problèmes et apporter des correctifs au projet Cilium.

Fonction Cilium	Soutenu par AWS
Conformité au réseau Kubernetes	Oui
Connectivité au cluster principal	Oui
Famille d'IP	IPv4
La gestion du cycle de vie	Helm
Mode réseau	Encapsulation VXLAN
Gestion des adresses IP (IPAM)	Champ d'application du cluster Cilium IPAM
Stratégie de réseau	Stratégie réseau Kubernetes
Protocole de passerelle frontière (BGP)	Plan de contrôle Cilium BGP
Entrée Kubernetes	Cilium Ingress, Cilium Gateway
Allocation d' LoadBalancer adresses IP de service	Équilibreur de charge Cilium IPAM
Publicité d'adresse LoadBalancer IP du service	Plan de contrôle Cilium BGP
remplacement de kube-proxy	Oui

Considérations relatives à Cilium

- Référentiel Helm : AWS héberge le graphique Cilium Helm dans le Amazon Elastic Container Registry Public (Amazon ECR Public) sur Amazon EKS Cilium/Cilium. Les versions disponibles incluent :
  - Cilium v1.17.9 : `oci://public.ecr.aws/eks/cilium/cilium:1.17.9-0`
  - Cilium v1.18.3 : `oci://public.ecr.aws/eks/cilium/cilium:1.18.3-0`

    Les commandes de cette rubrique utilisent ce référentiel. Notez que certaines commandes `helm repo` ne sont pas valides pour les référentiels Helm dans Amazon ECR Public. Vous ne pouvez donc pas faire référence à ce référentiel à partir d'un nom de référentiel Helm local. Au lieu de cela, utilisez l'URI complet dans la plupart des commandes.
- Par défaut, Cilium est configuré pour fonctionner en mode overlay/tunnel avec VXLAN comme méthode d'encapsulation. Ce mode présente le moins d'exigences sur le réseau physique sous-jacent.
- Par défaut, Cilium masque l'adresse IP source de tout le trafic du pod quittant le cluster au profit de l'adresse IP du nœud. Si vous désactivez le masquage, votre pod CIDRs doit être routable sur votre réseau local.
- Si vous exécutez des webhooks sur des nœuds hybrides, votre pod CIDRs doit être routable sur votre réseau local. Si votre pod CIDRs n'est pas routable sur votre réseau local, il est recommandé d'exécuter des webhooks sur les nœuds cloud du même cluster. Pour plus d'informations, consultez the section called "Configuration de webhooks" et the section called "Préparez votre réseau".
- AWS recommande d'utiliser la fonctionnalité BGP intégrée de Cilium pour rendre votre pod CIDRs routable sur votre réseau local. Pour plus d'informations sur la configuration de Cilium BGP avec des nœuds hybrides, consultez the section called "Configuration de BGP".
- La gestion des adresses IP (IPAM) par défaut dans Cilium s'appelle Cluster Scope, où l'opérateur Cilium alloue des adresses IP à chaque nœud en fonction du pod configuré par l'utilisateur. CIDRs

Installer Cilium sur les nœuds hybrides

Procédure

1. Créez un fichier YAML nommé `cilium-values.yaml`. L'exemple suivant configure Cilium pour qu'il s'exécute uniquement sur des nœuds hybrides en définissant l'affinité pour l'étiquette `eks.amazonaws.com/compute-type: hybrid` de l'agent et de l'opérateur Cilium.

- Configurez `clusterPoolIpv4PodCIDRList` avec le même pod CIDRs que celui que vous avez configuré pour les réseaux de pods distants de votre cluster EKS. Par exemple, `10.100.0.0/24`. L'opérateur Cilium attribue des tranches d'adresses IP à partir de l'espace IP `clusterPoolIpv4PodCIDRList` configuré. Votre CIDR de pod ne doit pas chevaucher votre CIDR de nœud sur site, votre CIDR VPC ou votre CIDR de service Kubernetes.
- Configurez `clusterPoolIpv4MaskSize` en fonction du nombre de pods requis par nœud. Par exemple, 25 pour une taille de segment /25 de 128 pods par nœud.
- Ne modifiez pas `clusterPoolIpv4PodCIDRList` ou `clusterPoolIpv4MaskSize` après avoir déployé Cilium sur votre cluster, consultez la section Extension du pool de clusters pour plus d'informations.
- Si vous exécutez Cilium en mode de remplacement de kube-proxy, définissez `kubeProxyReplacement: "true"` dans vos valeurs Helm et assurez-vous qu'aucun déploiement kube-proxy n'est en cours d'exécution sur les mêmes nœuds que Cilium.
- L'exemple ci-dessous désactive le proxy Envoy Layer 7 (L7) utilisé par Cilium pour les politiques réseau L7 et l'entrée. Pour plus d'informations, consultez the section called "Configurer les politiques réseau" et the section called "Présentation de Cilium Ingress et Cilium Gateway".
- L'exemple ci-dessous configure `loadBalancer.serviceTopology` : `true` pour que la distribution du trafic de service fonctionne correctement si vous la configurez pour vos services. Pour de plus amples informations, veuillez consulter the section called "Configuration de la distribution du trafic de service".
- Pour obtenir la liste complète des valeurs Helm pour Cilium, consultez la référence Helm dans la documentation Cilium.

```
affinity:
  nodeAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
      - matchExpressions:
        - key: eks.amazonaws.com/compute-type
          operator: In
          values:
          - hybrid
ipam:
  mode: cluster-pool
  operator:
    clusterPoolIPv4MaskSize: 25
    clusterPoolIPv4PodCIDRList:
```

```
    - POD_CIDR
loadBalancer:
  serviceTopology: true
operator:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: eks.amazonaws.com/compute-type
            operator: In
            values:
              - hybrid
  unmanagedPodWatcher:
    restart: false
loadBalancer:
  serviceTopology: true
envoy:
  enabled: false
kubeProxyReplacement: "false"
```

2. Installez Cilium sur votre cluster.
   - `CILIUM_VERSION`Remplacez-le par une version Cilium (par exemple `1.17.9-0` ou`1.18.3-0`). Il est recommandé d'utiliser la dernière version du correctif pour la version mineure de Cilium.
   - Assurez-vous que vos nœuds répondent aux exigences du noyau pour la version que vous choisissez. Cilium v1.18.3 nécessite un noyau Linux >= 5.10.
   - Si vous utilisez un fichier kubeconfig spécifique, utilisez le drapeau `--kubeconfig` avec la commande d'installation Helm.

```
helm install cilium oci://public.ecr.aws/eks/cilium/cilium \
    --version CILIUM_VERSION \
    --namespace kube-system \
    --values cilium-values.yaml
```

3. Vérifiez que l'installation de Cilium s'est bien déroulée à l'aide des commandes suivantes. Vous devriez voir le déploiement `cilium-operator` et l'exécution `cilium-agent` sur chacun de vos nœuds hybrides. De plus, vos nœuds hybrides devraient désormais avoir un statut `Ready`. Pour plus d'informations sur la façon de configurer Cilium BGP pour promouvoir votre pod sur votre réseau local, passez CIDRs à. the section called "Configuration de BGP"

```
kubectl get pods -n kube-system
```

```
NAME                              READY   STATUS    RESTARTS   AGE
cilium-jjjn8                      1/1     Running   0          11m
cilium-operator-d4f4d7fcb-sc5xn   1/1     Running   0          11m
```

```
kubectl get nodes
```

```
NAME                   STATUS   ROLES    AGE   VERSION
mi-04a2cf999b7112233   Ready    <none>   19m   v1.31.0-eks-a737599
```

Mettre à niveau Cilium sur les nœuds hybrides

Avant de mettre à niveau votre déploiement Cilium, consultez attentivement la documentation relative à la mise à niveau de Cilium et les notes de mise à niveau afin de comprendre les modifications apportées à la version Cilium cible.

1. Assurez-vous d'avoir installé l'interface `helm` CLI dans votre environnement de ligne de commande. Consultez la documentation Helm pour obtenir les instructions d'installation.
2. Exécutez le contrôle de mise à niveau de Cilium avant le vol. Remplacez `CILIUM_VERSION` par votre version cible de Cilium. Nous vous recommandons d'utiliser la dernière version du correctif pour votre version mineure de Cilium. Vous trouverez la dernière version du correctif pour une version mineure donnée de Cilium dans la section Versions stables de la documentation Cilium.

```
helm install cilium-preflight oci://public.ecr.aws/eks/cilium/cilium --version
 CILIUM_VERSION \
  --namespace=kube-system \
  --set preflight.enabled=true \
  --set agent=false \
  --set operator.enabled=false
```

3. Après avoir appliqué le `cilium-preflight.yaml`, assurez-vous que le nombre de pods `READY` correspond au nombre de pods Cilium en cours d'exécution.

```
kubectl get ds -n kube-system | sed -n '1p;/cilium/p'
```

```
NAME                      DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE
 SELECTOR   AGE
cilium                    2         2         2       2            2           <none>
          1h20m
cilium-pre-flight-check   2         2         2       2            2           <none>
          7m15s
```

4. Une fois que le nombre de pods READY est égal, assurez-vous que le déploiement préalable de Cilium est également marqué comme READY 1/1. Si le message READY 0/1 s'affiche, consultez la section Validation CNP et résolvez les problèmes liés au déploiement avant de poursuivre la mise à niveau.

```
kubectl get deployment -n kube-system cilium-pre-flight-check -w
```

```
NAME                      READY   UP-TO-DATE   AVAILABLE   AGE
cilium-pre-flight-check   1/1     1            0           12s
```

5. Supprimer le pré-vol

```
helm uninstall cilium-preflight --namespace kube-system
```

6. Avant d'exécuter la commande `helm upgrade`, conservez les valeurs de votre déploiement dans un `existing-cilium-values.yaml` ou utilisez les options de ligne de commande `--set` pour vos paramètres lorsque vous exécutez la commande de mise à niveau. L'opération de mise à niveau remplace le Cilium ConfigMap. Il est donc essentiel que vos valeurs de configuration soient transmises lors de la mise à niveau.

```
helm get values cilium --namespace kube-system -o yaml > existing-cilium-values.yaml
```

7. Pendant le fonctionnement normal du cluster, tous les composants Cilium doivent exécuter la même version. Les étapes suivantes décrivent comment mettre à niveau tous les composants d'une version stable vers une version stable ultérieure. Lors de la mise à niveau d'une version mineure vers une autre version mineure, il est recommandé de commencer par mettre à niveau vers la dernière version corrective de la version mineure existante de Cilium. Pour minimiser les perturbations, définissez l'option `upgradeCompatibility` sur la version initiale de Cilium que vous avez installée dans ce cluster.

```
helm upgrade cilium oci://public.ecr.aws/eks/cilium/cilium --version CILIUM_VERSION \
```

```
  --namespace kube-system \
  --set upgradeCompatibility=1.X \
  -f existing-cilium-values.yaml
```

8. (Facultatif) Si vous devez annuler votre mise à niveau en raison de problèmes, exécutez les commandes suivantes.

```
helm history cilium --namespace kube-system
helm rollback cilium [REVISION] --namespace kube-system
```

Supprimer Cilium des nœuds hybrides

1. Exécutez la commande suivante pour désinstaller tous les composants Cilium de votre cluster. Remarque : la désinstallation du CNI peut avoir un impact sur l'état des nœuds et des pods et ne doit pas être effectuée sur les clusters de production.

```
helm uninstall cilium --namespace kube-system
```

   Les interfaces et les routes configurées par Cilium ne sont pas supprimées par défaut lorsque le CNI est supprimé du cluster. Consultez le GitHub problème pour plus d'informations.

2. Pour nettoyer les fichiers de configuration et les ressources sur disque, si vous utilisez les répertoires de configuration standard, vous pouvez supprimer les fichiers comme indiqué par le `cni-uninstall.sh`script dans le référentiel Cilium sur. GitHub
3. Pour supprimer les définitions de ressources personnalisées Cilium (CRDs) de votre cluster, vous pouvez exécuter les commandes suivantes.

```
kubectl get crds -oname | grep "cilium" | xargs kubectl delete
```

Configurer les modules complémentaires pour les nœuds hybrides

Cette page décrit les considérations relatives à l'exécution de AWS modules complémentaires et de modules complémentaires communautaires sur les nœuds hybrides Amazon EKS. Pour en savoir plus sur les modules complémentaires Amazon EKS et les processus de création, de mise à niveau et de suppression de modules complémentaires de votre cluster, consultez the section called "Modules complémentaires Amazon EKS". Sauf indication contraire sur cette page, les processus de création, de mise à niveau et de suppression des modules complémentaires Amazon EKS sont les

mêmes pour les clusters Amazon EKS dotés de nœuds hybrides que pour les clusters Amazon EKS dotés de nœuds exécutés dans AWS le cloud. Seuls les modules complémentaires inclus dans cette page ont été validés pour leur compatibilité avec les nœuds hybrides Amazon EKS.

Les AWS modules complémentaires suivants sont compatibles avec les nœuds hybrides Amazon EKS.

AWS module complémentaire	Versions complémentaires compatibles
kube-proxy	v1.25.14-eksbuild.2 et versions ultérieures
CoreDNS	v1.9.3-eksbuild.7 et versions ultérieures
AWS Distribution pour OpenTelemetry (ADOT)	v0.102.1-eksbuild.2 et versions ultérieures
CloudWatch Agent d'observabilité	v2.2.1-eksbuild.1 et versions ultérieures
Agent d'identité du pod EKS	• v1.3.3-eksbuild.1 et versions ultérieures, à l'exception de Bottlerocket • v1.3.7-eksbuild.2 et versions ultérieures pour Bottlerocket
Agent de surveillance des nœuds	v1.2.0-eksbuild.1 et versions ultérieures
Contrôleur d'instantané CSI	v8.1.0-eksbuild.1 et ultérieures
AWS Connecteur CA privé pour Kubernetes	v1.6.0-eksbuild.1 et ultérieures
pilote Amazon FSx CSI	v1.7.0-eksbuild.1 et versions ultérieures
AWS Fournisseur de pilotes CSI Secrets Store	v2.1.1-eksbuild.1 et versions ultérieures

Les modules complémentaires communautaires suivants sont compatibles avec les nœuds hybrides Amazon EKS. Pour en savoir plus sur les modules complémentaires communautaires, consultez the section called "Extensions communautaires".

Module complémentaire communautaire	Versions complémentaires compatibles
Serveur de métriques Kubernetes	v0.7.2-eksbuild.1 et versions ultérieures

Module complémentaire communautaire	Versions complémentaires compatibles
cert-manager	v1.17.2-eksbuild.1 et versions ultérieures
Exportateur de nœuds Prometheus	v1.9.1-eksbuild.2 et versions ultérieures
kube-state-metrics	v2.15.0-eksbuild.4 et versions ultérieures
DNS externe	v0.19.0-eksbuild.1 et versions ultérieures

Outre les modules complémentaires Amazon EKS présentés dans les tableaux ci-dessus, le collecteur du service géré Amazon pour Prometheus et le contrôleur d'équilibreur de charge AWS pour l'entrée d'applications (HTTP) et l'équilibrage de charge (TCP/UDP) sont compatibles avec les nœuds hybrides.

Il existe des AWS modules complémentaires et des modules complémentaires communautaires qui ne sont pas compatibles avec les nœuds hybrides Amazon EKS. Les dernières versions de ces modules complémentaires disposent d'une règle anti-affinité pour l'étiquette `eks.amazonaws.com/compute-type: hybrid` par défaut appliquée aux nœuds hybrides. Cela les empêche de s'exécuter sur des nœuds hybrides lorsqu'ils sont déployés dans vos clusters. Si vous avez des clusters comportant à la fois des nœuds hybrides et des nœuds exécutés dans AWS le cloud, vous pouvez déployer ces modules complémentaires dans votre cluster sur des nœuds exécutés dans AWS le cloud. L'Amazon VPC CNI n'est pas compatible avec les nœuds hybrides, et Cilium et Calico sont pris en charge en tant qu'interfaces réseau de conteneurs () CNIs pour les nœuds hybrides Amazon EKS. Pour plus d'informations, consultez the section called "Configurer CNI".

AWS modules complémentaires

Les sections suivantes décrivent les différences entre l'exécution de AWS modules complémentaires compatibles sur des nœuds hybrides par rapport aux autres types de calcul Amazon EKS.

kube-proxy et CoreDNS

EKS installe kube-proxy et CoreDNS en tant que modules complémentaires autogérés par défaut lorsque vous créez un cluster EKS avec l'API AWS et, notamment, à partir de la CLI. AWS SDKs AWS Vous pouvez remplacer ces modules complémentaires par les modules complémentaires Amazon EKS après la création du cluster. Consultez la documentation EKS pour plus de détails sur the section called "kube-proxy" et the section called "CoreDNS". Si vous utilisez un cluster en mode mixte avec à la fois des nœuds hybrides et des nœuds dans le AWS cloud, il est AWS recommandé

d'avoir au moins une réplique CoreDNS sur les nœuds hybrides et au moins une réplique CoreDNS sur vos nœuds dans le cloud. AWS Consultez the section called “Configuration des réplicas de CoreDNS” pour les étapes de configuration.

CloudWatch Agent d'observabilité

L'opérateur de l'agent CloudWatch Observability utilise des webhooks. Si vous exécutez l'opérateur sur des nœuds hybrides, votre CIDR de pod sur site doit être routable sur votre réseau sur site et vous devez configurer votre cluster EKS avec votre réseau de pods distant. Pour plus d'informations, consultez Configurer les webhooks pour les nœuds hybrides.

Les métriques au niveau des nœuds ne sont pas disponibles pour les nœuds hybrides car CloudWatch Container Insights dépend de la disponibilité du service IMDS (Instance Metadata Service) pour les métriques au niveau des nœuds. Les métriques au niveau des clusters, des charges de travail, des pods et des conteneurs sont disponibles pour les nœuds hybrides.

Après avoir installé le module complémentaire en suivant les étapes décrites dans Installer l' CloudWatch agent avec Amazon CloudWatch Observability, le manifeste du module complémentaire doit être mis à jour pour que l'agent puisse s'exécuter correctement sur des nœuds hybrides. Modifiez la ressource `amazoncloudwatchagents` sur le cluster pour ajouter la variable d'environnement `RUN_WITH_IRSA` comme indiqué ci-dessous.

```
kubectl edit amazoncloudwatchagents -n amazon-cloudwatch cloudwatch-agent
```

```
apiVersion: v1
items:
- apiVersion: cloudwatch.aws.amazon.com/v1alpha1
  kind: AmazonCloudWatchAgent
  metadata:
    ...
    name: cloudwatch-agent
    namespace: amazon-cloudwatch
    ...
  spec:
    ...
    env:
    - name: RUN_WITH_IRSA # <-- Add this
      value: "True" # <-- Add this
    - name: K8S_NODE_NAME
      valueFrom:
```

```
        fieldRef:
          fieldPath: spec.nodeName
          ...
```

Collecteur géré Amazon Managed Prometheus pour nœuds hybrides

Un collecteur géré par le service géré Amazon pour Prometheus (AMP) consiste en un scraper qui détecte et collecte des métriques à partir des ressources d'un cluster Amazon EKS. AMP gère le scraper pour vous, vous évitant ainsi d'avoir à gérer vous-même les instances, les agents ou les scrapers.

Vous pouvez utiliser les collecteurs gérés AMP sans aucune configuration supplémentaire spécifique aux nœuds hybrides. Cependant, les points de terminaison métriques de vos applications sur les nœuds hybrides doivent être accessibles depuis le VPC, y compris les itinéraires entre le VPC et le CIDRs réseau de pods distants et les ports ouverts dans votre pare-feu sur site. De plus, votre cluster doit disposer d'un accès privé au point de terminaison du cluster.

Suivez les étapes décrites dans la section Utilisation d'un collecteur AWS géré dans le guide de l'utilisateur d'Amazon Managed Service for Prometheus.

AWS Distribution pour OpenTelemetry (ADOT)

Vous pouvez utiliser le module complémentaire AWS Distro for OpenTelemetry (ADOT) pour collecter des métriques, des journaux et des données de suivi à partir de vos applications exécutées sur des nœuds hybrides. ADOT utilise des webhooks d'admission pour modifier et valider les demandes de ressources personnalisées du collecteur. Si vous exécutez l'opérateur ADOT sur des nœuds hybrides, votre CIDR de pod sur site doit être routable sur votre réseau sur site et vous devez configurer votre cluster EKS avec votre réseau de pods distant. Pour plus d'informations, consultez Configurer les webhooks pour les nœuds hybrides.

Suivez les étapes décrites dans Getting Started with AWS Distro pour OpenTelemetry utiliser les modules complémentaires EKS dans la AWS distribution pour OpenTelemetry obtenir de la documentation.

AWS Contrôleur Load Balancer

Vous pouvez utiliser le AWS Load Balancer Controller et l'Application Load Balancer (ALB) ou le Network Load Balancer (NLB) avec le `ip` type de cible pour les charges de travail sur les nœuds hybrides. Les cibles IP utilisées avec l'ALB ou le NLB doivent être routables depuis. AWSLe

contrôleur AWS Load Balancer utilise également des webhooks. Si vous exécutez l'opérateur AWS Load Balancer Controller sur des nœuds hybrides, le CIDR de votre pod local doit être routable sur votre réseau local et vous devez configurer votre cluster EKS avec votre réseau de pods distants. Pour plus d'informations, consultez Configurer les webhooks pour les nœuds hybrides.

Pour installer le AWS Load Balancer Controller, suivez les étapes indiquées dans ou. the section called "AWS Application Load Balancer" the section called "AWS Network Load Balancer"

Pour l'entrée avec ALB, vous devez spécifier les annotations ci-dessous. Pour plus d'informations, consultez the section called "Répartition de la charge des applications".

```
alb.ingress.kubernetes.io/target-type: ip
```

Pour l'équilibrage de charge avec NLB, vous devez spécifier les annotations ci-dessous. Pour plus d'informations, consultez the section called "Répartition de charge réseau".

```
service.beta.kubernetes.io/aws-load-balancer-type: "external"
service.beta.kubernetes.io/aws-load-balancer-nlb-target-type: "ip"
```

Agent d'identité du pod EKS

> **Note**
>
> Pour déployer correctement le module complémentaire EKS Pod Identity Agent sur des nœuds hybrides exécutant Bottlerocket, assurez-vous que votre version de Bottlerocket est au moins v1.39.0. Pod Identity Agent n'est pas pris en charge sur les versions antérieures de Bottlerocket dans les environnements à nœuds hybrides.

L'agent d'identité Amazon EKS Pod d'origine DaemonSet s'appuie sur la disponibilité d'EC2 IMDS sur le nœud pour obtenir les informations d'identification requises AWS . Comme l'IMDS n'est pas disponible sur les nœuds hybrides, à partir de la version 1.3.3-eksbuild.1, le module complémentaire Pod Identity Agent déploie éventuellement un module qui monte les informations d'identification requises. DaemonSet Les nœuds hybrides exécutant Bottlerocket nécessitent une méthode différente pour monter les informations d'identification, et à partir de la version 1.3.7-eksbuild.2, le module complémentaire Pod Identity Agent déploie éventuellement une méthode qui cible spécifiquement les nœuds hybrides Bottlerocket. DaemonSet Les sections suivantes décrivent le processus d'activation de l'option DaemonSets.

Ubuntu/RHEL/AL2023

1. Pour utiliser l'agent Pod Identity sur Ubuntu/RHEL/Al 2023 nœuds hybrides, `enableCredentialsFile: true` configurez-le dans la section hybride de la `nodeadm` configuration comme indiqué ci-dessous :

```
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
    hybrid:
        enableCredentialsFile: true # <-- Add this
```

   Cela permettra à `nodeadm` de créer un fichier d'informations d'identification à configurer sur le nœud sous `/eks-hybrid/.aws/credentials`, qui sera utilisé par les pods `eks-pod-identity-agent`. Ce fichier d'informations d'identification contiendra des AWS informations d'identification temporaires qui seront actualisées périodiquement.

2. Après avoir mis à jour la configuration `nodeadm` sur chaque nœud, exécutez la commande `nodeadm init` suivante avec votre `nodeConfig.yaml` pour joindre vos nœuds hybrides à votre cluster Amazon EKS. Si vos nœuds ont déjà rejoint le cluster, exécutez à nouveau la commande `nodeadm init`.

```
nodeadm init -c file://nodeConfig.yaml
```

3. Installation `eks-pod-identity-agent` avec prise en charge des nœuds hybrides activée, à l'aide de la AWS CLI ou AWS Management Console.

   a. AWS CLI : depuis la machine que vous utilisez pour administrer le cluster, exécutez la commande suivante pour effectuer l'installation en `eks-pod-identity-agent` activant la prise en charge des nœuds hybrides. Remplacez `my-cluster` par le nom de votre cluster.

```
aws eks create-addon \
    --cluster-name my-cluster \
    --addon-name eks-pod-identity-agent \
    --configuration-values '{"daemonsets":{"hybrid":{"create": true}}}'
```

   b. AWS Management Console: Si vous installez le module complémentaire Pod Identity Agent via la AWS console, ajoutez ce qui suit à la configuration facultative pour déployer DaemonSet celui qui cible les nœuds hybrides.

```
{"daemonsets":{"hybrid":{"create": true}}}
```

Flacon Rocket

1. Pour utiliser l'agent Pod Identity sur les nœuds hybrides Bottlerocket, ajoutez l'indicateur `--enable-credentials-file=true` à la commande utilisée pour les données utilisateur du conteneur de démarrage Bottlerocket, comme décrit dans the section called "Connectez des nœuds hybrides avec Bottlerocket".

   a. Si vous utilisez le fournisseur d'informations d'identification SSM, votre commande devrait ressembler à ceci :

   ```
   eks-hybrid-ssm-setup --activation-id=<activation-id> --activation-
   code=<activation-code> --region=<region> --enable-credentials-file=true
   ```

   b. Si vous utilisez le fournisseur d'informations d'identification Rôles Anywhere IAM, votre commande devrait ressembler à ceci :

   ```
   eks-hybrid-iam-ra-setup --certificate=<certificate> --key=<private-key> --enable-
   credentials-file=true
   ```

   Cela configurera le script bootstrap pour créer un fichier d'informations d'identification sur le nœud sous `/var/eks-hybrid/.aws/credentials`, qui sera utilisé par les pods `eks-pod-identity-agent`. Ce fichier d'informations d'identification contiendra des AWS informations d'identification temporaires qui seront actualisées périodiquement.

2. Installation `eks-pod-identity-agent` avec prise en charge des nœuds hybrides Bottlerocket activée, à l'aide de la CLI AWS ou. AWS Management Console

   a. AWS CLI : depuis la machine que vous utilisez pour administrer le cluster, exécutez la commande suivante pour effectuer l'installation en activant la prise en charge `eks-pod-identity-agent` des nœuds hybrides Bottlerocket. Remplacez `my-cluster` par le nom de votre cluster.

   ```
   aws eks create-addon \
       --cluster-name my-cluster \
       --addon-name eks-pod-identity-agent \
       --configuration-values '{"daemonsets":{"hybrid-bottlerocket":{"create":
    true}}}'
   ```

b. AWS Management Console: Si vous installez le module complémentaire Pod Identity Agent via la AWS console, ajoutez ce qui suit à la configuration facultative pour déployer le module DaemonSet qui cible les nœuds hybrides Bottlerocket.

```
{"daemonsets":{"hybrid-bottlerocket":{"create": true}}}
```

Contrôleur d'instantané CSI

À partir de la version`v8.1.0-eksbuild.2`, le module complémentaire CSI Snapshot Controller applique une règle d'anti-affinité souple pour les nœuds hybrides, préférant que le contrôleur `deployment` s'exécute sur EC2 dans la même région AWS que le plan de contrôle Amazon EKS. La colocation `deployment` dans la même AWS région que le plan de contrôle Amazon EKS améliore la latence.

Modules complémentaires communautaires

Les sections suivantes décrivent les différences entre l'exécution d'extensions communautaires compatibles sur des nœuds hybrides et d'autres types de calcul Amazon EKS.

Serveur de métriques Kubernetes

Le plan de contrôle doit atteindre l'adresse IP du pod du serveur de métriques (ou l'adresse IP du nœud si hostNetwork est activé). Par conséquent, à moins que vous n'exécutiez Metrics Server en mode hostNetwork, vous devez configurer un réseau de pods distant lors de la création de votre cluster Amazon EKS, et vous devez rendre vos adresses IP de pods routables. La mise en œuvre du protocole de passerelle frontière (BGP) avec le CNI est une méthode courante pour rendre les adresses IP de vos pods routables.

cert-manager

`cert-manager` utilise des webhooks. Si vous utilisez `cert-manager` sur des nœuds hybrides, votre CIDR de pod sur site doit être routable sur votre réseau sur site et vous devez configurer votre cluster EKS avec votre réseau de pods distant. Pour plus d'informations, consultez Configurer les webhooks pour les nœuds hybrides.

Configurer les webhooks pour les nœuds hybrides

Cette page détaille les considérations relatives à l'exécution de webhooks avec des nœuds hybrides. Les webhooks sont utilisés dans les applications Kubernetes et les projets open source, tels que

l'AWS Load Balancer Controller et l'agent d'observabilité CloudWatch, pour exécuter des fonctions de mutation et de validation lors de l'exécution.

Réseaux de pods routables

Si vous pouvez rendre votre pod CIDR sur site routable sur votre réseau sur site, vous pouvez exécuter des webhooks sur des nœuds hybrides. Il existe plusieurs techniques que vous pouvez utiliser pour rendre votre pod CIDR sur site routable sur votre réseau sur site, notamment le protocole de passerelle frontière (BGP), les routes statiques ou d'autres solutions de routage personnalisées. BGP est la solution recommandée, car elle est plus évolutive et plus facile à gérer que les autres solutions qui nécessitent une configuration personnalisée ou manuelle des routes. AWS prend en charge les capacités BGP de Cilium et Calico pour la publicité des CIDR de pod. Pour plus d'informations, consultez the section called "Configurer CNI" et the section called "CIDR de pods distants routables".

Réseaux de pods non routables

Si vous ne pouvez pas rendre votre pod CIDR sur site routable sur votre réseau sur site et que vous devez exécuter des webhooks, nous vous recommandons d'exécuter tous les webhooks sur des nœuds cloud dans le même cluster EKS que vos nœuds hybrides.

Considérations relatives aux clusters en mode mixte

Les clusters en mode mixte sont définis comme des clusters EKS qui comportent à la fois des nœuds hybrides et des nœuds fonctionnant dans le cloud AWS. Lorsque vous exécutez un cluster en mode mixte, tenez compte des recommandations suivantes :

- Exécutez le VPC CNI sur des nœuds dans le cloud AWS et Cilium ou Calico sur des nœuds hybrides. Cilium et Calico ne sont pas pris en charge par AWS lorsqu'ils sont exécutés sur des nœuds dans le Cloud AWS.
- Configurez les webhooks pour qu'ils s'exécutent sur des nœuds dans le cloud AWS. Consultez the section called "Configuration des webhooks pour les modules complémentaires" pour découvrir comment configurer les webhooks avec AWS et les modules complémentaires communautaires.
- Si vos applications nécessitent que les pods exécutés sur des nœuds dans le cloud AWS communiquent directement avec les pods exécutés sur des nœuds hybrides (« communication est-ouest ») et que vous utilisez le VPC CNI sur les nœuds dans le cloud AWS, et Cilium ou Calico sur les nœuds hybrides, alors votre CIDR de pod sur site doit être routable sur votre réseau sur site.
- Exécutez au moins un réplica de CoreDNS sur les nœuds dans le cloud AWS et au moins un réplica de CoreDNS sur les nœuds hybrides.

- Configurez la distribution du trafic de service afin que celui-ci reste localisé dans la zone d'où il provient. Pour plus d'informations sur la distribution du trafic de service, voir the section called "Configuration de la distribution du trafic de service".
- Si vous utilisez des Application Load Balancers (ALB) ou des Network Load Balancers (NLB) AWS pour le trafic de charge de travail s'exécutant sur des nœuds hybrides, les cibles IP utilisées avec l'ALB ou le NLB doivent être routables à partir d'AWS.
- Le module complémentaire Metrics Server nécessite une connectivité entre le plan de contrôle EKS et l'adresse IP du pod Metrics Server. Si vous exécutez le module complémentaire Metrics Server sur des nœuds hybrides, votre CIDR de pod sur site doit être routable sur votre réseau sur site.
- Pour collecter des métriques pour les nœuds hybrides à l'aide des collecteurs gérés par le service géré Amazon pour Prometheus (AMP), votre CIDR de pod sur site doit être routable sur votre réseau sur site. Vous pouvez également utiliser le collecteur géré AMP pour les métriques et les ressources du plan de contrôle EKS s'exécutant dans le cloud AWS, ainsi que le module complémentaire AWS Distro for OpenTelemetry (ADOT) pour collecter les métriques des nœuds hybrides.

Configuration de clusters en mode mixte

Pour afficher les webhooks de mutation et de validation exécutés sur votre cluster, vous pouvez consulter le type de ressource Extensions dans le panneau Ressources de la console EKS de votre cluster, ou utiliser les commandes suivantes. EKS rapporte également les métriques webhook dans le tableau de bord d'observabilité du cluster. Pour plus d'informations, consultez the section called "Tableau de bord d'observabilité".

```
kubectl get mutatingwebhookconfigurations
```

```
kubectl get validatingwebhookconfigurations
```

Configuration de la distribution du trafic de service

Lorsque vous utilisez des clusters en mode mixte, nous vous recommandons d'utiliser la distribution du trafic de service afin de maintenir le trafic de service local dans la zone d'où il provient. La distribution du trafic des services (disponible pour les versions 1.31 et ultérieures de Kubernetes dans EKS) est la solution recommandée par rapport au routage topologique, car elle est plus prévisible. Avec la distribution du trafic de service, les terminaux sains de la zone recevront tout le trafic destiné à cette zone. Avec le routage topologique, chaque service doit remplir plusieurs conditions dans cette

zone pour appliquer le routage personnalisé, sinon il achemine le trafic de manière uniforme vers tous les points de terminaison.

Si vous utilisez Cilium comme CNI, vous devez exécuter le CNI avec le paramètre `enable-service-topology` défini sur `true` pour activer la distribution du trafic de service.
Vous pouvez transmettre cette configuration avec l'indicateur d'installation Helm `--set loadBalancer.serviceTopology=true` ou mettre à jour une installation existante à l'aide de la commande CLI Cilium `cilium config set enable-service-topology true`. L'agent Cilium exécuté sur chaque nœud doit être redémarré après la mise à jour de la configuration d'une installation existante.

La section suivante présente un exemple de configuration de la distribution du trafic de service pour le service CoreDNS. Nous vous recommandons d'activer cette option pour tous les services de votre cluster afin d'éviter tout trafic inter-environnements indésirable.

Configuration des réplicas de CoreDNS

Si vous utilisez un cluster en mode mixte avec à la fois des nœuds hybrides et des nœuds dans le cloud AWS, nous vous recommandons d'avoir au moins un réplica CoreDNS sur les nœuds hybrides et au moins un réplica CoreDNS sur vos nœuds dans le cloud AWS. Pour éviter les problèmes de latence et de réseau dans une configuration de cluster en mode mixte, vous pouvez configurer le service CoreDNS pour qu'il privilégie le réplica CoreDNS le plus proche avec la distribution du trafic de service.

La distribution du trafic de service (disponible pour les versions 1.31 et ultérieures de Kubernetes dans EKS) est la solution recommandée par rapport au routage topologique, car elle est plus prévisible. Dans la distribution du trafic en service, les points de terminaison sains de la zone recevront tout le trafic destiné à cette zone. Dans le routage sensible à la topologie, chaque service doit remplir plusieurs conditions dans cette zone pour appliquer le routage personnalisé, sinon il achemine le trafic de manière uniforme vers tous les points de terminaison. Les étapes suivantes permettent de configurer la distribution du trafic de service.

Si vous utilisez Cilium comme CNI, vous devez exécuter le CNI avec le paramètre `enable-service-topology` défini sur `true` pour activer la distribution du trafic de service.
Vous pouvez transmettre cette configuration avec l'indicateur d'installation Helm `--set loadBalancer.serviceTopology=true` ou mettre à jour une installation existante à l'aide de la commande CLI Cilium `cilium config set enable-service-topology true`. L'agent Cilium exécuté sur chaque nœud doit être redémarré après la mise à jour de la configuration d'une installation existante.

1. Ajoutez une étiquette de zone topologique pour chacun de vos nœuds hybrides, par exemple `topology.kubernetes.io/zone: onprem`. Vous pouvez également définir l'étiquette au niveau de la phase `nodeadm init` en spécifiant l'étiquette dans votre configuration `nodeadm`, voir the section called "Configuration du nœud pour personnaliser kubelet (facultatif)". Remarque : Les nœuds exécutés dans le cloud AWS se voient automatiquement attribuer une étiquette de zone topologique correspondant à la zone de disponibilité (AZ) du nœud.

```
kubectl label node hybrid-node-name topology.kubernetes.io/zone=zone
```

2. Ajoutez `podAntiAffinity` au déploiement CoreDNS avec la clé de zone de topologie. Vous pouvez également configurer le déploiement de CoreDNS lors de l'installation à l'aide des modules complémentaires EKS.

```
kubectl edit deployment coredns -n kube-system
```

```
spec:
  template:
    spec:
      affinity:
       ...
        podAntiAffinity:
          preferredDuringSchedulingIgnoredDuringExecution:
          - podAffinityTerm:
              labelSelector:
                matchExpressions:
                - key: k8s-app
                  operator: In
                  values:
                  - kube-dns
              topologyKey: kubernetes.io/hostname
            weight: 100
          - podAffinityTerm:
              labelSelector:
                matchExpressions:
                - key: k8s-app
                  operator: In
                  values:
                  - kube-dns
              topologyKey: topology.kubernetes.io/zone
            weight: 50
```

```
      ...
```

3. Ajoutez le paramètre `trafficDistribution: PreferClose` à la configuration du service `kube-dns` pour activer la distribution du trafic du service.

```
kubectl patch svc kube-dns -n kube-system --type=merge -p '{
  "spec": {
    "trafficDistribution": "PreferClose"
  }
}'
```

4. Vous pouvez confirmer que la distribution du trafic de service est activée en consultant les tranches de point de terminaison du service `kube-dns`. Vos tranches de point de terminaison doivent afficher les étiquettes `hints` de votre zone topologique, ce qui confirme que la distribution du trafic de service est activée. Si vous ne voyez pas l'adresse de chaque point de terminaison `hints`, cela signifie que la distribution du trafic de service n'est pas activée.

```
kubectl get endpointslice -A | grep "kube-dns"
```

```
kubectl get endpointslice [.replaceable]`kube-dns-<id>`  -n kube-system -o yaml
```

```
addressType: IPv4
apiVersion: discovery.k8s.io/v1
endpoints:
- addresses:
  - <your-hybrid-node-pod-ip>
  hints:
    forZones:
    - name: onprem
  nodeName: <your-hybrid-node-name>
  zone: onprem
- addresses:
  - <your-cloud-node-pod-ip>
  hints:
    forZones:
    - name: us-west-2a
  nodeName: <your-cloud-node-name>
  zone: us-west-2a
```

Configuration des webhooks pour les modules complémentaires

Les modules complémentaires suivants utilisent des webhooks et sont compatibles avec les nœuds hybrides.

- Contrôleur de l'équilibreur de charge AWS
- Agent d'observabilité CloudWatch
- AWS Distro for OpenTelemetry (ADOT)
- `cert-manager`

Consultez les sections suivantes pour configurer les webhooks utilisés par ces modules complémentaires afin qu'ils s'exécutent sur les nœuds dans Cloud AWS.

Contrôleur de l'équilibreur de charge AWS

Pour utiliser l'AWS Load Balancer Controller dans une configuration de cluster en mode mixte, vous devez exécuter le contrôleur sur les nœuds dans le Cloud AWS. Pour ce faire, ajoutez les éléments suivants à votre configuration des valeurs Helm ou spécifiez les valeurs à l'aide de la configuration du module complémentaire EKS.

```
affinity:
  nodeAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
      - matchExpressions:
        - key: eks.amazonaws.com/compute-type
          operator: NotIn
          values:
          - hybrid
```

Agent d'observabilité CloudWatch

Le module complémentaire Agent d'observabilité CloudWatch dispose d'un opérateur Kubernetes qui utilise des webhooks. Pour exécuter l'opérateur sur des nœuds du AWS cloud dans une configuration de cluster en mode mixte, modifiez la configuration de l'opérateur Agent d'observabilité CloudWatch. Vous ne pouvez pas configurer l'affinité des opérateurs lors de l'installation avec les modules complémentaires Helm et EKS (voir le problème #2431 de containers-roadmap).

```
kubectl edit -n amazon-cloudwatch deployment amazon-cloudwatch-observability-
controller-manager
```

```
spec:
  ...
  template:
    ...
    spec:
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: eks.amazonaws.com/compute-type
                operator: NotIn
                values:
                - hybrid
```

AWS Distro for OpenTelemetry (ADOT)

Le module complémentaire AWS Distro pour OpenTelemetry (ADOT) dispose d'un opérateur Kubernetes qui utilise des webhooks. Pour exécuter l'opérateur sur les nœuds dans le cloud AWS dans une configuration de cluster en mode mixte, ajoutez les éléments suivants à votre configuration de valeurs Helm ou spécifiez les valeurs à l'aide de la configuration du module complémentaire EKS.

```
affinity:
  nodeAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
      - matchExpressions:
        - key: eks.amazonaws.com/compute-type
          operator: NotIn
          values:
          - hybrid
```

Si votre CIDR de pod n'est pas routable sur votre réseau sur site, le collecteur ADOT doit alors s'exécuter sur des nœuds hybrides afin de récupérer les métriques de vos nœuds hybrides et des charges de travail qui s'y exécutent. Pour ce faire, modifiez la définition de ressource personnalisée (CRD).

```
kubectl -n opentelemetry-operator-system edit opentelemetrycollectors.opentelemetry.io
 adot-col-prom-metrics
```

```
spec:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: eks.amazonaws.com/compute-type
            operator: In
            values:
            - hybrid
```

Vous pouvez configurer le collecteur ADOT pour qu'il ne récupère que les métriques des nœuds hybrides et des ressources s'exécutant sur des nœuds hybrides en ajoutant les éléments `relabel_configs` suivants à chaque `scrape_configs` dans la configuration CRD du collecteur ADOT.

```
relabel_configs:
  - action: keep
    regex: hybrid
    source_labels:
    - __meta_kubernetes_node_label_eks_amazonaws_com_compute_type
```

Le module complémentaire ADOT doit installer `cert-manager` pour pouvoir utiliser les certificats TLS utilisés par le webhook de l'opérateur ADOT. `cert-manager` exécute également des webhooks et vous pouvez le configurer pour qu'il s'exécute sur des nœuds dans le cloud AWS à l'aide de la configuration Helm suivante.

```
affinity:
  nodeAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
      - matchExpressions:
        - key: eks.amazonaws.com/compute-type
          operator: NotIn
          values:
          - hybrid
webhook:
  affinity:
```

```
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: eks.amazonaws.com/compute-type
            operator: NotIn
            values:
            - hybrid
cainjector:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: eks.amazonaws.com/compute-type
            operator: NotIn
            values:
            - hybrid
startupapicheck:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: eks.amazonaws.com/compute-type
            operator: NotIn
            values:
            - hybrid
```

`cert-manager`

Le module complémentaire exécute des webhooks `cert-manager` et vous pouvez le configurer pour qu'il s'exécute sur des nœuds dans le cloud AWS à l'aide de la configuration Helm suivante.

```
affinity:
  nodeAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
      - matchExpressions:
        - key: eks.amazonaws.com/compute-type
          operator: NotIn
          values:
          - hybrid
```

```
webhook:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: eks.amazonaws.com/compute-type
            operator: NotIn
            values:
            - hybrid
cainjector:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: eks.amazonaws.com/compute-type
            operator: NotIn
            values:
            - hybrid
startupapicheck:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: eks.amazonaws.com/compute-type
            operator: NotIn
            values:
            - hybrid
```

Configurer le proxy pour les nœuds hybrides

Si vous utilisez un serveur proxy dans votre environnement sur site pour le trafic sortant de votre centre de données ou de votre environnement périphérique, vous devez configurer séparément vos nœuds et votre cluster pour utiliser votre serveur proxy.

Cluster

Sur votre cluster, vous devez configurer `kube-proxy` pour utiliser votre serveur proxy. Vous devez effectuer la configuration de `kube-proxy` après avoir créé votre cluster Amazon EKS.

Nœuds

Sur vos nœuds, vous devez configurer le système d'exploitation, `containerd`, `kubelet`, et l'agent Amazon SSM pour utiliser votre serveur proxy. Vous pouvez effectuer ces modifications pendant le processus de création des images de votre système d'exploitation ou avant d'exécuter `nodeadm init` sur chaque nœud hybride.

Configuration au niveau du nœud

Vous devez appliquer les configurations suivantes soit dans vos images de système d'exploitation, soit avant l'exécution de `nodeadm init` sur chaque nœud hybride.

Configuration du proxy **`containerd`**

`containerd` est l'environnement d'exécution de gestion de conteneurs par défaut pour Kubernetes. Si vous utilisez un proxy pour accéder à Internet, vous devez configurer `containerd` afin qu'il puisse extraire les images de conteneur requises par Kubernetes et Amazon EKS.

Créez un fichier sur chaque nœud hybride appelé `http-proxy.conf` dans le répertoire `/etc/systemd/system/containerd.service.d` avec le contenu suivant. Remplacez `proxy-domain` et `port` par les valeurs correspondant à votre environnement.

```
[Service]
Environment="HTTP_PROXY=http://proxy-domain:port"
Environment="HTTPS_PROXY=http://proxy-domain:port"
Environment="NO_PROXY=localhost"
```

`containerd` configuration à partir des données utilisateur

Le répertoire `containerd.service.d` devra être créé pour ce fichier. Vous devrez recharger systemd pour récupérer le fichier de configuration sans redémarrer. Dans AL2023, le service sera probablement déjà en cours d'exécution lorsque votre script s'exécutera, vous devrez donc également le redémarrer.

```
mkdir -p /etc/systemd/system/containerd.service.d
echo '[Service]' > /etc/systemd/system/containerd.service.d/http-proxy.conf
echo 'Environment="HTTP_PROXY=http://proxy-domain:port"' >> /etc/systemd/system/
containerd.service.d/http-proxy.conf
echo 'Environment="HTTPS_PROXY=http://proxy-domain:port"' >> /etc/systemd/system/
containerd.service.d/http-proxy.conf
```

```
echo 'Environment="NO_PROXY=localhost"' >> /etc/systemd/system/containerd.service.d/
http-proxy.conf
systemctl daemon-reload
systemctl restart containerd
```

Configuration du proxy **`kubelet`**

`kubelet` est l'agent de nœud Kubernetes qui s'exécute sur chaque nœud Kubernetes et qui est chargé de gérer le nœud et les pods qui s'y exécutent. Si vous utilisez un proxy dans votre environnement sur site, vous devez configurer le `kubelet` afin qu'il puisse communiquer avec les points de terminaison publics ou privés de votre cluster Amazon EKS.

Créez un fichier sur chaque nœud hybride appelé `http-proxy.conf` dans le répertoire `/etc/systemd/system/kubelet.service.d/` avec le contenu suivant. Remplacez `proxy-domain` et `port` par les valeurs correspondant à votre environnement.

```
[Service]
Environment="HTTP_PROXY=http://proxy-domain:port"
Environment="HTTPS_PROXY=http://proxy-domain:port"
Environment="NO_PROXY=localhost"
```

Configuration **`kubelet`** à partir des données utilisateur

Le répertoire `kubelet.service.d` doit être créé pour ce fichier. Vous devrez recharger systemd pour récupérer le fichier de configuration sans redémarrer. Dans AL2023, le service sera probablement déjà en cours d'exécution lorsque votre script s'exécutera, vous devrez donc également le redémarrer.

```
mkdir -p /etc/systemd/system/kubelet.service.d
echo '[Service]' > /etc/systemd/system/kubelet.service.d/http-proxy.conf
echo 'Environment="HTTP_PROXY=http://proxy-domain:port"' >> /etc/systemd/system/
kubelet.service.d/http-proxy.conf
echo 'Environment="HTTPS_PROXY=http://proxy-domain:port"' >> /etc/systemd/system/
kubelet.service.d/http-proxy.conf
echo 'Environment="NO_PROXY=localhost"' >> /etc/systemd/system/kubelet.service.d/http-
proxy.conf
systemctl daemon-reload
systemctl restart kubelet
```

Configuration du proxy **`ssm`**

`ssm` est l'un des fournisseurs d'informations d'identification pouvant être utilisés pour initialiser un nœud hybride. `ssm` est chargé de l'authentification avec AWS et de la génération d'informations d'identification temporaires utilisées par `kubelet`. Si vous utilisez un proxy dans votre environnement sur site et que vous utilisez `ssm` comme fournisseur d'informations d'identification sur le nœud, vous devez le configurer le `ssm` afin qu'il puisse communiquer avec les points de terminaison du service Amazon SSM.

Créez un fichier sur chaque nœud hybride appelé `http-proxy.conf` dans le chemin ci-dessous en fonction du système d'exploitation

- Ubuntu – `/etc/systemd/system/snap.amazon-ssm-agent.amazon-ssm-agent.service.d/http-proxy.conf`
- Amazon Linux 2023 et Red Hat Enterprise Linux – `/etc/systemd/system/amazon-ssm-agent.service.d/http-proxy.conf`

Remplissez le fichier avec le contenu suivant. Remplacez `proxy-domain` et `port` par les valeurs correspondant à votre environnement.

```
[Service]
Environment="HTTP_PROXY=http://proxy-domain:port"
Environment="HTTPS_PROXY=http://proxy-domain:port"
Environment="NO_PROXY=localhost"
```

Configuration à partir des données utilisateur **`ssm`**

Le répertoire des fichiers de service systemd `ssm` doit être créé pour ce fichier. Le chemin d'accès au répertoire dépend du système d'exploitation utilisé sur le nœud.

- Ubuntu – `/etc/systemd/system/snap.amazon-ssm-agent.amazon-ssm-agent.service.d`
- Amazon Linux 2023 et Red Hat Enterprise Linux – `/etc/systemd/system/amazon-ssm-agent.service.d`

Remplacer le nom du service systemd dans la commande de redémarrage ci-dessous en fonction du système d'exploitation utilisé sur le nœud

- Ubuntu – `snap.amazon-ssm-agent.amazon-ssm-agent`
- Amazon Linux 2023 et Red Hat Enterprise Linux – `amazon-ssm-agent`

```
mkdir -p systemd-service-file-directory
echo '[Service]' > [.replaceable]#systemd-service-file-directory/http-proxy.conf
echo 'Environment="HTTP_PROXY=http://[.replaceable]#proxy-domain:port"' >> systemd-
service-file-directory/http-proxy.conf
echo 'Environment="HTTPS_PROXY=http://[.replaceable]#proxy-domain:port"' >>
 [.replaceable]#systemd-service-file-directory/http-proxy.conf
echo 'Environment="NO_PROXY=localhost"' >> [.replaceable]#systemd-service-file-
directory/http-proxy.conf
systemctl daemon-reload
systemctl restart [.replaceable]#systemd-service-name
```

Configuration du proxy du système d'exploitation

Si vous utilisez un proxy pour accéder à Internet, vous devez configurer votre système d'exploitation afin de pouvoir extraire les dépendances des nœuds hybrides à partir du gestionnaire de paquets de votre système d'exploitation.

Ubuntu

1. Configurez `snap` pour utiliser votre proxy à l'aide des commandes suivantes :

   ```
   sudo snap set system proxy.https=http://proxy-domain:port
   sudo snap set system proxy.http=http://proxy-domain:port
   ```

2. Pour activer le proxy pour `apt`, créez un fichier nommé `apt.conf` dans le répertoire `/etc/apt/`. Remplacez proxy-domain et port par les valeurs correspondant à votre environnement.

   ```
   Acquire::http::Proxy "http://proxy-domain:port";
   Acquire::https::Proxy "http://proxy-domain:port";
   ```

Amazon Linux 2023

1. Configurez `dnf` pour utiliser votre proxy. Créez un fichier `/etc/dnf/dnf.conf` contenant le domaine proxy et les valeurs de port de votre environnement.

```
proxy=http://proxy-domain:port
```

Utilisation de Red Hat Enterprise Linux

1. Configurez `yum` pour utiliser votre proxy. Créez un fichier `/etc/yum.conf` contenant le domaine proxy et les valeurs de port de votre environnement.

```
proxy=http://proxy-domain:port
```

Configuration du proxy de Rôles Anywhere IAM

Le service de fournisseur d'informations d'identification Rôles Anywhere IAM est chargé d'actualiser les informations d'identification lors de l'utilisation de Rôles Anywhere IAM avec l'indicateur `enableCredentialsFile` (voir the section called "Agent d'identité du pod EKS"). Si vous utilisez un proxy dans votre environnement sur site, vous devez configurer le service afin qu'il puisse communiquer avec les points de terminaison Rôles Anywhere IAM.

Créez un fichier nommé `http-proxy.conf` dans le répertoire `/etc/systemd/system/aws_signing_helper_update.service.d/` avec le contenu suivant. Remplacez `proxy-domain` et `port` par les valeurs correspondant à votre environnement.

```
[Service]
Environment="HTTP_PROXY=http://proxy-domain:port"
Environment="HTTPS_PROXY=http://proxy-domain:port"
Environment="NO_PROXY=localhost"
```

Configuration à l'échelle du cluster

Les configurations de cette section doivent être appliquées après avoir créé votre cluster Amazon EKS et avant d'exécuter `nodeadm init` sur chaque nœud hybride.

Configuration du proxy kube-proxy

Amazon EKS installe automatiquement `kube-proxy` sur chaque nœud hybride en tant que DaemonSet lorsque vos nœuds hybrides rejoignent le cluster. `kube-proxy` permet le routage

entre les services pris en charge par les pods sur les clusters Amazon EKS. Pour configurer chaque hôte, `kube-proxy` demande une résolution DNS pour le point de terminaison de votre cluster Amazon EKS.

1. Modifiez le DaemonSet `kube-proxy` à l'aide de la commande suivante

   ```
   kubectl -n kube-system edit ds kube-proxy
   ```

   Cela ouvrira la définition du DaemonSet `kube-proxy` dans l'éditeur que vous avez configuré.

2. Ajoutez les variables d'environnement pour `HTTP_PROXY` et `HTTPS_PROXY`. Notez que la variable d'environnement `NODE_NAME` doit déjà exister dans votre configuration. Remplacez `proxy-domain` et `port` par les valeurs correspondant à votre environnement.

   ```
   containers:
     - command:
       - kube-proxy
       - --v=2
       - --config=/var/lib/kube-proxy-config/config - --hostname-override=$(NODE_NAME)
       env:
       - name: HTTP_PROXY
         value: http://proxy-domain:port
       - name: HTTPS_PROXY
         value: http://proxy-domain:port
       - name: NODE_NAME
         valueFrom:
           fieldRef:
             apiVersion: v1
             fieldPath: spec.nodeName
   ```

Configurer Cilium BGP pour les nœuds hybrides

Cette rubrique décrit comment configurer le protocole de passerelle frontière (BGP) de Cilium pour les nœuds hybrides Amazon EKS. La fonctionnalité BGP de Cilium s'appelle le plan de contrôle Cilium BGP et peut être utilisée pour diffuser les adresses des pods et des services vers votre réseau sur site. Pour connaître les autres méthodes permettant de rendre les CIDR pod routables sur votre réseau local, consultez the section called "CIDR de pods distants routables".

Configurer Cilium BGP

Prérequis

- Cilium installé en suivant les instructions dans the section called "Configurer CNI".

Procédure

1. Pour utiliser BGP avec Cilium afin de publier les adresses des pods ou des services sur votre réseau local, Cilium doit être installé avec `bgpControlPlane.enabled: true`. Si vous activez BGP pour un déploiement Cilium existant, vous devez redémarrer l'opérateur Cilium pour appliquer la configuration BGP si BGP n'était pas activé auparavant. Vous pouvez définir `operator.rollOutPods` sur `true` dans vos valeurs Helm pour redémarrer l'opérateur Cilium dans le cadre du processus d'installation/mise à niveau Helm.

```
helm upgrade cilium oci://public.ecr.aws/eks/cilium/cilium \
  --namespace kube-system \
  --reuse-values \
  --set operator.rollOutPods=true \
  --set bgpControlPlane.enabled=true
```

2. Vérifiez que l'opérateur Cilium et les agents ont été redémarrés et fonctionnent correctement.

```
kubectl -n kube-system get pods --selector=app.kubernetes.io/part-of=cilium
```

```
NAME                               READY   STATUS    RESTARTS   AGE
cilium-grwlc                       1/1     Running   0          4m12s
cilium-operator-68f7766967-5nnbl   1/1     Running   0          4m20s
cilium-operator-68f7766967-7spfz   1/1     Running   0          4m20s
cilium-pnxcv                       1/1     Running   0          6m29s
cilium-r7qkj                       1/1     Running   0          4m12s
cilium-wxhfn                       1/1     Running   0          4m1s
cilium-z7hlb                       1/1     Running   0          6m30s
```

3. Créez un fichier nommé `cilium-bgp-cluster.yaml` avec une définition `CiliumBGPClusterConfig`. Vous devrez peut-être obtenir les informations suivantes auprès de votre administrateur réseau.
   - Configurez `localASN` avec l'ASN pour les nœuds exécutant Cilium.
   - Configurez `peerASN` avec l'ASN pour votre routeur local.

- Configurez le `peerAddress` du routeur local avec lequel chaque nœud exécutant Cilium sera appairé.

```
apiVersion: cilium.io/v2alpha1
kind: CiliumBGPClusterConfig
metadata:
  name: cilium-bgp
spec:
  nodeSelector:
    matchExpressions:
    - key: eks.amazonaws.com/compute-type
      operator: In
      values:
      - hybrid
  bgpInstances:
  - name: "rack0"
    localASN: NODES_ASN
    peers:
    - name: "onprem-router"
      peerASN: ONPREM_ROUTER_ASN
      peerAddress: ONPREM_ROUTER_IP
      peerConfigRef:
        name: "cilium-peer"
```

4. Appliquer la configuration du cluster Cilium BGP à votre cluster.

```
kubectl apply -f cilium-bgp-cluster.yaml
```

5. Créez un fichier nommé `cilium-bgp-peer.yaml` d'après la ressource `CiliumBGPPeerConfig` qui définit une configuration de pair BGP. Plusieurs pairs peuvent partager la même configuration et fournir une référence à la ressource `CiliumBGPPeerConfig` commune. Consultez la configuration BGP Peer dans la documentation Cilium pour obtenir la liste complète des options de configuration.

   Les valeurs des paramètres Cilium peer suivants doivent correspondre à celles du routeur local avec lequel vous établissez une connexion peer-to-peer.

   - Configurez `holdTimeSeconds`, qui détermine la durée pendant laquelle un pair BGP attend un message de maintien ou de mise à jour avant de déclarer la session comme interrompue. La durée par défaut est de 90 secondes.

- Configurez `keepAliveTimeSeconds`, ce qui détermine si un pair BGP est toujours accessible et si la session BGP est active. Le durée par défaut est 30 secondes.
- Configurez `restartTimeSeconds`, le délai pendant lequel le plan de contrôle BGP de Cilium doit rétablir la session BGP après un redémarrage. La durée par défaut est de 120 secondes.

```
apiVersion: cilium.io/v2alpha1
kind: CiliumBGPPeerConfig
metadata:
  name: cilium-peer
spec:
  timers:
    holdTimeSeconds: 90
    keepAliveTimeSeconds: 30
  gracefulRestart:
    enabled: true
    restartTimeSeconds: 120
  families:
    - afi: ipv4
      safi: unicast
      advertisements:
        matchLabels:
          advertise: "bgp"
```

6. Appliquez la configuration de pair Cilium BGP à votre cluster.

```
kubectl apply -f cilium-bgp-peer.yaml
```

7. Créez un fichier nommé `cilium-bgp-advertisement-pods.yaml` avec une ressource `CiliumBGPAdvertisement` pour annoncer les CIDR du pod à votre réseau sur site.
   - Cette ressource `CiliumBGPAdvertisement` sert à définir les types de publicités et les attributs qui leur sont associés. L'exemple ci-dessous configure Cilium pour qu'il n'annonce que les CIDR des pods. Consultez les exemples dans the section called "Type de service LoadBalancer" et the section called "Équilibrage de charge dans le cluster Cilium" pour plus d'informations sur la configuration de Cilium afin d'annoncer les adresses de service.
   - Chaque nœud hybride exécutant l'agent Cilium est apparié avec le routeur en amont compatible BGP. Chaque nœud annonce la plage CIDR du pod dont il est propriétaire lorsque le `advertisementType` de Cilium est configuré sur `PodCIDR` comme dans l'exemple ci-dessous. Pour plus d'informations, consultez la configuration des annonces BGP dans la documentation Cilium.

```
apiVersion: cilium.io/v2alpha1
kind: CiliumBGPAdvertisement
metadata:
  name: bgp-advertisement-pods
  labels:
    advertise: bgp
spec:
  advertisements:
    - advertisementType: "PodCIDR"
```

8. Appliquez la configuration Cilium BGP Advertisement à votre cluster.

```
kubectl apply -f cilium-bgp-advertisement-pods.yaml
```

9. Vous pouvez vérifier que le peering BGP fonctionne avec l'interface CLI Cilium à l'aide de la commande `cilium bgp peers`. Vous devriez voir les valeurs correctes dans la sortie pour votre environnement et l'état de session cen tant que `established`. Pour plus d'informations sur le dépannage, consultez le guide de dépannage et d'utilisation dans la documentation Cilium.

   Dans les exemples ci-dessous, cinq nœuds hybrides exécutent l'agent Cilium et chaque nœud annonce la plage CIDR Pod qu'il possède.

```
cilium bgp peers
```

```
Node                   Local AS    Peer AS               Peer Address        Session
 State   Uptime     Family         Received   Advertised
mi-026d6a261e355fba7   NODES_ASN
                   ONPREM_ROUTER_ASN
                   ONPREM_ROUTER_IP    established     1h18m58s   ipv4/unicast   1
      2
mi-082f73826a163626e   NODES_ASN
                   ONPREM_ROUTER_ASN
                   ONPREM_ROUTER_IP    established     1h19m12s   ipv4/unicast   1
      2
mi-09183e8a3d755abf6   NODES_ASN
                   ONPREM_ROUTER_ASN
                   ONPREM_ROUTER_IP    established     1h18m47s   ipv4/unicast   1
      2
mi-0d78d815980ed202d   NODES_ASN
                   ONPREM_ROUTER_ASN
```

```
                  ONPREM_ROUTER_IP    established     1h19m12s   ipv4/unicast   1
      2
mi-0daa253999fe92daa   NODES_ASN
                  ONPREM_ROUTER_ASN
                  ONPREM_ROUTER_IP    established     1h18m58s   ipv4/unicast   1
      2
```

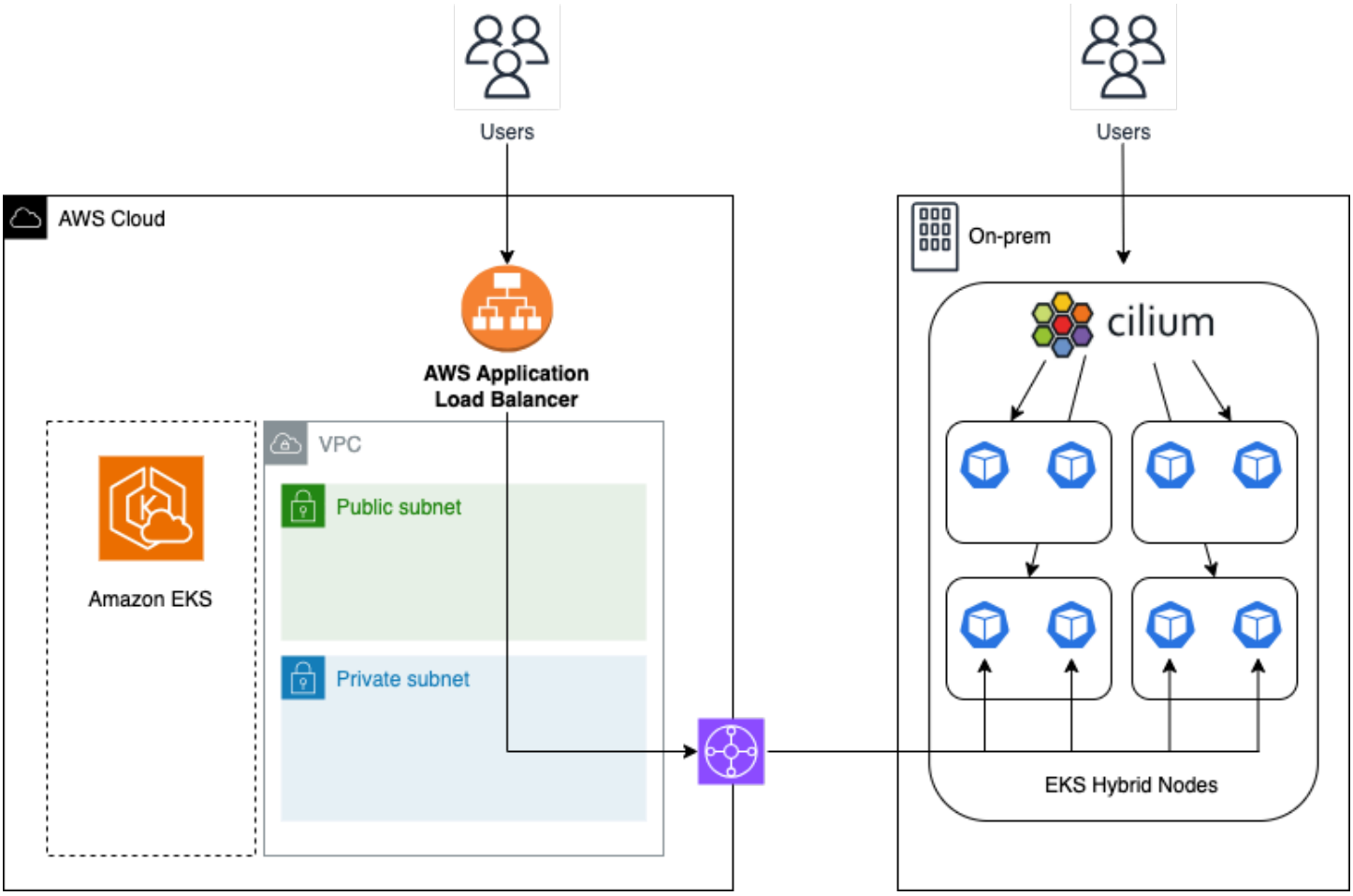
```
cilium bgp routes
```

```
Node                   VRouter       Prefix           NextHop   Age         Attrs
mi-026d6a261e355fba7   NODES_ASN     10.86.2.0/26     0.0.0.0   1h16m46s   [{Origin:
 i} {Nexthop: 0.0.0.0}]
mi-082f73826a163626e   NODES_ASN     10.86.2.192/26   0.0.0.0   1h16m46s   [{Origin:
 i} {Nexthop: 0.0.0.0}]
mi-09183e8a3d755abf6   NODES_ASN     10.86.2.64/26    0.0.0.0   1h16m46s   [{Origin:
 i} {Nexthop: 0.0.0.0}]
mi-0d78d815980ed202d   NODES_ASN     10.86.2.128/26   0.0.0.0   1h16m46s   [{Origin:
 i} {Nexthop: 0.0.0.0}]
mi-0daa253999fe92daa   NODES_ASN     10.86.3.0/26     0.0.0.0   1h16m46s   [{Origin:
 i} {Nexthop: 0.0.0.0}]
```

Configurer Kubernetes Ingress pour les nœuds hybrides

Cette rubrique décrit comment configurer Kubernetes Ingress pour les charges de travail exécutées sur des nœuds hybrides Amazon EKS. Kubernetes Ingress expose les routes HTTP et HTTPS depuis l'extérieur du cluster vers les services au sein du cluster. Pour utiliser les ressources Ingress, un contrôleur Kubernetes Ingress est nécessaire pour configurer l'infrastructure réseau et les composants qui gèrent le trafic réseau.

AWS prend en charge AWS Application Load Balancer (ALB) et Cilium pour Kubernetes Ingress pour les charges de travail exécutées sur des nœuds hybrides EKS. La décision d'utiliser ALB ou Cilium pour Ingress dépend de la source du trafic applicatif. Si le trafic applicatif provient d'une région AWS, AWS recommande d'utiliser AWS ALB et l'AWS Load Balancer Controller. Si le trafic applicatif provient de l'environnement local sur site ou périphérique, AWS recommande d'utiliser les fonctionnalités Ingress intégrées à Cilium, qui peuvent être utilisées avec ou sans infrastructure d'équilibrage de charge dans votre environnement.



AWS Application Load Balancer

Vous pouvez utiliser l'AWS Load Balancer Controller et l'Application Load Balancer (ALB) avec le type de cible `ip` pour les charges de travail s'exécutant sur des nœuds hybrides. Lorsque vous utilisez le type de cible `ip`, ALB transfère le trafic directement vers les pods, en contournant le chemin réseau de la couche Service. Pour qu'ALB atteigne les cibles IP des pods sur les nœuds hybrides, votre CIDR de pod sur site doit être routable sur votre réseau sur site. De plus, l'AWS Load Balancer Controller utilise des webhooks et nécessite une communication directe depuis le plan de contrôle EKS. Pour de plus amples informations, consultez the section called "Configuration de webhooks".

Considérations

- Voir the section called "Répartition de la charge des applications" et the section called "Installer avec Helm" pour plus d'informations sur AWS Application Load Balancer et AWS Load Balancer Controller.

- Consultez la section Meilleures pratiques pour l'équilibrage de charge pour savoir comment choisir entre AWSApplication Load Balancer et AWS Network Load Balancer.
- Consultez les annotations sur l'AWS Load Balancer Controller Ingress pour obtenir la liste des annotations pouvant être configurées pour les ressources d'entrée avec AWS Application Load Balancer.

Prérequis

- Cilium installé en suivant les instructions dans the section called "Configurer CNI".
- Le plan de contrôle Cilium BGP a été activé en suivant les instructions fournies dans the section called "Configuration de BGP". Si vous ne souhaitez pas utiliser BGP, vous devez utiliser une autre méthode pour rendre vos CIDR de pods sur site routables sur votre réseau sur site. Si vous ne rendez pas vos CIDR de pods sur site routables, ALB ne pourra pas enregistrer ni contacter vos cibles IP de pods.
- Helm installé dans votre environnement de ligne de commande, consultez les instructions de configuration de Helm pour plus d'informations.???
- eksctl installé dans votre environnement de ligne de commande, consultez les instructions d'installation d'eksctl pour plus d'informations.

Procédure

1. Téléchargez une politique IAM pour le contrôleur de l'équilibreur de charge AWS qui lui permet d'effectuer des appels aux API AWS en votre nom.

```
curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-
controller/refs/heads/main/docs/install/iam_policy.json
```

2. Créez une politique IAM à l'aide de la politique téléchargée à l'étape précédente.

```
aws iam create-policy \
    --policy-name AWSLoadBalancerControllerIAMPolicy \
    --policy-document file://iam_policy.json
```

3. Remplacez la valeur du nom du cluster (`CLUSTER_NAME`), de AWS la région (`AWS_REGION`) et de l'ID de AWS compte (`AWS_ACCOUNT_ID`) par vos paramètres et exécutez la commande suivante.

```
eksctl create iamserviceaccount \
```

```
    --cluster=CLUSTER_NAME \
    --namespace=kube-system \
    --name=aws-load-balancer-controller \
    --attach-policy-arn=arn:aws:iam::AWS_ACCOUNT_ID:policy/
AWSLoadBalancerControllerIAMPolicy \
    --override-existing-serviceaccounts \
    --region AWS_REGION \
    --approve
```

4. Ajoutez le référentiel des Charts de Helm eks-charts et mettez à jour votre référentiel Helm local pour vous assurer que vous disposez des graphiques les plus récents.

```
helm repo add eks https://aws.github.io/eks-charts
```

```
helm repo update eks
```

5. Installez le contrôleur de l'équilibreur de charge AWS. Remplacez les valeurs correspondant au nom du cluster (`CLUSTER_NAME`), à la région AWS (`AWS_REGION`), à l'ID VPC `VPC_ID`() et à la version des Charts de Helm de l'AWS Load Balancer Controller (`AWS_LBC_HELM_VERSION`) par vos paramètres, puis exécutez la commande suivante. Si vous utilisez un cluster en mode mixte avec des nœuds hybrides et des nœuds dans le cloud AWS, vous pouvez exécuter l'AWS Load Balancer Controller sur les nœuds cloud en suivant les instructions disponibles à l'adresse the section called "Contrôleur de l'équilibreur de charge AWS".
   - Vous pouvez trouver la dernière version des Charts de Helm en exécutant `helm search repo eks/aws-load-balancer-controller --versions`.

```
helm install aws-load-balancer-controller eks/aws-load-balancer-controller \
  -n kube-system \
  --version AWS_LBC_HELM_VERSION \
  --set clusterName=CLUSTER_NAME \
  --set region=AWS_REGION \
  --set vpcId=VPC_ID \
  --set serviceAccount.create=false \
  --set serviceAccount.name=aws-load-balancer-controller
```

6. Vérifiez que l'AWS Load Balancer Controller a été installé correctement.

```
kubectl get -n kube-system deployment aws-load-balancer-controller
```

```
NAME                           READY   UP-TO-DATE   AVAILABLE   AGE
```

```
aws-load-balancer-controller   2/2     2            2           84s
```

7. Créer un exemple d'application. L'exemple ci-dessous utilise l'application microservices Istio Bookinfo.

```
kubectl apply -f https://raw.githubusercontent.com/istio/istio/refs/heads/master/
samples/bookinfo/platform/kube/bookinfo.yaml
```

8. Créez un fichier nommé `my-ingress-alb.yaml` avec les contenus suivants.

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: my-ingress
  namespace: default
  annotations:
    alb.ingress.kubernetes.io/load-balancer-name: "my-ingress-alb"
    alb.ingress.kubernetes.io/target-type: "ip"
    alb.ingress.kubernetes.io/scheme: "internet-facing"
    alb.ingress.kubernetes.io/healthcheck-path: "/details/1"
spec:
  ingressClassName: alb
  rules:
  - http:
      paths:
      - backend:
          service:
            name: details
            port:
              number: 9080
        path: /details
        pathType: Prefix
```

9. Appliquez la configuration Ingress à votre cluster.

```
kubectl apply -f my-ingress-alb.yaml
```

10. La configuration de l'ALB pour votre ressource Ingress peut prendre quelques minutes. Une fois l'ALB provisionné, votre ressource Ingress se verra attribuer une adresse correspondant au nom DNS du déploiement ALB. L'adresse aura le format `<alb-name>-<random-string>.<region>.elb.amazonaws.com`.

```
kubectl get ingress my-ingress
```

```
NAME         CLASS   HOSTS   ADDRESS
    PORTS   AGE
my-ingress   alb     *       my-ingress-alb-<random-
string>.<region>.elb.amazonaws.com   80      23m
```

11 Accédez au service à l'aide de l'adresse de l'ALB.

```
curl -s http//my-ingress-alb-<random-string>.<region>.elb.amazonaws.com:80/details/1
 | jq
```

```
{
  "id": 1,
  "author": "William Shakespeare",
  "year": 1595,
  "type": "paperback",
  "pages": 200,
  "publisher": "PublisherA",
  "language": "English",
  "ISBN-10": "1234567890",
  "ISBN-13": "123-1234567890"
  "details": "This is the details page"
}
```

Présentation de Cilium Ingress et Cilium Gateway

Les fonctionnalités Ingress de Cilium sont intégrées à l'architecture de Cilium et peuvent être gérées à l'aide de l'API Ingress ou de l'API Gateway de Kubernetes. Si vous ne disposez pas encore de ressources Ingress, AWS recommandoe de commencer par l'API Gateway, car elle offre un moyen plus expressif et plus flexible de définir et de gérer les ressources réseau Kubernetes. L'API Kubernetes Gateway vise à normaliser la manière dont les ressources réseau pour Ingress, l'équilibreur de charge et le maillage de services sont définies et gérées dans les clusters Kubernetes.

Lorsque vous activez les fonctionnalités Ingress ou Gateway de Cilium, l'opérateur Cilium synchronise les objets Ingress / Gateway dans le cluster et les proxys Envoy sur chaque nœud traitent le trafic réseau de couche 7 (L7). Cilium ne fournit pas directement d'infrastructure Ingress /

Gateway telle que des équilibreurs de charge. Si vous prévoyez d'utiliser Cilium Ingress / Gateway avec un équilibreur de charge, vous devez utiliser les outils de l'équilibreur de charge, généralement un contrôleur Ingress ou Gateway, pour déployer et gérer l'infrastructure de l'équilibreur de charge.

Pour le trafic Ingress/Gateway, Cilium gère le trafic réseau central et l'application des politiques L3/L4, tandis que les proxys Envoy intégrés traitent le trafic réseau L7. Avec Cilium Ingress / Gateway, Envoy est chargé d'appliquer les règles de routage L7, les politiques et la manipulation des requêtes, la gestion avancée du trafic telle que la répartition et la mise en miroir du trafic, ainsi que la terminaison et l'origine TLS. Les proxys Envoy de Cilium sont déployés par défaut sous forme de DaemonSet (`cilium-envoy`) distinct, ce qui permet de mettre à jour, de dimensionner et de gérer séparément Envoy et l'agent Cilium.

Pour plus d'informations sur le fonctionnement de Cilium Ingress et Cilium Gateway, consultez les pages Cilium Ingress et Cilium Gateway dans la documentation Cilium.

Comparaison entre Cilium Ingress et Gateway

Le tableau ci-dessous résume les fonctionnalités de Cilium Ingress et Cilium Gateway dans la version 1.17.x de Cilium.

Fonctionnalité	Ingress	Passerelle
Type de service LoadBalancer	Oui	Oui
Type de service NodePort	Oui	N°[1]
Réseau hôte	Oui	Oui
Équilibreur de charge partagé	Oui	Oui
Équilibreur de charge dédié	Oui	No[2]
Stratégies réseau	Oui	Oui
Protocoles	Couche 7 (HTTP (S), gRPC)	Couche 7 (HTTP(S), gRPC)[3]
TLS Passthrough	Oui	Oui
Gestion du trafic	Routage du chemin et d'hôte	Routage de chemin et d'hôte, redirection

Fonctionnalité	Ingress	Passerelle
		et réécriture d'URL, répartition du trafic, modification d'en-tête

[1] La prise en charge de Cilium Gateway pour les services NodePort est prévue pour la version 1.18.x de Cilium (#27273)

[2] Prise en charge de Cilium Gateway pour les équilibreurs de charge dédiés (#25567)

[3] Prise en charge de Cilium Gateway pour TCP/UDP (#21929)

Installer Cilium Gateway

Considérations

- Cilium doit être configuré avec `nodePort.enabled` réglé sur `true`, comme indiqué dans les exemples ci-dessous. Si vous utilisez la fonctionnalité de remplacement kube-proxy de Cilium, vous n'avez pas besoin de définir `nodePort.enabled` sur `true`.
- Cilium doit être configuré avec `envoy.enabled` réglé sur `true`, comme indiqué dans les exemples ci-dessous.
- Cilium Gateway peut être déployé en mode équilibreur de charge (par défaut) ou en mode réseau hôte.
- Lorsque vous utilisez Cilium Gateway en mode équilibreur de charge, l'annotation `service.beta.kubernetes.io/aws-load-balancer-type: "external"` doit être définie sur la ressource Gateway afin d'empêcher le fournisseur de cloud hérité AWS de créer un Classic Load Balancer pour le service de type LoadBalancer que Cilium crée pour la ressource Gateway.
- Lorsque vous utilisez Cilium Gateway en mode réseau hôte, le service de type LoadBalancer est désactivé. Le mode réseau hôte est utile pour les environnements qui ne disposent pas d'une infrastructure d'équilibrage de charge. Pour plus d'informations, consultez the section called "Réseau hôte".

Prérequis

1. Helm installé dans votre environnement de ligne de commande, consultez les Instructions de configuration de Helm.

2. Cilium installé en suivant les instructions dans the section called "Configurer CNI".

Procédure

1. Installez les définitions de ressources personnalisées (CRD) de l'API Kubernetes Gateway.

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-sigs/gateway-api/
v1.2.1/config/crd/standard/gateway.networking.k8s.io_gatewayclasses.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-sigs/gateway-api/
v1.2.1/config/crd/standard/gateway.networking.k8s.io_gateways.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-sigs/gateway-api/
v1.2.1/config/crd/standard/gateway.networking.k8s.io_httproutes.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-sigs/gateway-api/
v1.2.1/config/crd/standard/gateway.networking.k8s.io_referencegrants.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-sigs/gateway-api/
v1.2.1/config/crd/standard/gateway.networking.k8s.io_grpcroutes.yaml
```

2. Créez un fichier nommé `cilium-gateway-values.yaml` avec le contenu suivant. L'exemple ci-dessous configure Cilium Gateway pour utiliser le mode d'équilibrage de charge par défaut et utiliser un DaemonSet `cilium-envoy` distinct pour les proxys Envoy configurés pour s'exécuter uniquement sur des nœuds hybrides.

```
gatewayAPI:
  enabled: true
  # uncomment to use host network mode
  # hostNetwork:
  #   enabled: true
nodePort:
  enabled: true
envoy:
  enabled: true
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: eks.amazonaws.com/compute-type
            operator: In
            values:
            - hybrid
```

3. Appliquez le fichier de valeurs Helm à votre cluster.

```
helm upgrade cilium oci://public.ecr.aws/eks/cilium/cilium \
  --namespace kube-system \
  --reuse-values \
  --set operator.rollOutPods=true \
  --values cilium-gateway-values.yaml
```

4. Vérifiez que l'opérateur Cilium, l'agent et les pods Envoy fonctionnent correctement.

```
kubectl -n kube-system get pods --selector=app.kubernetes.io/part-of=cilium
```

```
NAME                               READY   STATUS    RESTARTS   AGE
cilium-envoy-5pgnd                 1/1     Running   0          6m31s
cilium-envoy-6fhg4                 1/1     Running   0          6m30s
cilium-envoy-jskrk                 1/1     Running   0          6m30s
cilium-envoy-k2xtb                 1/1     Running   0          6m31s
cilium-envoy-w5s9j                 1/1     Running   0          6m31s
cilium-grwlc                       1/1     Running   0          4m12s
cilium-operator-68f7766967-5nnbl   1/1     Running   0          4m20s
cilium-operator-68f7766967-7spfz   1/1     Running   0          4m20s
cilium-pnxcv                       1/1     Running   0          6m29s
cilium-r7qkj                       1/1     Running   0          4m12s
cilium-wxhfn                       1/1     Running   0          4m1s
cilium-z7hlb                       1/1     Running   0          6m30s
```

Configurer la passerelle Cilium

Cilium Gateway est activé sur les objets Gateway en définissant le paramètre `gatewayClassName` sur `cilium`. Le service créé par Cilium pour les ressources Gateway peut être configuré à l'aide des champs de l'objet Gateway. Les annotations couramment utilisées par les contrôleurs Gateway pour configurer l'infrastructure d'équilibrage de charge peuvent être configurées à l'aide du champ `infrastructure` de l'objet Gateway. Lorsque vous utilisez l'IPAM LoadBalancer de Cilium (voir exemple dans the section called "Type de service LoadBalancer"), l'adresse IP à utiliser pour le service de type LoadBalancer peut être configurée dans le champ `addresses` de l'objet Gateway. Pour plus d'informations sur la configuration de la passerelle, consultez la spécification de l'API Kubernetes Gateway.

```
apiVersion: gateway.networking.k8s.io/v1
kind: Gateway
metadata:
```

```
  name: my-gateway
spec:
  gatewayClassName: cilium
  infrastructure:
    annotations:
      service.beta.kubernetes.io/...
      service.kuberentes.io/...
  addresses:
  - type: IPAddress
    value: <LoadBalancer IP address>
  listeners:
  ...
```

Cilium et la spécification Kubernetes Gateway prennent en charge les ressources GatewayClass, Gateway, HTTPRoute, GRPCRoute et ReferenceGrant.

- Consultez les spécifications HTTPRoute et GRPCRoute pour obtenir la liste des champs disponibles.
- Consultez les exemples dans la section the section called "Déployer Cilium Gateway" ci-dessous et ceux dans la documentation Cilium pour savoir comment utiliser et configurer ces ressources.

Déployer Cilium Gateway

1. Créer un exemple d'application. L'exemple ci-dessous utilise l'application microservices Istio Bookinfo.

   ```
   kubectl apply -f https://raw.githubusercontent.com/istio/istio/refs/heads/master/
   samples/bookinfo/platform/kube/bookinfo.yaml
   ```

2. Vérifiez que l'application fonctionne correctement.

   ```
   kubectl get pods
   ```

   ```
   NAME                              READY   STATUS    RESTARTS   AGE
   details-v1-766844796b-9965p       1/1     Running   0          81s
   productpage-v1-54bb874995-jmc8j   1/1     Running   0          80s
   ratings-v1-5dc79b6bcd-smzxz       1/1     Running   0          80s
   reviews-v1-598b896c9d-vj7gb       1/1     Running   0          80s
   reviews-v2-556d6457d-xbt8v        1/1     Running   0          80s
   reviews-v3-564544b4d6-cpmvq       1/1     Running   0          80s
   ```

3. Créez un fichier nommé `my-gateway.yaml` avec les contenus suivants. L'exemple ci-dessous utilise l'annotation `service.beta.kubernetes.io/aws-load-balancer-type: "external"` pour empêcher l'ancien fournisseur de cloud AWS de créer un Classic Load Balancer pour le service de type LoadBalancer que Cilium crée pour la ressource Gateway.

```
---
apiVersion: gateway.networking.k8s.io/v1
kind: Gateway
metadata:
  name: my-gateway
spec:
  gatewayClassName: cilium
  infrastructure:
    annotations:
      service.beta.kubernetes.io/aws-load-balancer-type: "external"
  listeners:
  - protocol: HTTP
    port: 80
    name: web-gw
    allowedRoutes:
      namespaces:
        from: Same
---
apiVersion: gateway.networking.k8s.io/v1
kind: HTTPRoute
metadata:
  name: http-app-1
spec:
  parentRefs:
  - name: my-gateway
    namespace: default
  rules:
  - matches:
    - path:
        type: PathPrefix
        value: /details
    backendRefs:
    - name: details
      port: 9080
```

4. Appliquez la ressource Gateway à votre cluster.

```
kubectl apply -f my-gateway.yaml
```

5. Confirmez que la ressource Gateway et le service correspondant ont été créés. À ce stade, il est prévu que le champ ADDRESS de la ressource Gateway ne soit pas renseigné avec une adresse IP ou un nom d'hôte, et que le service de type LoadBalancer pour la ressource Gateway ne dispose pas non plus d'adresse IP ou de nom d'hôte attribué.

```
kubectl get gateway my-gateway
```

```
NAME         CLASS    ADDRESS   PROGRAMMED   AGE
my-gateway   cilium             True         10s
```

```
kubectl get svc cilium-gateway-my-gateway
```

```
NAME                         TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)
    AGE
cilium-gateway-my-gateway   LoadBalancer   172.16.227.247   <pending>     80:30912/
TCP   24s
```

6. Procédez à the section called "Type de service LoadBalancer" pour configurer la ressource Gateway afin d'utiliser une adresse IP attribuée par Cilium Load Balancer IPAM, et the section called "Type de service NodePort" ou the section called "Réseau hôte" pour configurer la ressource Gateway afin d'utiliser les adresses NodePort ou réseau hôte.

Installer Cilium Ingress

Considérations

- Cilium doit être configuré avec `nodePort.enabled` réglé sur `true`, comme indiqué dans les exemples ci-dessous. Si vous utilisez la fonctionnalité de remplacement kube-proxy de Cilium, vous n'avez pas besoin de définir `nodePort.enabled` sur `true`.
- Cilium doit être configuré avec `envoy.enabled` réglé sur `true`, comme indiqué dans les exemples ci-dessous.
- Lorsque `ingressController.loadbalancerMode` est défini sur `dedicated`, Cilium crée des services dédiés pour chaque ressource Ingress. Lorsque `ingressController.loadbalancerMode` est défini sur `shared`, Cilium crée un service

partagé de type LoadBalancer pour toutes les ressources Ingress du cluster. Lorsque vous utilisez le mode équilibreur de charge `shared`, les paramètres du service partagé tels que `labels`, `annotations`, `type` et `loadBalancerIP` sont configurés dans la section `ingressController.service` des valeurs Helm. Pour plus d'informations, consultez la référence sur les valeurs Cilium Helm.

- Avec `ingressController.default` défini sur `true`, Cilium est configuré comme contrôleur Ingress par défaut pour le cluster et créera des entrées Ingress même lorsque `ingressClassName` n'est pas spécifié sur les ressources Ingress.
- Cilium Ingress peut être déployé en mode équilibreur de charge (par défaut), port de nœud ou réseau hôte. Lorsque Cilium est installé en mode réseau hôte, les modes Service de type LoadBalancer et Service de type NodePort sont désactivés. Pour plus d'informations, consultez the section called "Réseau hôte".
- Définissez toujours `ingressController.service.annotations` sur `service.beta.kubernetes.io/aws-load-balancer-type: "external"` dans les valeurs Helm afin d'empêcher l'ancien fournisseur de cloud AWS de créer un Classic Load Balancer pour le service par défaut `cilium-ingress` créé par les Charts de Helm Cilium.

Prérequis

1. Helm installé dans votre environnement de ligne de commande, consultez les Instructions de configuration de Helm.
2. Cilium installé en suivant les instructions dans the section called "Configurer CNI".

Procédure

1. Créez un fichier nommé `cilium-ingress-values.yaml` avec le contenu suivant. L'exemple ci-dessous configure Cilium Ingress pour utiliser le mode d'équilibrage de charge `dedicated` par défaut et utiliser un DaemonSet distinct `cilium-envoy` pour les proxys Envoy configurés pour s'exécuter uniquement sur des nœuds hybrides.

```
ingressController:
  enabled: true
  loadbalancerMode: dedicated
  service:
    annotations:
      service.beta.kubernetes.io/aws-load-balancer-type: "external"
nodePort:
```

```
  enabled: true
envoy:
  enabled: true
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: eks.amazonaws.com/compute-type
            operator: In
            values:
            - hybrid
```

2. Appliquez le fichier de valeurs Helm à votre cluster.

```
helm upgrade cilium oci://public.ecr.aws/eks/cilium/cilium \
  --namespace kube-system \
  --reuse-values \
  --set operator.rollOutPods=true \
  --values cilium-ingress-values.yaml
```

3. Vérifiez que l'opérateur Cilium, l'agent et les pods Envoy fonctionnent correctement.

```
kubectl -n kube-system get pods --selector=app.kubernetes.io/part-of=cilium
```

```
NAME                               READY   STATUS    RESTARTS   AGE
cilium-envoy-5pgnd                 1/1     Running   0          6m31s
cilium-envoy-6fhg4                 1/1     Running   0          6m30s
cilium-envoy-jskrk                 1/1     Running   0          6m30s
cilium-envoy-k2xtb                 1/1     Running   0          6m31s
cilium-envoy-w5s9j                 1/1     Running   0          6m31s
cilium-grwlc                       1/1     Running   0          4m12s
cilium-operator-68f7766967-5nnbl   1/1     Running   0          4m20s
cilium-operator-68f7766967-7spfz   1/1     Running   0          4m20s
cilium-pnxcv                       1/1     Running   0          6m29s
cilium-r7qkj                       1/1     Running   0          4m12s
cilium-wxhfn                       1/1     Running   0          4m1s
cilium-z7hlb                       1/1     Running   0          6m30s
```

Configurer Cilium Ingress

Cilium Ingress est activé sur les objets Ingress en définissant le paramètre `ingressClassName` sur `cilium`. Les services créés par Cilium pour les ressources Ingress peuvent être configurés à l'aide d'annotations sur les objets Ingress lorsque vous utilisez le mode équilibreur de charge `dedicated`, et dans la configuration Cilium / Helm lorsque vous utilisez le mode équilibreur de charge `shared`. Ces annotations sont couramment utilisées par les contrôleurs Ingress pour configurer l'infrastructure de l'équilibreur de charge ou d'autres attributs du service, tels que le type de service, le mode de l'équilibreur de charge, les ports et le transfert TLS. Les annotations clés sont décrites ci-dessous. Pour obtenir la liste complète des annotations prises en charge, consultez les annotations Cilium Ingress dans la documentation Cilium.

Annotation	Description
`ingress.cilium.io/loadbalancer-mode`	`dedicated` : Service dédié de type LoadBalancer pour chaque ressource Ingress (par défaut). `shared` : Service unique de type LoadBalancer pour toutes les ressources Ingress.
`ingress.cilium.io/service-type`	`LoadBalancer` : le service sera de type LoadBalancer (par défaut) `NodePort` : Le service sera de type NodePort.
`service.beta.kubernetes.io/aws-load-balancer-type`	`"external"` : Empêcher les anciens fournisseurs de cloud AWS de provisionner un Classic Load Balancer pour les services de type LoadBalancer.
`lbipam.cilium.io/ips`	Liste des adresses IP à attribuer à partir de Cilium LoadBalancer IPAM

Cilium et la spécification Kubernetes Ingress prennent en charge les règles de correspondance exacte, préfixe et spécifique à l'implémentation pour les chemins d'accès Ingress. Cilium prend en charge les expressions régulières comme règle de correspondance spécifique à son implémentation. Pour plus d'informations, consultez les sections Types et priorité des chemins d'entrée et Exemples de types de chemins dans la documentation Cilium, ainsi que les exemples présentés dans cette section the section called "Déployer Cilium Ingress".

Un exemple d'objet Cilium Ingress est présenté ci-dessous.

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: my-ingress
  annotations:
    service.beta.kuberentes.io/...
    service.kuberentes.io/...
spec:
  ingressClassName: cilium
  rules:
  ...
```

Déployer Cilium Ingress

1. Créer un exemple d'application. L'exemple ci-dessous utilise l'application microservices Istio Bookinfo.

   ```
   kubectl apply -f https://raw.githubusercontent.com/istio/istio/refs/heads/master/
   samples/bookinfo/platform/kube/bookinfo.yaml
   ```

2. Vérifiez que l'application fonctionne correctement.

   ```
   kubectl get pods
   ```

   ```
   NAME                              READY   STATUS    RESTARTS   AGE
   details-v1-766844796b-9965p       1/1     Running   0          81s
   productpage-v1-54bb874995-jmc8j   1/1     Running   0          80s
   ratings-v1-5dc79b6bcd-smzxz       1/1     Running   0          80s
   reviews-v1-598b896c9d-vj7gb       1/1     Running   0          80s
   reviews-v2-556d6457d-xbt8v        1/1     Running   0          80s
   reviews-v3-564544b4d6-cpmvq       1/1     Running   0          80s
   ```

3. Créez un fichier nommé `my-ingress.yaml` avec les contenus suivants. L'exemple ci-dessous utilise l'annotation `service.beta.kubernetes.io/aws-load-balancer-type: "external"` pour empêcher l'ancien fournisseur de cloud AWS de créer un Classic Load Balancer pour le service de type LoadBalancer que Cilium crée pour la ressource Ingress.

   ```
   apiVersion: networking.k8s.io/v1
   kind: Ingress
   ```

```
metadata:
  name: my-ingress
  namespace: default
  annotations:
    service.beta.kubernetes.io/aws-load-balancer-type: "external"
spec:
  ingressClassName: cilium
  rules:
  - http:
      paths:
      - backend:
          service:
            name: details
            port:
              number: 9080
        path: /details
        pathType: Prefix
```

4. Appliquez la ressource Ingress à votre cluster.

```
kubectl apply -f my-ingress.yaml
```

5. Confirmez que la ressource Ingress et le service correspondant ont été créés. À ce stade, il est prévu que le champ ADDRESS de la ressource Ingress ne soit pas renseigné avec une adresse IP ou un nom d'hôte, et que le service partagé ou dédié de type LoadBalancer pour la ressource Ingress ne dispose pas non plus d'adresse IP ou de nom d'hôte attribué.

```
kubectl get ingress my-ingress
```

```
NAME         CLASS    HOSTS   ADDRESS   PORTS   AGE
my-ingress   cilium   *                 80      8s
```

Pour le mode équilibreur de charge `shared`

```
kubectl -n kube-system get svc cilium-ingress
```

```
NAME             TYPE           CLUSTER-IP      EXTERNAL-IP   PORT(S)
      AGE
cilium-ingress   LoadBalancer   172.16.217.48   <pending>     80:32359/TCP,443:31090/
TCP   10m
```

Pour le mode équilibreur de charge `dedicated`

```
kubectl -n default get svc cilium-ingress-my-ingress
```

```
NAME                         TYPE           CLUSTER-IP      EXTERNAL-IP   PORT(S)
                  AGE
cilium-ingress-my-ingress    LoadBalancer   172.16.193.15   <pending>     80:32088/
TCP,443:30332/TCP   25s
```

6. Procédez à the section called "Type de service LoadBalancer" pour configurer la ressource Ingress afin d'utiliser une adresse IP attribuée par Cilium Load Balancer IPAM, et the section called "Type de service NodePort" ou the section called "Réseau hôte" pour configurer la ressource Ingress afin d'utiliser les adresses NodePort ou du réseau hôte.

Type de service LoadBalancer

Infrastructure existante de l'équilibreur de charge

Par défaut, pour Cilium Ingress et Cilium Gateway, Cilium crée des services Kubernetes de type LoadBalancer pour les ressources Ingress / Gateway. Les attributs du ou des services créés par Cilium peuvent être configurés via les ressources Ingress et Gateway. Lorsque vous créez des ressources Ingress ou Gateway, l'adresse IP ou les noms d'hôte exposés en externe pour Ingress ou Gateway sont attribués à partir de l'infrastructure d'équilibrage de charge, qui est généralement fournie par un contrôleur Ingress ou Gateway.

De nombreux contrôleurs Ingress et Gateway utilisent des annotations pour détecter et configurer l'infrastructure d'équilibrage de charge. Les annotations pour ces contrôleurs Ingress et Gateway sont configurées sur les ressources Ingress ou Gateway, comme indiqué dans les exemples précédents ci-dessus. Consultez la documentation de votre contrôleur Ingress ou Gateway pour connaître les annotations qu'il prend en charge et consultez la documentation Kubernetes Ingress et la documentation Kubernetes Gateway pour obtenir une liste des contrôleurs courants.

> **Important**
>
> Cilium Ingress et Gateway ne peuvent pas être utilisés avec l'AWS Load Balancer Controller et les Network Load Balancers (NLB) AWS avec les nœuds hybrides EKS. Tenter de les utiliser ensemble entraîne des cibles non enregistrées, car le NLB tente de se connecter directement aux adresses IP des pods qui prennent en charge le service de type

LoadBalancer lorsque le NLB `target-type` est défini sur `ip` (condition requise pour utiliser NLB avec des charges de travail s'exécutant sur des nœuds hybrides EKS).

Pas d'infrastructure d'équilibrage de charge

Si votre environnement ne dispose pas d'une infrastructure d'équilibrage de charge et du contrôleur Ingress / Gateway correspondant, les ressources Ingress / Gateway et les services de type LoadBalancer correspondants peuvent être configurés pour utiliser les adresses IP attribuées par la gestion des adresses IP de l'équilibreur de charge de Cilium (LB IPAM). L'IPAM du LB Cilium peut être configuré avec des plages d'adresses IP connues provenant de votre environnement sur site et peut utiliser la prise en charge intégrée du protocole de passerelle frontière (BGP) de Cilium ou les annonces L2 pour diffuser les adresses IP du LoadBalancer vers votre réseau sur site.

L'exemple ci-dessous montre comment configurer l'IPAM LB de Cilium avec une adresse IP à utiliser pour vos ressources Ingress / Gateway, et comment configurer le plan de contrôle BGP de Cilium pour annoncer l'adresse IP de l'équilibreur de charge au réseau sur site. La fonctionnalité LB IPAM de Cilium est activée par défaut, mais n'est pas activée tant qu'une ressource `CiliumLoadBalancerIPPool` n'est pas créée.

Prérequis

- Cilium Ingress ou Gateway installé conformément aux instructions fournies dans the section called "Installer Cilium Ingress" ou the section called "Installer Cilium Gateway".
- Ressources Cilium Ingress ou Gateway avec exemple d'application déployée en suivant les instructions dans the section called "Déployer Cilium Ingress" ou the section called "Déployer Cilium Gateway".
- Le plan de contrôle Cilium BGP a été activé en suivant les instructions fournies dans the section called "Configuration de BGP". Si vous ne souhaitez pas utiliser BGP, vous pouvez ignorer cette condition préalable, mais vous ne pourrez pas accéder à votre ressource Ingress ou Gateway tant que l'adresse IP LoadBalancer attribuée par Cilium LB IPAM ne sera pas routable sur votre réseau sur site.

Procédure

1. Vous pouvez éventuellement modifier la ressource Ingress ou Gateway afin de demander une adresse IP spécifique à utiliser pour le service de type LoadBalancer. Si vous ne demandez pas d'adresse IP spécifique, Cilium attribuera une adresse IP à partir de la plage d'adresses IP

configurée dans la ressource `CiliumLoadBalancerIPPool` à l'étape suivante. Dans les commandes ci-dessous, remplacez `LB_IP_ADDRESS` par l'adresse IP pour demander le service de type LoadBalancer.

Passerelle

```
kubectl patch gateway -n default my-gateway --type=merge -p '{
  "spec": {
    "addresses": [{"type": "IPAddress", "value": "LB_IP_ADDRESS"}]
  }
}'
```

Ingress

```
kubectl patch ingress my-ingress --type=merge -p '{
  "metadata": {"annotations": {"lbipam.cilium.io/ips": "LB_IP_ADDRESS"}}
}'
```

2. Créez un fichier nommé `cilium-lbip-pool-ingress.yaml` avec une ressource `CiliumLoadBalancerIPPool` pour configurer la plage d'adresses IP de l'équilibreur de charge pour vos ressources Ingress / Gateway.
   - Si vous utilisez Cilium Ingress, Cilium applique automatiquement le label `cilium.io/ingress: "true"` aux services qu'il crée pour les ressources Ingress. Vous pouvez utiliser cette étiquette dans le champ `serviceSelector` de la définition de la ressource `CiliumLoadBalancerIPPool` pour sélectionner les services éligibles à LB IPAM.
   - Si vous utilisez Cilium Gateway, vous pouvez utiliser l'étiquette `gateway.networking.k8s.io/gateway-name` dans les champs `serviceSelector` de la définition de ressource `CiliumLoadBalancerIPPool` pour sélectionner les ressources Gateway éligibles pour LB IPAM.
   - Remplacez `LB_IP_CIDR` par la plage d'adresses IP à utiliser pour les adresses IP de l'équilibreur de charge. Pour sélectionner une seule adresse IP, utilisez un CIDR `/32`. Pour plus d'informations, consultez la section Gestion des adresses IP LoadBalancer dans la documentation Cilium.

   ```
   apiVersion: cilium.io/v2alpha1
   kind: CiliumLoadBalancerIPPool
   metadata:
     name: bookinfo-pool
   ```

```
spec:
  blocks:
  - cidr: "LB_IP_CIDR"
  serviceSelector:
    # if using Cilium Gateway
    matchExpressions:
      - { key: gateway.networking.k8s.io/gateway-name, operator: In, values: [ my-
gateway ] }
    # if using Cilium Ingress
    matchLabels:
      cilium.io/ingress: "true"
```

3. Appliquez la ressource `CiliumLoadBalancerIPPool` à votre cluster.

```
kubectl apply -f cilium-lbip-pool-ingress.yaml
```

4. Confirmez qu'une adresse IP a été attribuée par Cilium LB IPAM pour la ressource Ingress / Gateway.

   Passerelle

```
kubectl get gateway my-gateway
```

```
NAME         CLASS    ADDRESS        PROGRAMMED   AGE
my-gateway   cilium   LB_IP_ADDRESS    True         6m41s
```

   Ingress

```
kubectl get ingress my-ingress
```

```
NAME         CLASS    HOSTS   ADDRESS        PORTS   AGE
my-ingress   cilium   *       LB_IP_ADDRESS   80      10m
```

5. Créez un fichier nommé `cilium-bgp-advertisement-ingress.yaml` avec une ressource `CiliumBGPAdvertisement` pour annoncer l'adresse IP du LoadBalancer pour les ressources Ingress / Gateway. Si vous n'utilisez pas Cilium BGP, vous pouvez ignorer cette étape. L'adresse IP du LoadBalancer utilisée pour votre ressource Ingress / Gateway doit être routable sur votre réseau local afin que vous puissiez interroger le service à l'étape suivante.

```
apiVersion: cilium.io/v2alpha1
```

```
kind: CiliumBGPAdvertisement
metadata:
  name: bgp-advertisement-lb-ip
  labels:
    advertise: bgp
spec:
  advertisements:
    - advertisementType: "Service"
      service:
        addresses:
          - LoadBalancerIP
      selector:
        # if using Cilium Gateway
        matchExpressions:
          - { key: gateway.networking.k8s.io/gateway-name, operator: In, values:
 [ my-gateway ] }
        # if using Cilium Ingress
        matchLabels:
          cilium.io/ingress: "true"
```

6. Appliquez la ressource `CiliumBGPAdvertisement` à votre cluster.

```
kubectl apply -f cilium-bgp-advertisement-ingress.yaml
```

7. Accédez au service à l'aide de l'adresse IP attribuée par Cilium LB IPAM.

```
curl -s http://LB_IP_ADDRESS:80/details/1 | jq
```

```
{
  "id": 1,
  "author": "William Shakespeare",
  "year": 1595,
  "type": "paperback",
  "pages": 200,
  "publisher": "PublisherA",
  "language": "English",
  "ISBN-10": "1234567890",
  "ISBN-13": "123-1234567890"
}
```

Type de service NodePort

Si votre environnement ne dispose pas d'une infrastructure d'équilibrage de charge et d'un contrôleur Ingress correspondant, ou si vous gérez vous-même votre infrastructure d'équilibrage de charge ou utilisez un équilibrage de charge basée sur DNS, vous pouvez configurer Cilium Ingress pour créer des services de type NodePort pour les ressources Ingress. Lorsque vous utilisez NodePort avec Cilium Ingress, le service de type NodePort est exposé sur un port de chaque nœud dans la plage de ports 30000-32767. Dans ce mode, lorsque le trafic atteint un nœud quelconque du cluster sur le NodePort, il est ensuite transféré vers un pod qui prend en charge le service, lequel peut se trouver sur le même nœud ou sur un nœud différent.

> **Note**
>
> La prise en charge de Cilium Gateway pour les services NodePort est prévue pour la version 1.18.x de Cilium (#27273)

Prérequis

- Cilium Ingress installé en suivant les instructions fournies dans the section called "Installer Cilium Ingress".
- Ressources Cilium Ingress avec exemple d'application déployée en suivant les instructions dans the section called "Déployer Cilium Ingress".

Procédure

1. Modifiez la ressource Ingress `my-ingress` existante pour la faire passer du type de service LoadBalancer à NodePort.

   ```
   kubectl patch ingress my-ingress --type=merge -p '{
       "metadata": {"annotations": {"ingress.cilium.io/service-type": "NodePort"}}
   }'
   ```

   Si vous n'avez pas créé la ressource Ingress, vous pouvez la créer en appliquant la définition Ingress suivante à votre cluster. Remarque : la définition d'Ingress ci-dessous utilise l'application exemple Istio Bookinfo décrite dans the section called "Déployer Cilium Ingress".

   ```
   apiVersion: networking.k8s.io/v1
   ```

```
kind: Ingress
metadata:
  name: my-ingress
  namespace: default
  annotations:
    service.beta.kubernetes.io/aws-load-balancer-type: "external"
    "ingress.cilium.io/service-type": "NodePort"
spec:
  ingressClassName: cilium
  rules:
  - http:
      paths:
      - backend:
          service:
            name: details
            port:
              number: 9080
        path: /details
        pathType: Prefix
```

2. Confirmez que le service pour la ressource Ingress a été mis à jour pour utiliser le type de service NodePort. Notez le port pour le protocole HTTP dans la sortie. Dans l'exemple ci-dessous, ce port HTTP est 32353, qui sera utilisé dans une étape ultérieure pour interroger le service. L'avantage d'utiliser Cilium Ingress avec un service de type NodePort est que vous pouvez appliquer un routage basé sur le chemin et l'hôte, ainsi que des politiques réseau pour le trafic Ingress, ce qui n'est pas possible avec un service standard de type NodePort sans Ingress.

```
kubectl -n default get svc cilium-ingress-my-ingress
```

```
NAME                         TYPE       CLUSTER-IP      EXTERNAL-IP   PORT(S)
             AGE
cilium-ingress-my-ingress   NodePort   172.16.47.153   <none>        80:32353/
TCP,443:30253/TCP   27m
```

3. Obtenez les adresses IP des nœuds de votre cluster.

```
kubectl get nodes -o wide
```

```
NAME                   STATUS   ROLES    AGE   VERSION               INTERNAL-IP
 EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION       CONTAINER-RUNTIME
```

```
mi-026d6a261e355fba7   Ready    <none>   23h   v1.32.3-eks-473151a   10.80.146.150
 <none>        Ubuntu 22.04.5 LTS   5.15.0-142-generic   containerd://1.7.27
mi-082f73826a163626e   Ready    <none>   23h   v1.32.3-eks-473151a   10.80.146.32
 <none>        Ubuntu 22.04.4 LTS   5.15.0-142-generic   containerd://1.7.27
mi-09183e8a3d755abf6   Ready    <none>   23h   v1.32.3-eks-473151a   10.80.146.33
 <none>        Ubuntu 22.04.4 LTS   5.15.0-142-generic   containerd://1.7.27
mi-0d78d815980ed202d   Ready    <none>   23h   v1.32.3-eks-473151a   10.80.146.97
 <none>        Ubuntu 22.04.4 LTS   5.15.0-142-generic   containerd://1.7.27
mi-0daa253999fe92daa   Ready    <none>   23h   v1.32.3-eks-473151a   10.80.146.100
 <none>        Ubuntu 22.04.4 LTS   5.15.0-142-generic   containerd://1.7.27
```

4. Accédez au service de type NodePort à l'aide des adresses IP de vos nœuds et du NodePort capturé ci-dessus. Dans l'exemple ci-dessous, l'adresse IP du nœud utilisée est `10.80.146.32` et le NodePort est `32353`. Remplacez-les par les valeurs correspondant à votre environnement.

```
curl -s http://10.80.146.32:32353/details/1 | jq
```

```
{
  "id": 1,
  "author": "William Shakespeare",
  "year": 1595,
  "type": "paperback",
  "pages": 200,
  "publisher": "PublisherA",
  "language": "English",
  "ISBN-10": "1234567890",
  "ISBN-13": "123-1234567890"
}
```

Réseau hôte

Comme pour le service de type NodePort, si vous ne disposez pas d'une infrastructure d'équilibrage de charge et d'un contrôleur Ingress ou Gateway, ou si vous gérez vous-même votre équilibrage de charge à l'aide d'un équilibreur de charge externe, vous pouvez configurer Cilium Ingress et Cilium Gateway pour exposer les ressources Ingress et Gateway directement sur le réseau hôte. Lorsque le mode réseau hôte est activé pour une ressource Ingress ou Gateway, les modes Service de type LoadBalancer et NodePort sont automatiquement désactivés. Le mode réseau hôte est mutuellement exclusif avec ces modes alternatifs pour chaque ressource Ingress ou Gateway. Par rapport au service de type NodePort, le mode réseau hôte offre une flexibilité supplémentaire pour

la plage de ports pouvant être utilisés (il n'est pas limité à la plage NodePort 30000-32767) et vous pouvez configurer un sous-ensemble de nœuds où les proxys Envoy s'exécutent sur le réseau hôte.

Prérequis

- Cilium Ingress ou Gateway installé conformément aux instructions fournies dans the section called "Installer Cilium Ingress" ou the section called "Installer Cilium Gateway".

Procédure

Passerelle

1. Créez un fichier nommé `cilium-gateway-host-network.yaml` avec le contenu suivant.

```
gatewayAPI:
  enabled: true
  hostNetwork:
    enabled: true
    # uncomment to restrict nodes where Envoy proxies run on the host network
    # nodes:
    #   matchLabels:
    #     role: gateway
```

2. Appliquez la configuration du réseau hôte Cilium Gateway à votre cluster.

```
helm upgrade cilium oci://public.ecr.aws/eks/cilium/cilium \
  --namespace kube-system \
  --reuse-values \
  --set operator.rollOutPods=true \
  -f cilium-gateway-host-network.yaml
```

Si vous n'avez pas créé la ressource Gateway, vous pouvez la créer en appliquant la définition Gateway suivante à votre cluster. La définition Gateway ci-dessous utilise l'application exemple Istio Bookinfo décrite dans the section called "Déployer Cilium Gateway". Dans l'exemple ci-dessous, la ressource Gateway est configurée pour utiliser le port 8111 pour l'écouteur HTTP, qui est le port d'écouteur partagé pour les proxys Envoy s'exécutant sur le réseau hôte. Si vous utilisez un port privilégié (inférieur à 1023) pour la ressource Gateway, consultez la documentation Cilium pour obtenir des instructions.

```
---
```

```
apiVersion: gateway.networking.k8s.io/v1
kind: Gateway
metadata:
  name: my-gateway
spec:
  gatewayClassName: cilium
  listeners:
  - protocol: HTTP
    port: 8111
    name: web-gw
    allowedRoutes:
      namespaces:
        from: Same
---
apiVersion: gateway.networking.k8s.io/v1
kind: HTTPRoute
metadata:
  name: http-app-1
spec:
  parentRefs:
  - name: my-gateway
    namespace: default
  rules:
  - matches:
    - path:
        type: PathPrefix
        value: /details
    backendRefs:
    - name: details
      port: 9080
```

Vous pouvez observer la configuration Cilium Envoy appliquée à l'aide de la commande suivante.

```
kubectl get cec cilium-gateway-my-gateway -o yaml
```

Vous pouvez obtenir le port d'écouteur Envoy pour le service `cilium-gateway-my-gateway` à l'aide de la commande suivante. Dans cet exemple, le port d'écouteur partagé est `8111`.

```
kubectl get cec cilium-gateway-my-gateway -o jsonpath={.spec.services[0].ports[0]}
```

3. Obtenez les adresses IP des nœuds de votre cluster.

```
kubectl get nodes -o wide
```

```
NAME                   STATUS   ROLES    AGE   VERSION               INTERNAL-IP
 EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION       CONTAINER-RUNTIME
mi-026d6a261e355fba7   Ready    <none>   23h   v1.32.3-eks-473151a   10.80.146.150
 <none>        Ubuntu 22.04.5 LTS   5.15.0-142-generic   containerd://1.7.27
mi-082f73826a163626e   Ready    <none>   23h   v1.32.3-eks-473151a   10.80.146.32
 <none>        Ubuntu 22.04.4 LTS   5.15.0-142-generic   containerd://1.7.27
mi-09183e8a3d755abf6   Ready    <none>   23h   v1.32.3-eks-473151a   10.80.146.33
 <none>        Ubuntu 22.04.4 LTS   5.15.0-142-generic   containerd://1.7.27
mi-0d78d815980ed202d   Ready    <none>   23h   v1.32.3-eks-473151a   10.80.146.97
 <none>        Ubuntu 22.04.4 LTS   5.15.0-142-generic   containerd://1.7.27
mi-0daa253999fe92daa   Ready    <none>   23h   v1.32.3-eks-473151a   10.80.146.100
 <none>        Ubuntu 22.04.4 LTS   5.15.0-142-generic   containerd://1.7.27
```

4. Accédez au service à l'aide des adresses IP de vos nœuds et du port d'écouteur de la ressource `cilium-gateway-my-gateway`. Dans l'exemple ci-dessous, l'adresse IP du nœud utilisée est `10.80.146.32` et le port d'écouteur est `8111`. Remplacez-les par les valeurs correspondant à votre environnement.

```
curl -s http://10.80.146.32:8111/details/1 | jq
```

```
{
  "id": 1,
  "author": "William Shakespeare",
  "year": 1595,
  "type": "paperback",
  "pages": 200,
  "publisher": "PublisherA",
  "language": "English",
  "ISBN-10": "1234567890",
  "ISBN-13": "123-1234567890"
}
```

Ingress

En raison d'un problème en amont avec Cilium (#34028), Cilium Ingress en mode réseau hôte nécessite l'utilisation de `loadbalancerMode: shared`, qui crée un seul service de type ClusterIP

pour toutes les ressources Ingress du cluster. Si vous utilisez un port privilégié (inférieur à 1023) pour la ressource Ingress, consultez la documentation Cilium pour obtenir des instructions.

1. Créez un fichier nommé `cilium-ingress-host-network.yaml` avec le contenu suivant.

```
ingressController:
  enabled: true
  loadbalancerMode: shared
  # This is a workaround for the upstream Cilium issue
  service:
    externalTrafficPolicy: null
    type: ClusterIP
  hostNetwork:
    enabled: true
    # ensure the port does not conflict with other services on the node
    sharedListenerPort: 8111
    # uncomment to restrict nodes where Envoy proxies run on the host network
    # nodes:
    #   matchLabels:
    #     role: ingress
```

2. Appliquez la configuration Cilium Ingress du réseau hôte à votre cluster.

```
helm upgrade cilium oci://public.ecr.aws/eks/cilium/cilium \
  --namespace kube-system \
  --reuse-values \
  --set operator.rollOutPods=true \
  -f cilium-ingress-host-network.yaml
```

Si vous n'avez pas créé la ressource Ingress, vous pouvez la créer en appliquant la définition Ingress suivante à votre cluster. La définition d'Ingress ci-dessous utilise l'application exemple Istio Bookinfo décrite dans the section called "Déployer Cilium Ingress".

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: my-ingress
  namespace: default
spec:
  ingressClassName: cilium
  rules:
  - http:
```

```
      paths:
      - backend:
          service:
            name: details
            port:
              number: 9080
        path: /details
        pathType: Prefix
```

Vous pouvez observer la configuration Cilium Envoy appliquée à l'aide de la commande suivante.

```
kubectl get cec -n kube-system cilium-ingress -o yaml
```

Vous pouvez obtenir le port d'écouteur Envoy pour le service `cilium-ingress` à l'aide de la commande suivante. Dans cet exemple, le port d'écouteur partagé est 8111.

```
kubectl get cec -n kube-system cilium-ingress -o
 jsonpath={.spec.services[0].ports[0]}
```

3. Obtenez les adresses IP des nœuds de votre cluster.

```
kubectl get nodes -o wide
```

```
NAME                   STATUS   ROLES    AGE   VERSION               INTERNAL-IP
 EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION       CONTAINER-RUNTIME
mi-026d6a261e355fba7   Ready    <none>   23h   v1.32.3-eks-473151a   10.80.146.150
 <none>        Ubuntu 22.04.5 LTS   5.15.0-142-generic   containerd://1.7.27
mi-082f73826a163626e   Ready    <none>   23h   v1.32.3-eks-473151a   10.80.146.32
 <none>        Ubuntu 22.04.4 LTS   5.15.0-142-generic   containerd://1.7.27
mi-09183e8a3d755abf6   Ready    <none>   23h   v1.32.3-eks-473151a   10.80.146.33
 <none>        Ubuntu 22.04.4 LTS   5.15.0-142-generic   containerd://1.7.27
mi-0d78d815980ed202d   Ready    <none>   23h   v1.32.3-eks-473151a   10.80.146.97
 <none>        Ubuntu 22.04.4 LTS   5.15.0-142-generic   containerd://1.7.27
mi-0daa253999fe92daa   Ready    <none>   23h   v1.32.3-eks-473151a   10.80.146.100
 <none>        Ubuntu 22.04.4 LTS   5.15.0-142-generic   containerd://1.7.27
```

4. Accédez au service à l'aide des adresses IP de vos nœuds et de `sharedListenerPort` pour la ressource `cilium-ingress`. Dans l'exemple ci-dessous, l'adresse IP du nœud utilisée est `10.80.146.32` et le port d'écouteur est 8111. Remplacez-les par les valeurs correspondant à votre environnement.

```
curl -s http://10.80.146.32:8111/details/1 | jq
```

```
{
  "id": 1,
  "author": "William Shakespeare",
  "year": 1595,
  "type": "paperback",
  "pages": 200,
  "publisher": "PublisherA",
  "language": "English",
  "ISBN-10": "1234567890",
  "ISBN-13": "123-1234567890"
}
```

Configurer les services de type LoadBalancer pour les nœuds hybrides

Cette rubrique décrit comment configurer l'équilibrage de charge de couche 4 (L4) pour les applications s'exécutant sur des nœuds hybrides Amazon EKS. Les services Kubernetes de type LoadBalancer sont utilisés pour exposer les applications Kubernetes à l'extérieur du cluster. Les services de type LoadBalancer sont couramment utilisés avec une infrastructure d'équilibrage de charge physique dans le cloud ou dans un environnement sur site pour traiter le trafic de la charge de travail. Cette infrastructure d'équilibrage de charge est généralement fournie avec un contrôleur spécifique à l'environnement.

AWS prend en charge Network Load Balancer (NLB) AWS et Cilium pour les services de type LoadBalancer s'exécutant sur des nœuds hybrides EKS. La décision d'utiliser NLB ou Cilium dépend de la source du trafic applicatif. Si le trafic applicatif provient d'une région AWS, AWS recommande d'utiliser AWS NLB ainsi que l'AWS Load Balancer Controller. Si le trafic applicatif provient de l'environnement local sur site ou périphérique, AWS recommande d'utiliser les capacités de répartition de charge intégrées à Cilium, qui peuvent être utilisées avec ou sans infrastructure d'équilibrage de charge dans votre environnement.

Pour l'équilibrage de charge du trafic des applications de couche 7 (L7), voir the section called "Configurer Ingress". Pour obtenir des informations générales sur l'équilibrage de charge avec EKS, consultez la section Meilleures pratiques pour l'équilibrage de charge.

AWS Network Load Balancer

Vous pouvez utiliser l'AWS Load Balancer Controller et NLB avec le type de cible `ip` pour les charges de travail s'exécutant sur des nœuds hybrides. Lorsque vous utilisez le type de cible `ip`, NLB transfère le trafic directement vers les pods, en contournant le chemin réseau de la couche Service. Pour que NLB atteigne les cibles IP des pods sur les nœuds hybrides, vos CIDR de pods sur site doivent être routables sur votre réseau sur site. De plus, l'AWS Load Balancer Controller utilise des webhooks et nécessite une communication directe depuis le plan de contrôle EKS. Pour de plus amples informations, consultez the section called "Configuration de webhooks".

- Consultez the section called "Répartition de charge réseau" pour connaître les exigences de configuration du sous-réseau, et the section called "Installer avec Helm" et les Bonnes pratiques en matière d'équilibrage de charge pour obtenir des informations supplémentaires sur le Network Load Balancer AWS et l'AWS Load Balancer Controller.
- Consultez les configurations NLB de l'AWS Load Balancer Controller pour connaître les configurations pouvant être appliquées aux services de type `LoadBalancer` avec le Network Load Balancer AWS.

Prérequis

- Cilium installé en suivant les instructions dans the section called "Configurer CNI".
- Le plan de contrôle Cilium BGP a été activé en suivant les instructions fournies dans the section called "Configuration de BGP". Si vous ne souhaitez pas utiliser BGP, vous devez utiliser une autre méthode pour rendre vos CIDR de pods sur site routables sur votre réseau sur site. Pour plus d'informations, consultez the section called "CIDR de pods distants routables".
- Helm installé dans votre environnement de ligne de commande, consultez les Instructions de configuration de Helm.
- eksctl installé dans votre environnement de ligne de commande, consultez les instructions de configuration d'eksctl.

Procédure

1. Téléchargez une politique IAM pour le contrôleur de l'équilibreur de charge AWS qui lui permet d'effectuer des appels aux API AWS en votre nom.

```
curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-
controller/refs/heads/main/docs/install/iam_policy.json
```

2. Créez une politique IAM à l'aide de la politique téléchargée à l'étape précédente.

```
aws iam create-policy \
    --policy-name AWSLoadBalancerControllerIAMPolicy \
    --policy-document file://iam_policy.json
```

3. Remplacez les valeurs pour le nom du cluster (`CLUSTER_NAME`), la région AWS (`AWS_REGION`) et l'ID du compte AWS (`AWS_ACCOUNT_ID`) par vos paramètres, puis exécutez la commande suivante.

```
eksctl create iamserviceaccount \
    --cluster=CLUSTER_NAME \
    --namespace=kube-system \
    --name=aws-load-balancer-controller \
    --attach-policy-arn=arn:aws:iam::AWS_ACCOUNT_ID:policy/
AWSLoadBalancerControllerIAMPolicy \
    --override-existing-serviceaccounts \
    --region AWS_REGION \
    --approve
```

4. Ajoutez le référentiel des Charts de Helm eks-charts. AWS maintient ce référentiel sur GitHub.

```
helm repo add eks https://aws.github.io/eks-charts
```

5. Mettez à jour votre référentiel Helm local pour vous assurer que vous disposez des graphiques les plus récents.

```
helm repo update eks
```

6. Installez le contrôleur de l'équilibreur de charge AWS. Remplacez les valeurs pour le nom du cluster (`CLUSTER_NAME`), la région AWS (`AWS_REGION`), l'ID VPC (`VPC_ID`) et la version des Charts de Helm de l'AWS Load Balancer Controller (`AWS_LBC_HELM_VERSION`) par vos paramètres. Vous pouvez trouver la dernière version des Charts de Helm en exécutant `helm search repo eks/aws-load-balancer-controller --versions`. Si vous utilisez un cluster en mode mixte avec des nœuds hybrides et des nœuds dans le cloud AWS, vous pouvez exécuter l'AWS Load Balancer Controller sur les nœuds cloud en suivant les instructions disponibles à l'adresse the section called "Contrôleur de l'équilibreur de charge AWS".

```
helm install aws-load-balancer-controller eks/aws-load-balancer-controller \
  -n kube-system \
  --version AWS_LBC_HELM_VERSION \
  --set clusterName=CLUSTER_NAME \
  --set region=AWS_REGION \
  --set vpcId=VPC_ID \
  --set serviceAccount.create=false \
  --set serviceAccount.name=aws-load-balancer-controller
```

7. Vérifiez que l'AWS Load Balancer Controller a été installé correctement.

```
kubectl get -n kube-system deployment aws-load-balancer-controller
```

```
NAME                           READY   UP-TO-DATE   AVAILABLE   AGE
aws-load-balancer-controller   2/2     2            2           84s
```

8. Définissez un exemple d'application dans un fichier nommé `tcp-sample-app.yaml`. L'exemple ci-dessous utilise un déploiement NGINX simple avec un port TCP.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: tcp-sample-app
  namespace: default
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: public.ecr.aws/nginx/nginx:1.23
          ports:
            - name: tcp
              containerPort: 80
```

9. Appliquez le déploiement à votre cluster.

```
kubectl apply -f tcp-sample-app.yaml
```

10. Définissez un service de type LoadBalancer pour le déploiement dans un fichier nommé `tcp-sample-service.yaml`.

```
apiVersion: v1
kind: Service
metadata:
  name: tcp-sample-service
  namespace: default
  annotations:
    service.beta.kubernetes.io/aws-load-balancer-type: external
    service.beta.kubernetes.io/aws-load-balancer-nlb-target-type: ip
    service.beta.kubernetes.io/aws-load-balancer-scheme: internet-facing
spec:
  ports:
    - port: 80
      targetPort: 80
      protocol: TCP
  type: LoadBalancer
  selector:
    app: nginx
```

11. Appliquez la configuration du service à votre cluster.

```
kubectl apply -f tcp-sample-service.yaml
```

12. La configuration du NLB pour le service peut prendre quelques minutes. Une fois le NLB provisionné, le service se verra attribuer une adresse correspondant au nom DNS du déploiement NLB.

```
kubectl get svc tcp-sample-service
```

```
NAME                TYPE           CLUSTER-IP       EXTERNAL-IP
                                                PORT(S)        AGE
tcp-sample-service   LoadBalancer   172.16.115.212   k8s-default-tcpsampl-xxxxxxxxxx-
xxxxxxxxxxxxxxxx.elb.<region>.amazonaws.com   80:30396/TCP   8s
```

13. Accédez au service à l'aide de l'adresse du NLB.

```
curl k8s-default-tcpsampl-xxxxxxxxxx-xxxxxxxxxxxxxxxx.elb.<region>.amazonaws.com
```

Un exemple de résultat est présenté ci-dessous.

```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
[...]
```

14 Nettoyez les ressources que vous avez créées.

```
kubectl delete -f tcp-sample-service.yaml
kubectl delete -f tcp-sample-app.yaml
```

Équilibrage de charge dans le cluster Cilium

Cilium peut être utilisé comme équilibreur de charge intra-cluster pour les charges de travail exécutées sur des nœuds hybrides EKS, ce qui peut s'avérer utile pour les environnements qui ne disposent pas d'infrastructure d'équilibrage de charge. Les capacités d'équilibrage de charge de Cilium reposent sur une combinaison de fonctionnalités Cilium, notamment le remplacement de kube-proxy, la gestion des adresses IP de l'équilibreur de charge (IPAM) et le plan de contrôle BGP. Les responsabilités de ces fonctionnalités sont détaillées ci-dessous :

- Remplacement de Cilium kube-proxy : gère le routage du trafic des services vers les pods dorsaux.
- Cilium Load Balancer IPAM : gère les adresses IP pouvant être attribuées aux services de type `LoadBalancer`.
- Plan de contrôle Cilium BGP : annonce les adresses IP attribuées par le Load Balancer IPAM au réseau local.

Si vous n'utilisez pas le remplacement kube-proxy de Cilium, vous pouvez toujours utiliser Cilium Load Balancer IPAM et BGP Control Plane pour allouer et attribuer des adresses IP aux services de type LoadBalancer. Si vous n'utilisez pas le remplacement kube-proxy de Cilium, l'équilibrage de charge pour les services vers les pods dorsaux est géré par kube-proxy et les règles iptables par défaut dans EKS.

Prérequis

- Cilium installé en suivant les instructions dans the section called "Configurer CNI", avec ou sans remplacement de kube-proxy activé. Le remplacement de kube-proxy par Cilium nécessite l'exécution d'un système d'exploitation avec un noyau Linux au moins aussi récent que v4.19.57, v5.1.16 ou v5.2.0. Toutes les versions récentes des systèmes d'exploitation pris en charge pour une utilisation avec des nœuds hybrides répondent à ces critères, à l'exception de Red Hat Enterprise Linux (RHEL) 8.x.
- Le plan de contrôle Cilium BGP a été activé en suivant les instructions fournies dans the section called "Configuration de BGP". Si vous ne souhaitez pas utiliser BGP, vous devez utiliser une autre méthode pour rendre vos CIDR de pods sur site routables sur votre réseau sur site. Pour plus d'informations, consultez the section called "CIDR de pods distants routables".
- Helm installé dans votre environnement de ligne de commande, consultez les Instructions de configuration de Helm.

Procédure

1. Créez un fichier nommé `cilium-lbip-pool-loadbalancer.yaml` avec une ressource `CiliumLoadBalancerIPPool` pour configurer la plage d'adresses IP du Load Balancer pour vos services de type LoadBalancer.
   - Remplacez `LB_IP_CIDR` par la plage d'adresses IP à utiliser pour les adresses IP de l'équilibreur de charge. Pour sélectionner une seule adresse IP, utilisez un CIDR `/32`. Pour plus d'informations, consultez la section Gestion des adresses IP LoadBalancer dans la documentation Cilium.
   - Le champ `serviceSelector` est configuré pour correspondre au nom du service que vous créerez à l'étape suivante. Avec cette configuration, les adresses IP de ce pool ne seront attribuées qu'aux services portant le nom `tcp-sample-service`.

```
apiVersion: cilium.io/v2alpha1
kind: CiliumLoadBalancerIPPool
metadata:
  name: tcp-service-pool
spec:
  blocks:
  - cidr: "LB_IP_CIDR"
  serviceSelector:
    matchLabels:
```

```
      io.kubernetes.service.name: tcp-sample-service
```

2. Appliquez la ressource `CiliumLoadBalancerIPPool` à votre cluster.

```
kubectl apply -f cilium-lbip-pool-loadbalancer.yaml
```

3. Vérifiez qu'au moins une adresse IP est disponible dans le pool.

```
kubectl get ciliumloadbalancerippools.cilium.io
```

```
NAME               DISABLED   CONFLICTING   IPS AVAILABLE   AGE
tcp-service-pool   false      False         1               24m
```

4. Créez un fichier nommé `cilium-bgp-advertisement-loadbalancer.yaml` avec une ressource `CiliumBGPAdvertisement` pour annoncer l'adresse IP de l'équilibreur de charge pour le service que vous allez créer à l'étape suivante. Si vous n'utilisez pas Cilium BGP, vous pouvez ignorer cette étape. L'adresse IP de l'équilibreur de charge utilisée pour votre service doit être routable sur votre réseau sur site afin que vous puissiez interroger le service lors de la dernière étape.
   - Le champ `advertisementType` est défini sur `Service` et `service.addresses` est défini sur `LoadBalancerIP` pour annoncer uniquement le `LoadBalancerIP` pour les services de type `LoadBalancer`.
   - Le champ `selector` est configuré pour correspondre au nom du service que vous créerez à l'étape suivante. Avec cette configuration, seul `LoadBalancerIP` pour les services portant le nom `tcp-sample-service` seront annoncés.

```
apiVersion: cilium.io/v2alpha1
kind: CiliumBGPAdvertisement
metadata:
  name: bgp-advertisement-tcp-service
  labels:
    advertise: bgp
spec:
  advertisements:
    - advertisementType: "Service"
      service:
        addresses:
          - LoadBalancerIP
      selector:
        matchLabels:
```

```
            io.kubernetes.service.name: tcp-sample-service
```

5. Appliquez la ressource `CiliumBGPAdvertisement` à votre cluster. Si vous n'utilisez pas Cilium BGP, vous pouvez ignorer cette étape.

```
kubectl apply -f cilium-bgp-advertisement-loadbalancer.yaml
```

6. Définissez un exemple d'application dans un fichier nommé `tcp-sample-app.yaml`. L'exemple ci-dessous utilise un déploiement NGINX simple avec un port TCP.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: tcp-sample-app
  namespace: default
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: public.ecr.aws/nginx/nginx:1.23
          ports:
            - name: tcp
              containerPort: 80
```

7. Appliquez le déploiement à votre cluster.

```
kubectl apply -f tcp-sample-app.yaml
```

8. Définissez un service de type LoadBalancer pour le déploiement dans un fichier nommé `tcp-sample-service.yaml`.
   - Vous pouvez demander une adresse IP spécifique à partir du pool d'adresses IP de l'équilibreur de charge à l'aide de l'annotation `lbipam.cilium.io/ips` sur l'objet Service. Vous pouvez supprimer cette annotation si vous ne souhaitez pas demander une adresse IP spécifique pour le Service.

- Le champ spec `loadBalancerClass` est obligatoire afin d'empêcher l'ancien fournisseur de cloud AWS de créer un Classic Load Balancer pour le service. Dans l'exemple ci-dessous, cela est configuré sur `io.cilium/bgp-control-plane` pour utiliser le plan de contrôle BGP de Cilium comme classe d'équilibrage de charge. Ce champ peut également être configuré sur `io.cilium/l2-announcer` pour utiliser la fonctionnalité L2 Announcements de Cilium (actuellement en version bêta et non officiellement prise en charge par AWS).

```
apiVersion: v1
kind: Service
metadata:
  name: tcp-sample-service
  namespace: default
  annotations:
    lbipam.cilium.io/ips: "LB_IP_ADDRESS"
spec:
  loadBalancerClass: io.cilium/bgp-control-plane
  ports:
    - port: 80
      targetPort: 80
      protocol: TCP
  type: LoadBalancer
  selector:
    app: nginx
```

9. Appliquez le service à votre cluster. Le service sera créé avec une adresse IP externe que vous pourrez utiliser pour accéder à l'application.

```
kubectl apply -f tcp-sample-service.yaml
```

10. Vérifiez que le service a été créé avec succès et qu'une adresse IP lui a été attribuée à partir du `CiliumLoadBalancerIPPool` créé à l'étape précédente.

```
kubectl get svc tcp-sample-service
```

```
NAME                TYPE           CLUSTER-IP      EXTERNAL-IP     PORT(S)
 AGE
tcp-sample-service   LoadBalancer   172.16.117.76   LB_IP_ADDRESS   80:31129/TCP
 14m
```

11. Si vous utilisez Cilium en mode de remplacement de kube-proxy, vous pouvez vérifier que Cilium gère l'équilibrage de charge pour le service en exécutant la commande suivante. Dans la sortie ci-dessous, les adresses `10.86.2.x` sont les adresses IP des pods dorsaux du service.

```
kubectl -n kube-system exec ds/cilium -- cilium-dbg service list
```

```
ID   Frontend              Service Type   Backend
...
41   LB_IP_ADDRESS:80/TCP   LoadBalancer   1 => 10.86.2.76:80/TCP (active)
                                           2 => 10.86.2.130:80/TCP (active)
                                           3 => 10.86.2.141:80/TCP (active)
```

12. Confirmez que Cilium annonce l'adresse IP au réseau local via BGP. Dans l'exemple ci-dessous, il existe cinq nœuds hybrides, chacun annonçant le `LB_IP_ADDRESS` pour le service `tcp-sample-service` au réseau local.

```
Node                    VRouter      Prefix             NextHop   Age     Attrs
mi-026d6a261e355fba7    NODES_ASN
                   LB_IP_ADDRESS/32   0.0.0.0   12m3s   [{Origin: i} {Nexthop:
 0.0.0.0}]
mi-082f73826a163626e    NODES_ASN
                   LB_IP_ADDRESS/32   0.0.0.0   12m3s   [{Origin: i} {Nexthop:
 0.0.0.0}]
mi-09183e8a3d755abf6    NODES_ASN
                   LB_IP_ADDRESS/32   0.0.0.0   12m3s   [{Origin: i} {Nexthop:
 0.0.0.0}]
mi-0d78d815980ed202d    NODES_ASN
                   LB_IP_ADDRESS/32   0.0.0.0   12m3s   [{Origin: i} {Nexthop:
 0.0.0.0}]
mi-0daa253999fe92daa    NODES_ASN
                   LB_IP_ADDRESS/32   0.0.0.0   12m3s   [{Origin: i} {Nexthop:
 0.0.0.0}]
```

13. Accédez au service à l'aide de l'adresse IP de l'équilibreur de charge attribué.

```
curl LB_IP_ADDRESS
```

Un exemple de résultat est présenté ci-dessous.

```
<!DOCTYPE html>
```

```
<html>
<head>
<title>Welcome to nginx!</title>
[...]
```

14 Nettoyez les ressources que vous avez créées.

```
kubectl delete -f tcp-sample-service.yaml
kubectl delete -f tcp-sample-app.yaml
kubectl delete -f cilium-lb-ip-pool.yaml
kubectl delete -f cilium-bgp-advertisement.yaml
```

Configurer les politiques réseau Kubernetes pour les nœuds hybrides

AWS prend en charge les politiques réseau Kubernetes (couche 3 / couche 4) pour le trafic entrant et sortant des pods lors de l'utilisation de Cilium comme CNI avec les nœuds hybrides EKS. Si vous exécutez des clusters EKS avec des nœuds dans le cloud AWS, AWS prend en charge Amazon VPC CNI pour les politiques réseau Kubernetes.

Cette rubrique explique comment configurer Cilium et les politiques réseau Kubernetes avec les nœuds hybrides EKS. Pour plus d'informations sur les politiques réseau Kubernetes, consultez la section Politiques réseau Kubernetes dans la documentation Kubernetes.

Configurer des stratégies réseau

Considérations

- AWS prend en charge les politiques réseau Kubernetes en amont et les spécifications pour les pods ingress et egress. AWS ne prend actuellement pas en charge ni `CiliumNetworkPolicy` ni `CiliumClusterwideNetworkPolicy`.
- La valeur Helm `policyEnforcementMode` peut être utilisée pour contrôler le comportement par défaut de l'application des politiques Cilium. Le comportement par défaut autorise tout le trafic sortant et entrant. Lorsqu'un point de terminaison est sélectionné par une stratégie réseau, il passe à un état de refus par défaut, dans lequel seul le trafic explicitement autorisé est autorisé. Consultez la documentation Cilium pour plus d'informations sur le mode de politique par défaut et les modes d'application des politiques.
- Si vous modifiez `policyEnforcementMode` pour une installation Cilium existante, vous devez redémarrer le DaemonSet de l'agent Cilium pour appliquer le nouveau mode d'application des politiques.

- Utilisez `namespaceSelector` et `podSelector` pour autoriser ou refuser le trafic vers/depuis les espaces de noms et les pods avec des étiquettes correspondantes. Le `namespaceSelector` et `podSelector` peut être utilisé avec `matchLabels` ou `matchExpressions` pour choisir des espaces de noms et des pods en fonction de leurs étiquettes.
- Utilisez `ingress.ports` et `egress.ports` pour autoriser ou refuser le trafic vers/depuis les ports ainsi que les protocoles.
- Le champ `ipBlock` ne peut pas être utilisé pour autoriser ou refuser de façon sélective le trafic vers/depuis les adresses IP des pods (#9209). Utiliser des sélecteurs `ipBlock` pour les adresses IP des nœuds est une fonctionnalité bêta de Cilium qui n'est pas prise en charge par AWS.
- Consultez la ressource NetworkPolicy dans la documentation Kubernetes pour obtenir des informations sur les champs disponibles pour les politiques réseau Kubernetes.

Prérequis

- Cilium installé en suivant les instructions dans the section called "Configurer CNI".
- Helm installé dans votre environnement de ligne de commande, consultez les Instructions de configuration de Helm.

Procédure

La procédure suivante configure les stratégies réseau pour un exemple d'application de microservices afin que les composants ne puissent communiquer qu'avec les autres composants nécessaires au fonctionnement de l'application. La procédure utilise l'application microservices Istio Bookinfo.

L'application Bookinfo se compose de quatre microservices distincts ayant les relations suivantes :

- pageproduit. Le microservice de la page produit appelle les microservices de détails et d'avis pour remplir la page.
- détails. Le microservice details contient des informations sur les livres.
- reviews. Le microservice « critiques » contient des critiques de livres. Il fait également appel au microservice de notation.
- évaluations. Le microservice « évaluations » contient des informations sur le classement des livres qui accompagnent les critiques littéraires.

1. Créer l'application exemple.

```
kubectl apply -f https://raw.githubusercontent.com/istio/istio/refs/heads/master/
samples/bookinfo/platform/kube/bookinfo.yaml
```

2. Vérifiez que l'application fonctionne correctement et notez l'adresse IP du pod pour le microservice pageproduit. Vous utiliserez cette adresse IP du pod pour interroger chaque microservice dans les étapes suivantes.

```
kubectl get pods -o wide
```

```
NAME                              READY   STATUS    RESTARTS   AGE   IP
 NODE
details-v1-766844796b-9wff2       1/1     Running   0          7s    10.86.3.7
 mi-0daa253999fe92daa
productpage-v1-54bb874995-lwfgg   1/1     Running   0          7s    10.86.2.193
 mi-082f73826a163626e
ratings-v1-5dc79b6bcd-59njm       1/1     Running   0          7s    10.86.2.232
 mi-082f73826a163626e
reviews-v1-598b896c9d-p2289       1/1     Running   0          7s    10.86.2.47
 mi-026d6a261e355fba7
reviews-v2-556d6457d-djktc        1/1     Running   0          7s    10.86.3.58
 mi-0daa253999fe92daa
reviews-v3-564544b4d6-g8hh4       1/1     Running   0          7s    10.86.2.69
 mi-09183e8a3d755abf6
```

3. Créez un pod qui sera utilisé tout au long du processus pour tester les politiques réseau. Notez que le pod est créé dans l'espace de noms `default` avec le label `access: true`.

```
kubectl run curl-pod --image=curlimages/curl -i --tty --labels=access=true --
namespace=default --overrides='{"spec": { "nodeSelector": {"eks.amazonaws.com/
compute-type": "hybrid"}}}' -- /bin/sh
```

4. Testez l'accès au microservice de la page produit. Dans l'exemple ci-dessous, nous utilisons l'adresse IP du pod productpage (`10.86.2.193`) pour interroger le microservice. Remplacez ceci par l'adresse IP du pod productpage dans votre environnement.

```
curl -s http://10.86.2.193:9080/productpage | grep -o "<title>.*</title>"
```

```
<title>Simple Bookstore App</title>
```

5. Vous pouvez quitter le pod test curl en tapant `exit` et vous pouvez vous reconnecter au pod en exécutant la commande suivante.

```
kubectl attach curl-pod -c curl-pod -i -t
```

6. Pour démontrer les effets des stratégies réseau dans les étapes suivantes, nous créons d'abord une stratégie réseau qui refuse tout trafic pour les microservices BookInfo. Créez un fichier nommé `network-policy-deny-bookinfo.yaml` qui définit la stratégie réseau de refus.

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: deny-bookinfo
  namespace: default
spec:
  podSelector:
    matchExpressions:
    - key: app
      operator: In
      values: ["productpage", "details", "reviews", "ratings"]
  policyTypes:
  - Ingress
  - Egress
```

7. Appliquez la stratégie réseau de refus à votre cluster.

```
kubectl apply -f network-policy-default-deny-bookinfo.yaml
```

8. Testez l'accès à l'application BookInfo. Dans l'exemple ci-dessous, nous utilisons l'adresse IP du pod productpage (`10.86.2.193`) pour interroger le microservice. Remplacez ceci par l'adresse IP du pod productpage dans votre environnement.

```
curl http://10.86.2.193:9080/productpage --max-time 10
```

```
curl: (28) Connection timed out after 10001 milliseconds
```

9. Créez un fichier nommé `network-policy-productpage.yaml` qui définit la politique réseau de la page produit. La politique prévoit les règles suivantes :
   - autorise le trafic entrant provenant des pods portant le label `access: true` (le pod curl créé à l'étape précédente)

- autorise le trafic TCP sortant sur le port `9080` pour les microservices détaillant les informations, les avis et les évaluations
- autorise le trafic TCP/UDP sortant sur le port `53` pour CoreDNS qui s'exécute dans l'espace de noms `kube-system`

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: productpage-policy
  namespace: default
spec:
  podSelector:
    matchLabels:
      app: productpage
  policyTypes:
  - Ingress
  - Egress
  ingress:
  - from:
    - podSelector:
        matchLabels:
          access: "true"
  egress:
  - to:
    - podSelector:
        matchExpressions:
        - key: app
          operator: In
          values: ["details", "reviews", "ratings"]
    ports:
    - port: 9080
      protocol: TCP
  - to:
    - namespaceSelector:
        matchLabels:
          kubernetes.io/metadata.name: kube-system
      podSelector:
        matchLabels:
          k8s-app: kube-dns
    ports:
    - port: 53
      protocol: UDP
```

```
    - port: 53
      protocol: TCP
```

10. Appliquez la stratégie réseau de la page produit à votre cluster.

```
kubectl apply -f network-policy-productpage.yaml
```

11. Connectez-vous au pod curl et testez l'accès à l'application Bookinfo. L'accès au microservice de la page produit est désormais autorisé, mais les autres microservices sont toujours refusés, car ils sont toujours soumis à la politique réseau de refus. Dans les exemples ci-dessous, nous utilisons l'adresse IP du pod productpage (`10.86.2.193`) pour interroger le microservice. Remplacez ceci par l'adresse IP du pod productpage dans votre environnement.

```
kubectl attach curl-pod -c curl-pod -i -t
```

```
curl -s http://10.86.2.193:9080/productpage | grep -o "<title>.*</title>"
<title>Simple Bookstore App</title>
```

```
curl -s http://10.86.2.193:9080/api/v1/products/1
{"error": "Sorry, product details are currently unavailable for this book."}
```

```
curl -s http://10.86.2.193:9080/api/v1/products/1/reviews
{"error": "Sorry, product reviews are currently unavailable for this book."}
```

```
curl -s http://10.86.2.193:9080/api/v1/products/1/ratings
{"error": "Sorry, product ratings are currently unavailable for this book."}
```

12. Créez un fichier nommé `network-policy-details.yaml` qui définit les détails de la stratégie réseau. La politique autorise uniquement le trafic entrant provenant du microservice productpage.

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: details-policy
  namespace: default
spec:
  podSelector:
    matchLabels:
```

```
      app: details
  policyTypes:
  - Ingress
  ingress:
  - from:
    - podSelector:
        matchLabels:
          app: productpage
```

13. Créez un fichier nommé `network-policy-reviews.yaml` qui définit la politique réseau des avis. La politique autorise uniquement le trafic entrant provenant du microservice productpage et uniquement le trafic sortant vers le microservice ratings et CoreDNS.

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: reviews-policy
  namespace: default
spec:
  podSelector:
    matchLabels:
      app: reviews
  policyTypes:
  - Ingress
  - Egress
  ingress:
  - from:
    - podSelector:
        matchLabels:
          app: productpage
  egress:
  - to:
    - podSelector:
        matchLabels:
          app: ratings
  - to:
    - namespaceSelector:
        matchLabels:
          kubernetes.io/metadata.name: kube-system
      podSelector:
        matchLabels:
          k8s-app: kube-dns
    ports:
```

```
    - port: 53
      protocol: UDP
    - port: 53
      protocol: TCP
```

14. Créez un fichier nommé `network-policy-ratings.yaml` qui définit la politique du réseau de notation. La politique autorise uniquement le trafic entrant provenant des microservices de la page produit et des avis.

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: ratings-policy
  namespace: default
spec:
  podSelector:
    matchLabels:
      app: ratings
  policyTypes:
  - Ingress
  ingress:
  - from:
    - podSelector:
        matchExpressions:
        - key: app
          operator: In
          values: ["productpage", "reviews"]
```

15. Appliquez les politiques réseau relatives aux détails, aux avis et aux évaluations à votre cluster.

```
kubectl apply -f network-policy-details.yaml
kubectl apply -f network-policy-reviews.yaml
kubectl apply -f network-policy-ratings.yaml
```

16. Connectez-vous au pod curl et testez l'accès à l'application Bookinfo. Dans les exemples ci-dessous, nous utilisons l'adresse IP du pod productpage (`10.86.2.193`) pour interroger le microservice. Remplacez ceci par l'adresse IP du pod productpage dans votre environnement.

```
kubectl attach curl-pod -c curl-pod -i -t
```

Testez les détails du microservice.

```
curl -s http://10.86.2.193:9080/api/v1/products/1
```

```
{"id": 1, "author": "William Shakespeare", "year": 1595, "type": "paperback",
 "pages": 200, "publisher": "PublisherA", "language": "English", "ISBN-10":
 "1234567890", "ISBN-13": "123-1234567890"}
```

Testez le microservice des évaluations.

```
curl -s http://10.86.2.193:9080/api/v1/products/1/reviews
```

```
{"id": "1", "podname": "reviews-v1-598b896c9d-p2289", "clustername": "null",
 "reviews": [{"reviewer": "Reviewer1", "text": "An extremely entertaining play
 by Shakespeare. The slapstick humour is refreshing!"}, {"reviewer": "Reviewer2",
 "text": "Absolutely fun and entertaining. The play lacks thematic depth when
 compared to other plays by Shakespeare."}]}
```

Testez le microservice de notation.

```
curl -s http://10.86.2.193:9080/api/v1/products/1/ratings
```

```
{"id": 1, "ratings": {"Reviewer1": 5, "Reviewer2": 4}}
```

17 Nettoyez les ressources que vous avez créées au cours de cette procédure.

```
kubectl delete -f network-policy-deny-bookinfo.yaml
kubectl delete -f network-policy-productpage.yaml
kubectl delete -f network-policy-details.yaml
kubectl delete -f network-policy-reviews.yaml
kubectl delete -f network-policy-ratings.yaml
kubectl delete -f https://raw.githubusercontent.com/istio/istio/refs/heads/master/
samples/bookinfo/platform/kube/bookinfo.yaml
kubectl delete pod curl-pod
```

Concepts pour les nœuds hybrides

Avec les nœuds hybrides Amazon EKS, vous pouvez connecter des machines physiques ou virtuelles fonctionnant dans des environnements sur site ou en périphérie à des clusters Amazon

EKS fonctionnant dans le cloud AWS. Cette approche présente de nombreux avantages, mais introduit également de nouveaux concepts et architectures de réseau pour ceux qui sont habitués à exploiter des clusters Kubernetes dans un environnement réseau unique.

Les sections suivantes approfondissent les concepts Kubernetes et réseau pour les nœuds hybrides EKS et détaillent la manière dont le trafic circule à travers l'architecture hybride. Ces sections nécessitent que vous ayez des connaissances de base sur les réseaux Kubernetes, telles que les concepts de pods, de nœuds, de services, du plan de contrôle Kubernetes, de kubelet et de kube-proxy.

Nous vous recommandons de lire ces pages dans l'ordre, en commençant par la the section called "Concepts de réseau", puis la the section called "Concepts Kubernetes", et enfin la the section called "Flux de trafic".

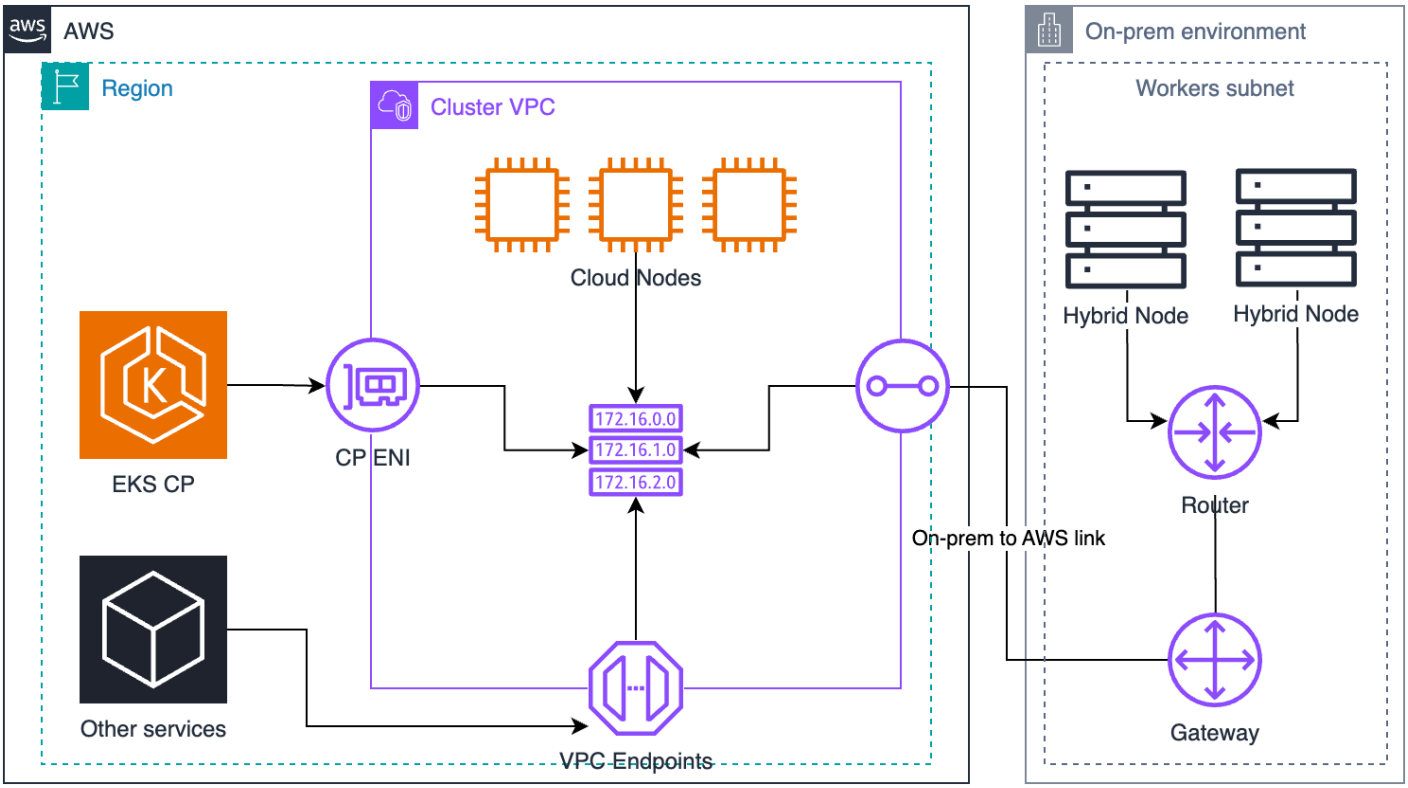
Rubriques

- Concepts de mise en réseau pour les nœuds hybrides
- Concepts Kubernetes pour les nœuds hybrides
- Flux de trafic réseau pour les nœuds hybrides

Concepts de mise en réseau pour les nœuds hybrides

Cette section détaille les concepts fondamentaux du réseau et les contraintes à prendre en compte lors de la conception de la topologie réseau pour les nœuds hybrides EKS.

Concepts de mise en réseau pour les nœuds hybrides EKS



Le VPC en tant que hub réseau

Tout le trafic qui traverse les limites du cloud passe par votre VPC. Cela inclut le trafic entre le plan de contrôle EKS ou les pods s'exécutant AWS vers les nœuds hybrides ou les pods exécutés sur ceux-ci. Vous pouvez considérer le VPC de votre cluster comme le hub réseau entre vos nœuds hybrides et le reste du cluster. Cette architecture vous donne un contrôle total sur le trafic et son routage, mais vous rend également responsable de la configuration correcte des routes, des groupes de sécurité et des pare-feu pour le VPC.

Plan de contrôle EKS vers le VPC

Le plan de contrôle EKS attache les interfaces réseau élastiques (ENIs) à votre VPC. Ils ENIs gèrent le trafic à destination et en provenance du serveur API EKS. Vous contrôlez le placement du plan de contrôle EKS ENIs lorsque vous configurez votre cluster, car EKS s'attache ENIs aux sous-réseaux que vous transmettez lors de la création du cluster.

EKS associe les groupes de sécurité aux groupes ENIs qu'EKS attache à vos sous-réseaux. Ces groupes de sécurité autorisent le trafic à destination et en provenance du plan de contrôle EKS via le

ENIs. Ceci est important pour les nœuds hybrides EKS, car vous devez autoriser le trafic provenant des nœuds hybrides et des pods qui s'y exécutent vers le plan de contrôle EKS ENIs.

Réseau de nœuds distants

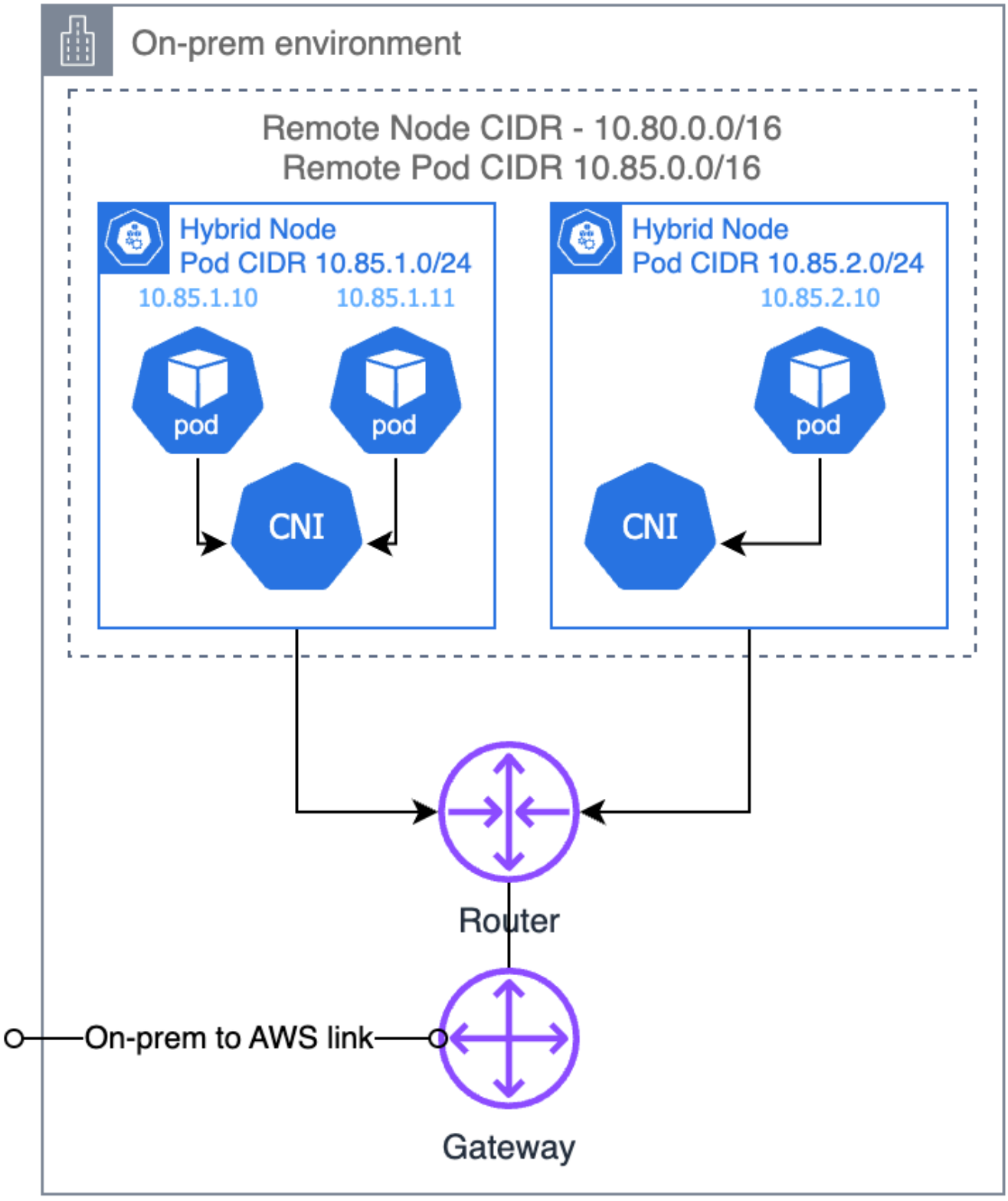
Les réseaux de nœuds distants, en particulier le nœud distant CIDRs, sont les plages de nœuds IPs attribuées aux machines que vous utilisez en tant que nœuds hybrides. Lorsque vous provisionnez des nœuds hybrides, ceux-ci résident dans votre centre de données sur site ou à la périphérie, qui est un domaine réseau différent du plan de contrôle EKS et du VPC. Chaque nœud hybride possède une ou plusieurs adresses IP provenant d’un CIDR de nœud distant distinct des sous-réseaux de votre VPC.

Vous configurez le cluster EKS avec ces nœuds distants CIDRs afin qu'EKS sache qu'il doit acheminer tout le trafic destiné aux nœuds hybrides IPs via le VPC de votre cluster, comme les demandes adressées à l'API kubelet. Les connexions à l'`kubelet`API sont utilisées dans les `kubectl port-forward` commandes `kubectl attach` `kubectl cp`kubectl exec,`kubectl logs`,, et.

Réseaux de pods distants

Les réseaux de pods distants sont les plages IPs attribuées aux pods exécutés sur les nœuds hybrides. En général, vous configurez votre CNI avec ces plages et la fonctionnalité de gestion des adresses IP (IPAM) du CNI se charge d’attribuer une tranche de ces plages à chaque nœud hybride. Lorsque vous créez un pod, le CNI attribue une adresse IP au pod à partir de la tranche allouée au nœud où le pod a été planifié.

Vous configurez le cluster EKS avec ces pods distants CIDRs afin que le plan de contrôle EKS sache qu'il doit acheminer tout le trafic destiné aux pods exécutés sur les nœuds hybrides via le VPC de votre cluster, par exemple pour les communications avec les webhooks.



Sur site, sur le VPC

Le réseau local que vous utilisez pour les nœuds hybrides doit être acheminé vers le VPC que vous utilisez pour votre cluster EKS. Plusieurs options de connectivité Network-to-Amazon VPC sont disponibles pour connecter votre réseau local à un VPC. Vous pouvez également utiliser votre propre solution VPN.

Il est important de configurer correctement le routage côté AWS cloud dans le VPC et dans votre réseau sur site, afin que les deux réseaux acheminent le trafic approprié via la connexion des deux réseaux.

Dans le VPC, tout le trafic à destination du nœud distant et des réseaux de pods distants doit transiter par la connexion à votre réseau local (appelée « passerelle »). Si certains de vos sous-réseaux ont des tables de routage différentes, vous devez configurer chaque table de routage avec les routes pour vos nœuds hybrides et les pods qui s'exécutent sur ceux-ci. Cela est vrai pour les sous-réseaux auxquels le plan ENIs de contrôle EKS est attaché, ainsi que pour les sous-réseaux contenant des EC2 nœuds ou des pods devant communiquer avec des nœuds hybrides.

Dans votre réseau sur site, vous devez configurer votre réseau pour autoriser le trafic à destination et en provenance du VPC de votre cluster EKS et les AWS autres services requis pour les nœuds hybrides. Le trafic pour le cluster EKS traverse la passerelle dans les deux sens.

Contraintes du réseau

Réseau entièrement routé

La principale contrainte réside dans le fait que le plan de contrôle EKS et tous les nœuds, qu'ils soient cloud ou hybrides, doivent former un réseau entièrement routé. Cela signifie que tous les nœuds doivent pouvoir se joindre mutuellement au niveau de la couche trois, par adresse IP.

Le plan de contrôle EKS et les nœuds cloud sont déjà accessibles les uns depuis les autres, car ils se trouvent dans un réseau plat (le VPC). Les nœuds hybrides se trouvent toutefois dans un domaine de réseau différent. C'est pourquoi vous devez configurer un routage supplémentaire dans le VPC et sur votre réseau local afin d'acheminer le trafic entre les nœuds hybrides et le reste du cluster. Si les nœuds hybrides sont accessibles les uns depuis les autres et depuis le VPC, vos nœuds hybrides peuvent se trouver dans un seul réseau plat ou dans plusieurs réseaux segmentés.

Module de télécommande routable CIDRs

Pour que le plan de contrôle EKS puisse communiquer avec les pods s'exécutant sur des nœuds hybrides (par exemple, les webhooks ou le serveur Metrics) ou pour que les pods s'exécutant sur des nœuds cloud puissent communiquer avec les pods s'exécutant sur des nœuds hybrides

(communication est-ouest de la charge de travail), votre CIDR de pod distant doit être routable à partir du VPC. Cela signifie que le VPC doit être en mesure d'acheminer le trafic vers le pod CIDRs via la passerelle vers votre réseau local et que votre réseau local doit être en mesure d'acheminer le trafic d'un pod vers le bon nœud.

Il est important de noter la distinction entre les exigences de routage des pods dans le VPC et sur site. Le VPC doit uniquement savoir que tout trafic destiné à un pod distant doit passer par la passerelle. Si vous ne possédez qu'un seul CIDR du pod distant, vous n'avez besoin que d'un seul itinéraire.

Cette exigence s'applique à tous les sauts de votre réseau local jusqu'au routeur local situé dans le même sous-réseau que vos nœuds hybrides. C'est le seul routeur qui doit connaître la tranche CIDR du pod attribuée à chaque nœud, afin de s'assurer que le trafic destiné à un pod particulier est acheminé vers le nœud où le pod a été planifié.

Vous pouvez choisir de propager ces routes pour le pod local CIDRs depuis votre routeur local vers les tables de routage VPC, mais cela n'est pas nécessaire. Si votre espace sur site CIDRs change fréquemment et que vos tables de routage VPC doivent être mises à jour pour refléter le changement d' CIDRsespace, nous vous recommandons de propager l'espace sur site CIDRs vers les tables de routage VPC, mais cela est rare.

Notez que la contrainte permettant de rendre votre module local CIDRs routable est facultative. Si vous n'avez pas besoin d'exécuter des webhooks sur vos nœuds hybrides ou de laisser les pods sur les nœuds cloud communiquer avec les pods sur les nœuds hybrides, vous n'avez pas besoin de configurer le routage pour le pod CIDRs sur votre réseau local.

Pourquoi le pod sur site CIDRs doit-il être routable avec des nœuds hybrides ?

Lorsque vous utilisez EKS avec le VPC CNI pour vos nœuds de cloud, le VPC CNI est attribué directement IPs du VPC aux pods. Cela signifie qu'aucun routage spécial n'est nécessaire, car les modules cloud et le plan de contrôle EKS peuvent accéder IPs directement au pod.

Lorsqu'ils sont exécutés sur site (et avec d'autres CNIs dans le cloud), les pods s'exécutent généralement sur un réseau superposé isolé et le CNI se charge de distribuer le trafic entre les pods. Cela se fait généralement par le biais de l'encapsulation : le CNI convertit le pod-to-pod trafic en node-to-node trafic, en se chargeant de l'encapsulation et du désencapsulage des deux côtés. De cette manière, aucune configuration supplémentaire n'est nécessaire sur les nœuds et les routeurs.

La mise en réseau avec des nœuds hybrides est unique, car elle combine les deux topologies : le plan de contrôle EKS et les nœuds cloud (avec le VPC CNI) s'attendent à un réseau plat comprenant

des nœuds et des pods, tandis que les pods s'exécutant sur des nœuds hybrides se trouvent dans un réseau superposé utilisant VXLAN pour l'encapsulation (par défaut dans Cilium). Les pods exécutés sur des nœuds hybrides peuvent atteindre le plan de contrôle EKS et les pods exécutés sur des nœuds cloud, à condition que le réseau local puisse acheminer vers le VPC. Toutefois, sans routage pour le module CIDRs sur le réseau local, tout trafic revenant vers une adresse IP d'espace local sera finalement abandonné si le réseau ne sait pas comment atteindre le réseau superposé et acheminer vers les nœuds appropriés.

Concepts Kubernetes pour les nœuds hybrides

Cette page détaille les concepts clés de Kubernetes qui sous-tendent l'architecture système des nœuds hybrides EKS.

Plan de contrôle EKS dans le VPC

Les adresses IP des ENI du plan de contrôle EKS sont stockées dans l'objet `kubernetes Endpoints` dans l'espace de noms `default`. Lorsque EKS crée de nouveaux ENI ou supprime les anciens, EKS met à jour cet objet afin que la liste des adresses IP soit toujours à jour.

Vous pouvez utiliser ces points de terminaison via le service `kubernetes`, également dans l'espace de noms `default`. Ce service, de type `ClusterIP`, se voit toujours attribuer la première adresse IP du CIDR du service du cluster. Par exemple, pour le service CIDR `172.16.0.0/16`, l'adresse IP du service sera `172.16.0.1`.

En général, c'est ainsi que les pods (qu'ils fonctionnent dans le cloud ou sur des nœuds hybrides) accèdent au serveur API EKS Kubernetes. Les pods utilisent l'adresse IP du service comme adresse IP de destination, qui est traduite en adresses IP réelles de l'une des ENI du plan de contrôle EKS. La principale exception est `kube-proxy`, car elle configure la traduction.

point de terminaison de serveur d'API E

L'adresse IP du service `kubernetes` n'est pas le seul moyen d'accéder au serveur API EKS. EKS crée également un nom DNS Route53 lorsque vous créez votre cluster. Il s'agit du champ `endpoint` de votre cluster EKS lorsque vous appelez l'action API `DescribeCluster` EKS.

```
{
    "cluster": {
        "endpoint": "https://xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx.gr7.us-
west-2.eks.amazonaws.com",
        "name": "my-cluster",
        "status": "ACTIVE"
```

```
    }
}
```

Dans un cluster d'accès aux points de terminaison publics ou d'accès aux points de terminaison publics et privés, vos nœuds hybrides résoudront ce nom DNS en une adresse IP publique par défaut, routable via Internet. Dans un cluster d'accès aux points de terminaison privés, le nom DNS est résolu en adresses IP privées des ENI du plan de contrôle EKS.

Voici comment le `kubelet` et `kube-proxy` accèdent au serveur API Kubernetes. Si vous souhaitez que tout le trafic de votre cluster Kubernetes passe par le VPC, vous devez soit configurer votre cluster en mode d'accès privé, soit modifier votre serveur DNS sur site afin de résoudre le point de terminaison du cluster EKS vers les adresses IP privées des ENI du plan de contrôle EKS.

Point de terminaison **`kubelet`**

Le `kubelet` expose plusieurs points de terminaison REST, permettant à d'autres parties du système d'interagir avec chaque nœud et d'y recueillir des informations. Dans la plupart des clusters, la majorité du trafic vers le serveur `kubelet` provient du plan de contrôle, mais certains agents de surveillance peuvent également interagir avec lui.

Grâce à cette interface, le `kubelet` gère diverses requêtes : récupération des journaux (`kubectl logs`), exécution de commandes à l'intérieur des conteneurs (`kubectl exec`) et redirection du trafic (`kubectl port-forward`). Chacune de ces requêtes interagit avec l'exécution de conteneur sous-jacent via `kubelet`, ce qui semble transparent pour les administrateurs de cluster et les développeurs.

Le serveur d'API Kubernetes est le serveur d'API Kubernetes. Lorsque vous utilisez l'une des commandes `kubectl` mentionnées précédemment, `kubectl` envoie une requête API au serveur API, qui appelle ensuite l'API `kubelet` du nœud sur lequel le pod est exécuté. C'est la principale raison pour laquelle l'adresse IP du nœud doit être accessible depuis le plan de contrôle EKS et pourquoi, même si vos pods sont en cours d'exécution, vous ne pourrez pas accéder à leurs journaux ou à `exec` si la route du nœud est mal configurée.

IP des nœuds

Lorsque le plan de contrôle EKS communique avec un nœud, il utilise l'une des adresses indiquées dans l'état de l'objet `Node` (`status.addresses`).

Avec les nœuds cloud EKS, il est courant que le kubelet signale l'adresse IP privée de l'instance EC2 comme une `InternalIP` lors de l'enregistrement du nœud. Cette adresse IP est ensuite validée par

le Cloud Controller Manager (CCM) qui s'assure qu'elle appartient bien à l'instance EC2. De plus, le CCM ajoute généralement les adresses IP publiques (comme `ExternalIP`) et les noms DNS (`InternalDNS` et `ExternalDNS`) de l'instance au statut du nœud.

Cependant, il n'existe pas de CCM pour les nœuds hybrides. Lorsque vous enregistrez un nœud hybride avec la CLI des nœuds hybrides EKS (`nodeadm`), celle-ci configure le kubelet pour qu'il signale l'adresse IP de votre machine directement dans l'état du nœud, sans passer par le CCM.

```
apiVersion: v1
kind: Node
metadata:
  name: my-node-1
spec:
  providerID: eks-hybrid:///us-west-2/my-cluster/my-node-1
status:
  addresses:
  - address: 10.1.1.236
    type: InternalIP
  - address: my-node-1
    type: Hostname
```

Si votre machine dispose de plusieurs adresses IP, le kubelet en sélectionnera une selon sa propre logique. Vous pouvez contrôler l'adresse IP sélectionnée à l'aide du drapeau `--node-ip`, que vous pouvez passer dans la configuration `nodeadm` dans `spec.kubelet.flags`. Seule l'adresse IP indiquée dans l'objet `Node` nécessite une route depuis le VPC. Vos machines peuvent avoir d'autres adresses IP qui ne sont pas accessibles depuis le cloud.

kube-proxy

`kube-proxy` est responsable de la mise en œuvre de l'abstraction du service au niveau de la couche réseau de chaque nœud. Il agit comme un proxy réseau et un équilibreur de charge pour le trafic destiné aux services Kubernetes. En surveillant en permanence le serveur API Kubernetes pour détecter les changements liés aux services et aux points de terminaison, `kube-proxy` met à jour de manière dynamique les règles de réseau de l'hôte sous-jacent afin de garantir que le trafic est correctement acheminé.

En mode `iptables`, `kube-proxy` programme plusieurs chaînes `netfilter` pour gérer le trafic de service. Les règles forment la hiérarchie suivante :

1. Chaîne KUBE-SERVICES : point d'entrée pour tout le trafic de services. Il comporte des règles correspondant à chaque `ClusterIP` et port du service.

2. Chaînes KUBE-SVC-XXX : les chaînes spécifiques à un service disposent de règles d'équilibrage de charge pour chaque service.
3. Chaînes KUBE-SEP-XXX : les chaînes spécifiques aux points de terminaison contiennent les règles DNAT effectives.

Examinons ce qui se passe pour un service `test-server` dans l'espace de noms `default` : * Service ClusterIP : `172.16.31.14` * Port du service : `80` * Pods de sauvegarde : `10.2.0.110`, `10.2.1.39`, et `10.2.2.254`

Lorsque nous inspectons les règles `iptables` (à l'aide de `iptables-save 0 grep -A10 KUBE-SERVICES`) :

1. Dans la chaîne KUBE-SERVICES, nous trouvons une règle correspondant au service :

```
-A KUBE-SERVICES -d 172.16.31.14/32 -p tcp -m comment --comment "default/test-server
 cluster IP" -m tcp --dport 80 -j KUBE-SVC-XYZABC123456
```

   - Cette règle correspond aux paquets destinés à 172.16.31.14:80
   - Le commentaire indique à quoi sert cette règle : `default/test-server cluster IP`
   - Les paquets correspondants passent à la chaîne `KUBE-SVC-XYZABC123456`
2. La chaîne KUBE-SVC-XYZABC123456 dispose de règles d'équilibrage de charge basées sur la probabilité :

```
-A KUBE-SVC-XYZABC123456 -m statistic --mode random --probability 0.33333333349 -j
 KUBE-SEP-POD1XYZABC
-A KUBE-SVC-XYZABC123456 -m statistic --mode random --probability 0.50000000000 -j
 KUBE-SEP-POD2XYZABC
-A KUBE-SVC-XYZABC123456 -j KUBE-SEP-POD3XYZABC
```

   - Première règle : 33,3 % de chances de passer à `KUBE-SEP-POD1XYZABC`
   - Deuxième règle : 50 % de chances que le trafic restant (33,3 % du total) passe à `KUBE-SEP-POD2XYZABC`
   - Dernière règle : tout le trafic restant (33,3 % du total) passe à `KUBE-SEP-POD3XYZABC`
3. Les chaînes KUBE-SEP-XXX individuelles exécutent le DNAT (Destination NAT) :

```
-A KUBE-SEP-POD1XYZABC -p tcp -m tcp -j DNAT --to-destination 10.2.0.110:80
-A KUBE-SEP-POD2XYZABC -p tcp -m tcp -j DNAT --to-destination 10.2.1.39:80
```

```
-A KUBE-SEP-POD3XYZABC -p tcp -m tcp -j DNAT --to-destination 10.2.2.254:80
```

- Ces règles DNAT réécrivent l'adresse IP et le port de destination afin de diriger le trafic vers des pods spécifiques.
- Chaque règle gère environ 33,3 % du trafic, assurant ainsi un équilibrage de charge uniforme entre `10.2.0.110`, `10.2.1.39` et `10.2.2.254`.

Cette structure en chaîne à plusieurs niveaux permet à `kube-proxy` de mettre en œuvre efficacement l'équilibrage de charge et la redirection des services grâce à la manipulation des paquets au niveau du noyau, sans nécessiter de processus proxy dans le chemin de données.

Impact sur les opérations Kubernetes

Une `kube-proxy` rompu sur un nœud empêche ce dernier d'acheminer correctement le trafic du service, ce qui entraîne des délais d'attente ou des échecs de connexion pour les pods qui dépendent des services du cluster. Cela peut être particulièrement perturbant lors de la première inscription d'un nœud. Le CNI doit communiquer avec le serveur API Kubernetes pour obtenir des informations, telles que le CIDR du pod du nœud, avant de pouvoir configurer le réseau des pods. Pour ce faire, il utilise l'adresse IP du service `kubernetes`. Cependant, si `kube-proxy` n'a pas pu démarrer ou n'a pas réussi à définir les règles `iptables` appropriées, les requêtes envoyées à l'adresse IP du service `kubernetes` ne sont pas traduites en adresses IP réelles des ENI du plan de contrôle EKS. En conséquence, le CNI entrera dans une boucle de crash et aucun des pods ne pourra fonctionner correctement.

Nous savons que les pods utilisent l'adresse IP `kubernetes` du service pour communiquer avec le serveur API Kubernetes, mais il faut d'abord que `kube-proxy` définisse des règles `iptables` pour que cela fonctionne.

Comment `kube-proxy` communique-t-il avec le serveur API ?

Le `kube-proxy` doit être configuré pour utiliser les adresses IP réelles du serveur API Kubernetes ou un nom DNS qui les résout. Dans le cas d'EKS, EKS configure la valeur `kube-proxy` par défaut pour pointer vers le nom DNS Route53 créé par EKS lors de la création du cluster. Vous pouvez voir cette valeur dans le `kube-proxy` ConfigMap dans l'espace de noms `kube-system`. Le contenu de ce ConfigMap est un `kubeconfig` qui est injecté dans le pod `kube-proxy`, recherchez donc le champ `clusters0.cluster.server`. Cette valeur correspondra au champ `endpoint` de votre cluster EKS (lorsque vous appelez l'API `DescribeCluster` EKS).

```
apiVersion: v1
```

```
data:
  kubeconfig: |-
    kind: Config
    apiVersion: v1
    clusters:
    - cluster:
        certificate-authority: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
        server: https://xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx.gr7.us-
west-2.eks.amazonaws.com
      name: default
    contexts:
    - context:
        cluster: default
        namespace: default
        user: default
      name: default
    current-context: default
    users:
    - name: default
      user:
        tokenFile: /var/run/secrets/kubernetes.io/serviceaccount/token
kind: ConfigMap
metadata:
  name: kube-proxy
  namespace: kube-system
```

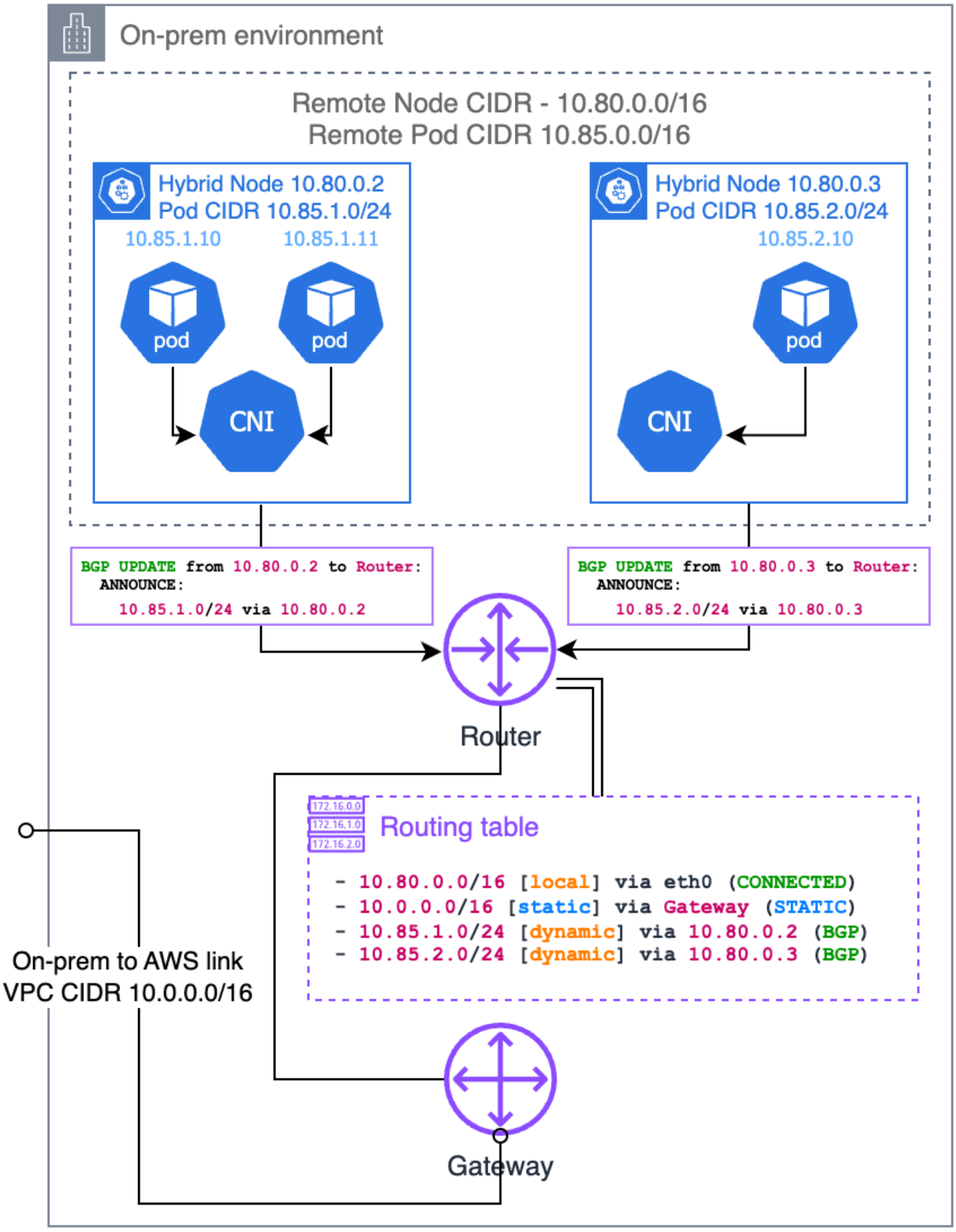
CIDR de pods distants routables

Cette page the section called "Concepts de réseau" détaille les conditions requises pour exécuter des webhooks sur des nœuds hybrides ou pour que les pods exécutés sur des nœuds cloud communiquent avec les pods exécutés sur des nœuds hybrides. La condition essentielle est que le routeur sur site doit savoir quel nœud est responsable d'une adresse IP de pod particulière. Il existe plusieurs façons d'y parvenir, notamment le protocole de passerelle frontière (BGP), les routes statiques et le proxy ARP (Address Resolution Protocol). Ces points sont abordés dans les sections suivantes.

Protocole de passerelle frontière (BGP)

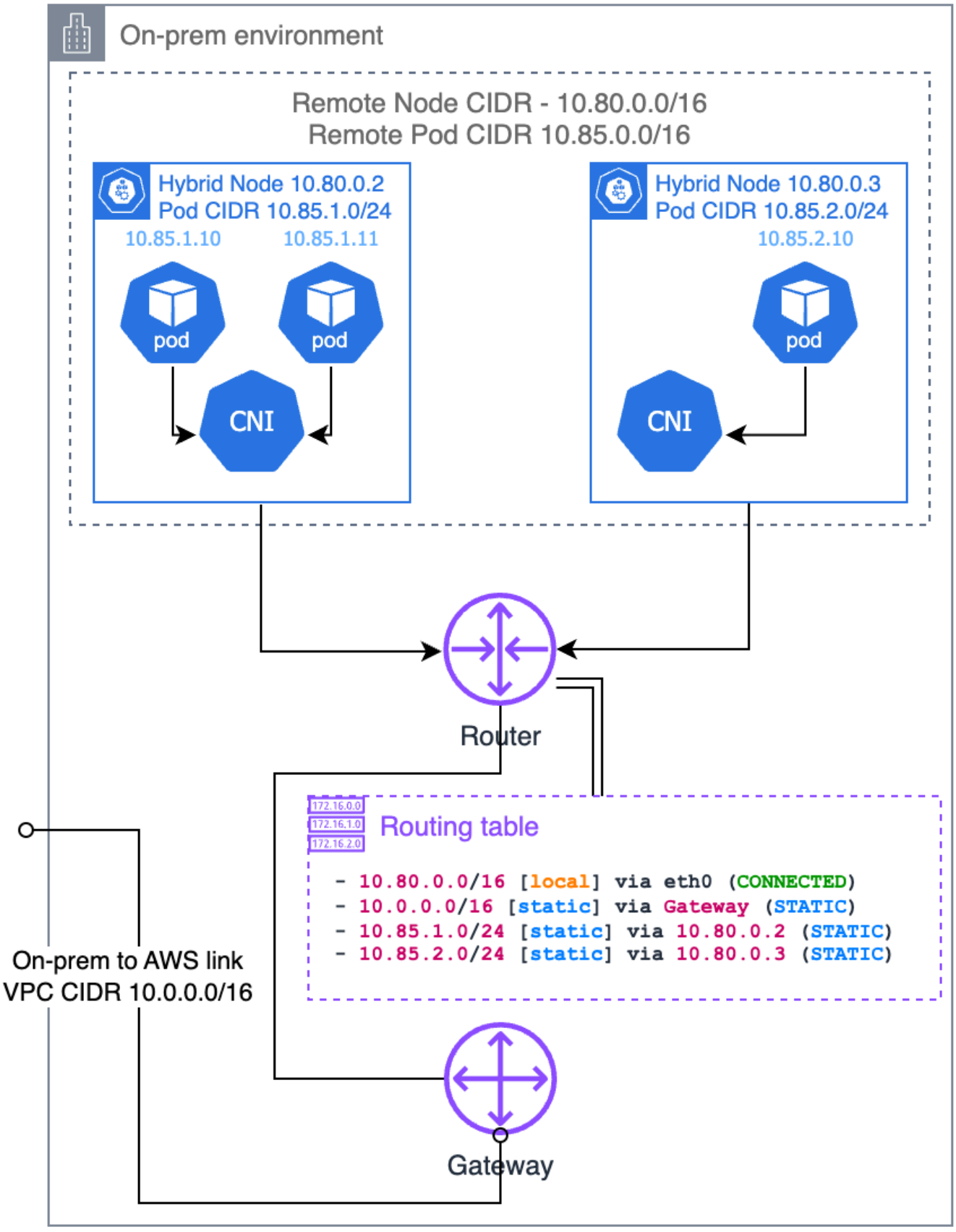
Si votre CNI le prend en charge (comme Cilium et Calico), vous pouvez utiliser le mode BGP de votre CNI pour propager les routes vers vos CIDR de pod par nœud depuis vos nœuds vers votre routeur local. Lorsque vous utilisez le mode BGP du CNI, votre CNI agit comme un routeur virtuel, de sorte

que votre routeur local considère que le CIDR du pod appartient à un sous-réseau différent et que votre nœud est la passerelle vers ce sous-réseau.


On-prem environment
Remote Node CIDR - 10.80.0.0/16
Remote Pod CIDR 10.85.0.0/16
Hybrid Node 10.80.0.2
Pod CIDR 10.85.1.0/24
10.85.1.10
10.85.1.11
pod
pod
CNI
Hybrid Node 10.80.0.3
Pod CIDR 10.85.2.0/24
10.85.2.10
pod
CNI
BGP UPDATE from 10.80.0.2 to Router:
ANNOUNCE:
10.85.1.0/24 via 10.80.0.2
BGP UPDATE from 10.80.0.3 to Router:
ANNOUNCE:
10.85.2.0/24 via 10.80.0.3
Router
Routing table
- 10.80.0.0/16 [local] via eth0 (CONNECTED)
- 10.0.0.0/16 [static] via Gateway (STATIC)
- 10.85.1.0/24 [dynamic] via 10.80.0.2 (BGP)
- 10.85.2.0/24 [dynamic] via 10.80.0.3 (BGP)
On-prem to AWS link
VPC CIDR 10.0.0.0/16
Gateway

Routes statiques

Vous pouvez également configurer des routes statiques dans votre routeur local. Il s’agit de la méthode la plus simple pour acheminer le CIDR du pod sur site vers votre VPC, mais c’est aussi la plus sujette aux erreurs et la plus difficile à maintenir. Vous devez vous assurer que les routes sont toujours à jour avec les nœuds existants et leurs CIDR de pod attribués. Si votre nombre de nœuds est faible et que votre infrastructure est statique, cette option est viable et élimine le besoin d’une prise en charge BGP dans votre routeur. Si vous optez pour cette solution, nous vous recommandons de configurer votre CNI avec la tranche CIDR du pod que vous souhaitez attribuer à chaque nœud, plutôt que de laisser son IPAM décider.


On-prem environment
Remote Node CIDR - 10.80.0.0/16
Remote Pod CIDR 10.85.0.0/16
Hybrid Node 10.80.0.2
Pod CIDR 10.85.1.0/24
10.85.1.10
10.85.1.11
pod
pod
CNI
Hybrid Node 10.80.0.3
Pod CIDR 10.85.2.0/24
10.85.2.10
pod
CNI
Router
172.16.0.0
172.16.1.0
172.16.2.0
Routing table
- 10.80.0.0/16 [local] via eth0 (CONNECTED)
- 10.0.0.0/16 [static] via Gateway (STATIC)
- 10.85.1.0/24 [static] via 10.80.0.2 (STATIC)
- 10.85.2.0/24 [static] via 10.80.0.3 (STATIC)
On-prem to AWS link
VPC CIDR 10.0.0.0/16
Gateway

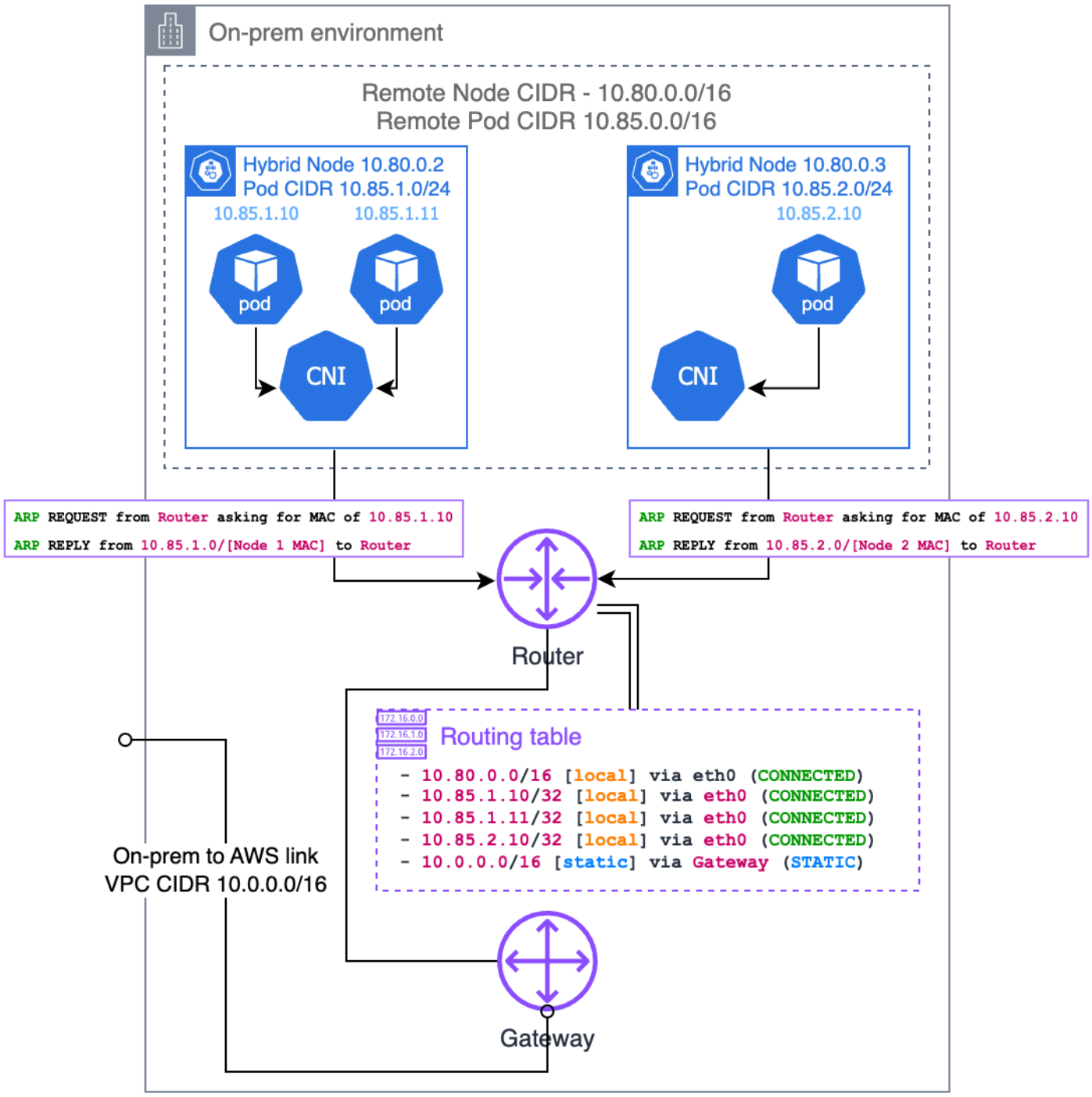
Proxy ARP (Address Resolution Protocol)

Le proxy ARP est une autre approche permettant de rendre routables les adresses IP des pods sur site, particulièrement utile lorsque vos nœuds hybrides se trouvent sur le même réseau de couche 2 que votre routeur local. Lorsque le proxy ARP est activé, un nœud répond aux requêtes ARP pour les adresses IP des pods qu'il héberge, même si ces adresses IP appartiennent à un sous-réseau différent.

Lorsqu'un appareil de votre réseau local tente d'atteindre une adresse IP de pod, il envoie d'abord une requête ARP demandant « Qui possède cette adresse IP ? ». Le nœud hybride hébergeant ce pod répondra avec sa propre adresse MAC, indiquant « Je peux gérer le trafic pour cette adresse IP ». Cela crée un chemin direct entre les appareils de votre réseau local et les pods sans nécessiter de configuration du routeur.

Pour que cela fonctionne, votre CNI doit prendre en charge la fonctionnalité ARP proxy. Cilium dispose d'un support intégré pour le proxy ARP que vous pouvez activer via la configuration. Le point essentiel à prendre en compte est que le CIDR du pod ne doit pas chevaucher aucun autre réseau de votre environnement, car cela pourrait entraîner des conflits de routage.

Cette approche présente plusieurs avantages : * Pas besoin de configurer votre routeur avec BGP ou de maintenir des routes statiques * Fonctionne bien dans les environnements où vous n'avez pas le contrôle sur la configuration de votre routeur



Encapsulation de pod à pod

Dans les environnements sur site, les CNI utilisent généralement des protocoles d'encapsulation pour créer des réseaux superposés qui peuvent fonctionner au-dessus du réseau physique sans qu'il soit nécessaire de le reconfigurer. Cette section explique comment fonctionne cette encapsulation. Veuillez noter que certains détails peuvent varier en fonction du CNI que vous utilisez.

L'encapsulation consiste à envelopper les paquets réseau du pod d'origine dans un autre paquet réseau qui peut être acheminé via le réseau physique sous-jacent. Cela permet aux pods de communiquer entre les nœuds exécutant le même CNI sans que le réseau physique ait besoin de savoir comment acheminer ces CIDR de pod.

Le protocole d'encapsulation le plus couramment utilisé avec Kubernetes est le Virtual Extensible LAN (VXLAN), mais d'autres protocoles (tels que `Geneve`) sont également disponibles en fonction de votre CNI.

Encapsulation VXLAN

VXLAN encapsule les trames Ethernet de couche 2 dans des paquets UDP. Lorsqu'un pod envoie du trafic à un autre pod sur un nœud différent, le CNI effectue les opérations suivantes :

1. Le CNI intercepte les paquets provenant du Pod A.
2. Le CNI encapsule le paquet d'origine dans un en-tête VXLAN.
3. Ce paquet encapsulé est ensuite envoyé via la pile réseau standard du nœud vers le nœud de destination.
4. Le CNI sur le nœud de destination déballe le paquet et le transmet au Pod B

Voici ce qui arrive à la structure du paquet lors de l'encapsulation VXLAN :

Paquet original de pod à pod :

```
+-----------------+---------------+-------------+-----------------+
| Ethernet Header | IP Header     | TCP/UDP     | Payload         |
| Src: Pod A MAC  | Src: Pod A IP | Src Port    |                 |
| Dst: Pod B MAC  | Dst: Pod B IP | Dst Port    |                 |
+-----------------+---------------+-------------+-----------------+
```

Après l'encapsulation VXLAN :

```
+-----------------+-------------+--------------+------------
+--------------------------+
| Outer Ethernet  | Outer IP    | Outer UDP    | VXLAN      | Original Pod-to-Pod
 |
| Src: Node A MAC | Src: Node A | Src: Random  | VNI: xx    | Packet (unchanged
 |
| Dst: Node B MAC | Dst: Node B | Dst: 4789    |            | from above)
 |
```

```
+-----------------+-------------+--------------+------------
+--------------------------+
```

L'identifiant réseau VXLAN (VNI) permet de distinguer les différents réseaux superposés.

Scénarios de communication de pod

Pods sur le même nœud hybride

Lorsque des pods sur le même nœud hybride communiquent, aucune encapsulation n'est généralement nécessaire. Le CNI configure des routes locales qui dirigent le trafic entre les pods via les interfaces virtuelles internes du nœud :

```
Pod A -> veth0 -> node's bridge/routing table -> veth1 -> Pod B
```

Le paquet ne quitte jamais le nœud et ne nécessite pas d'encapsulation.

Pods sur différents nœuds hybrides

La communication entre les pods sur différents nœuds hybrides nécessite une encapsulation :

```
Pod A -> CNI -> [VXLAN encapsulation] -> Node A network -> router or gateway -> Node B
 network -> [VXLAN decapsulation] -> CNI -> Pod B
```

Cela permet au trafic des pods de traverser l'infrastructure réseau physique sans que celle-ci ait besoin de comprendre le routage IP des pods.

Flux de trafic réseau pour les nœuds hybrides

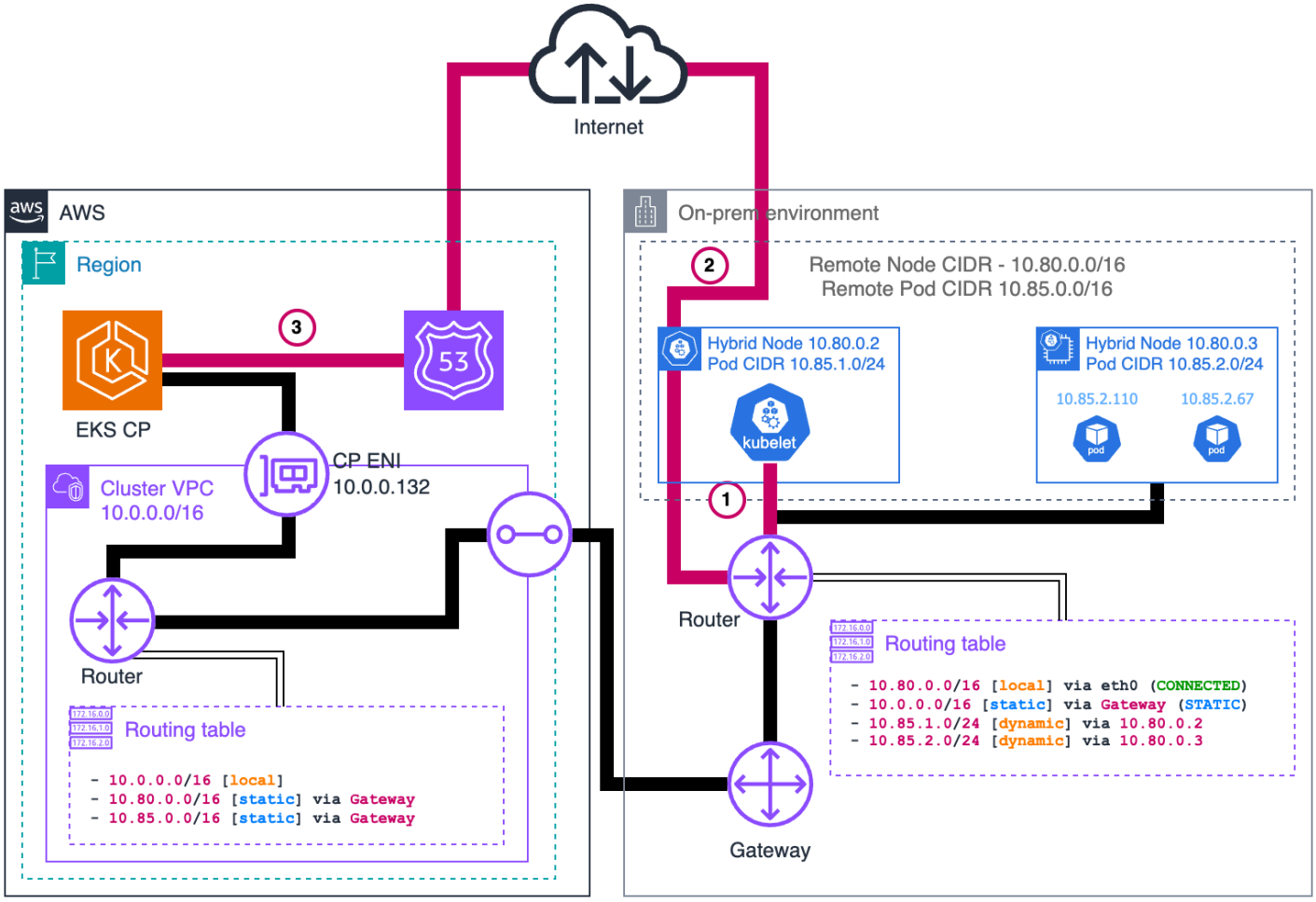
Cette page détaille les flux de trafic réseau pour les nœuds hybrides EKS avec des diagrammes montrant les chemins end-to-end réseau pour les différents types de trafic.

Les flux de trafic suivants sont couverts :

- the section called "Nœud hybride `kubelet` vers plan de contrôle EKS"
- the section called "Plan de contrôle EKS vers nœud hybride (serveur `kubelet`)"
- the section called "Pods exécutés sur des nœuds hybrides vers le plan de contrôle EKS"
- the section called "Plan de contrôle EKS pour les pods exécutés sur un nœud hybride (webhooks)"
- the section called "Pod-to-Pod exécution sur des nœuds hybrides"

- the section called "Des pods sur des nœuds cloud vers des pods sur des nœuds hybrides (trafic est-ouest)"

Nœud hybride **`kubelet`** vers plan de contrôle EKS



Demande

1**`kubelet`**. Lance une demande

Lorsque `kubelet` sur un nœud hybride doit communiquer avec le plan de contrôle EKS (par exemple, pour signaler l'état du nœud ou obtenir les spécifications du pod), il utilise le `kubeconfig` fourni lors de l'enregistrement du nœud. Ce `kubeconfig` possède l'URL du point de terminaison du serveur API (le nom DNS Route53) plutôt que d'adresses IP directes.

Le `kubelet` effectue une recherche DNS pour le point de terminaison (par exemple, `https://xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx.gr7.us-west-2.eks.amazonaws.com`). Dans un cluster à accès public, cela se résout en une adresse IP publique (par exemple `54.239.118.52`)

qui appartient au service EKS s'exécutant dans AWS. Le `kubelet` crée ensuite une requête HTTPS sécurisée vers ce point de terminaison. Le paquet initial ressemble à ceci :

```
+--------------------+---------------------+-----------------+
| IP Header          | TCP Header          | Payload         |
| Src: 10.80.0.2     | Src: 52390 (random) |                 |
| Dst: 54.239.118.52 | Dst: 443            |                 |
+--------------------+---------------------+-----------------+
```

2. Routage du routeur local

Étant donné que l'adresse IP de destination est une adresse IP publique et ne fait pas partie du réseau sur site, le `kubelet` envoie ce paquet à sa passerelle par défaut (le routeur local sur site). Le routeur examine l'adresse IP de destination et détermine qu'il s'agit d'une adresse IP publique.

Pour le trafic public, le routeur transfère généralement le paquet vers une passerelle Internet ou un routeur frontalier qui gère le trafic sortant vers Internet. Ceci n'apparaît pas dans le schéma et dépendra de la configuration de votre réseau sur site. Le paquet traverse votre infrastructure réseau sur site et atteint finalement le réseau de votre fournisseur d'accès Internet.

3. Livraison au plan de commande EKS

Le paquet traverse l'Internet public et les réseaux de transit jusqu'à ce qu' AWS il atteigne le réseau. AWS le réseau achemine le paquet vers le point de terminaison du service EKS dans la région appropriée. Lorsque le paquet atteint le service EKS, il est transféré vers le plan de contrôle EKS réel de votre cluster.

Ce routage via l'Internet public est différent du chemin privé routé par VPC que nous verrons dans d'autres flux de trafic. La principale différence réside dans le fait que, lorsque vous utilisez le mode d'accès public, le trafic provenant des `kubelet` locales (mais pas des pods) vers le plan de contrôle EKS ne passe pas par votre VPC, mais utilise plutôt l'infrastructure Internet mondiale.

Réponse

Une fois que le plan de contrôle EKS a traité la demande `kubelet`, il renvoie une réponse :

3. Le plan de contrôle EKS envoie une réponse

Le plan de contrôle EKS crée un paquet de réponse. Ce paquet a l'adresse IP publique comme source et l'adresse IP du nœud hybride comme destination :

```
+--------------------+---------------------+-----------------+
| IP Header          | TCP Header          | Payload         |
| Src: 54.239.118.52 | Src: 443            |                 |
| Dst: 10.80.0.2     | Dst: 52390          |                 |
+--------------------+---------------------+-----------------+
```

2. Routage Internet

Le paquet de réponse retourne sur Internet, en suivant le chemin de routage déterminé par les fournisseurs d'accès Internet, jusqu'à ce qu'il atteigne le routeur périphérique de votre réseau sur site.

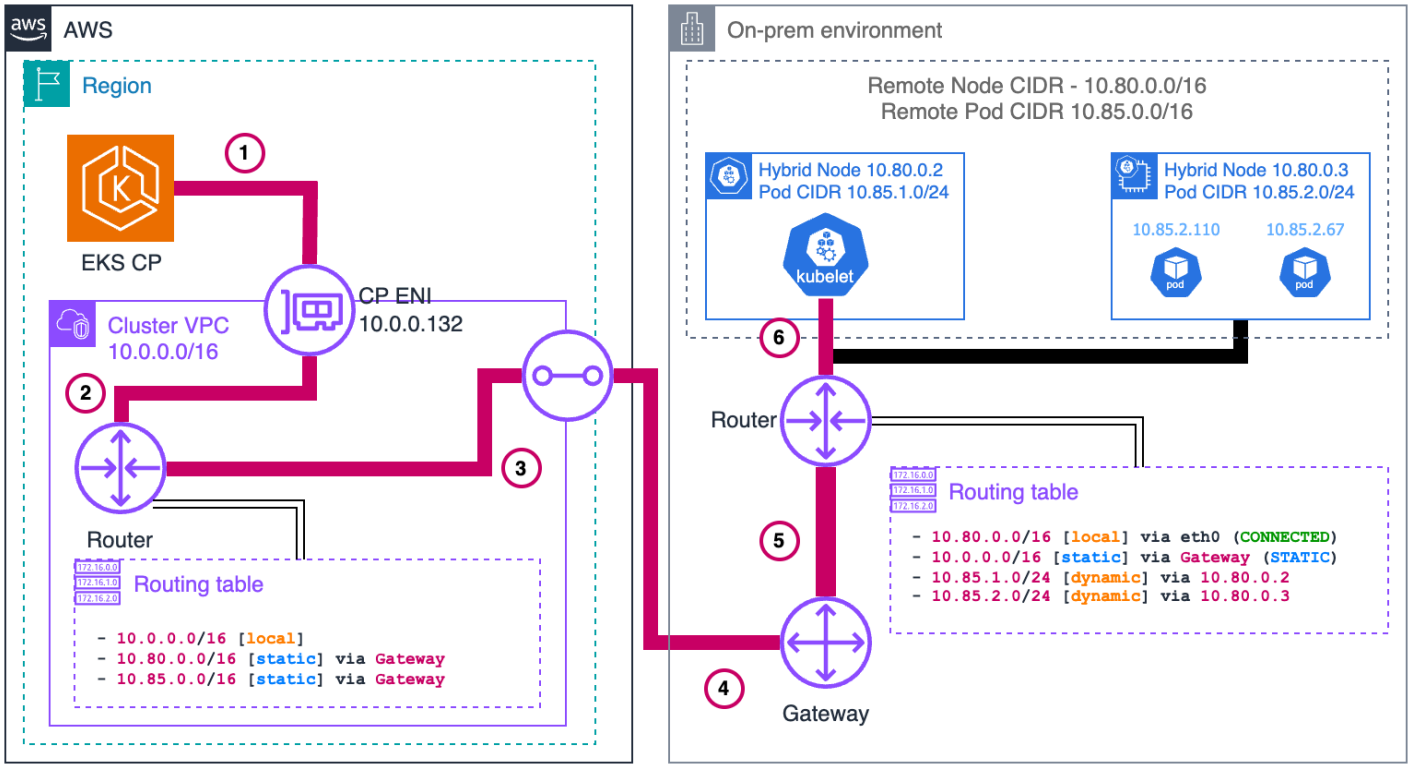
1. Livraison locale

Votre routeur local reçoit le paquet et reconnaît que l'adresse IP de destination (`10.80.0.2`) appartient à votre réseau sur site. Il transmet le paquet via votre infrastructure réseau locale jusqu'à ce qu'il atteigne le nœud hybride cible, où `kubelet` reçoit et traite la réponse.

Nœud hybride **`kube-proxy`** vers plan de contrôle EKS

Si vous activez l'accès public aux points de terminaison pour le cluster, le trafic de retour utilise l'Internet public. Ce trafic provient du `kube-proxy` sur le nœud hybride vers le plan de contrôle EKS et suit le même chemin que le trafic provenant du `kubelet` vers le plan de contrôle EKS.

Plan de contrôle EKS vers nœud hybride (serveur **`kubelet`**)



Demande

1. Le serveur d'API EKS Kubernetes lance une demande

Le serveur API EKS Kubernetes extrait l'adresse IP du nœud (`10.80.0.2`) à partir de l'état de l'objet nœud. Il achemine ensuite cette demande via son ENI dans le VPC, car l'adresse IP de destination appartient au nœud distant configuré CIDR (`10.80.0.0/16`). Le paquet initial ressemble à ceci :

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.0.0.132 | Src: 67493 (random) |                 |
| Dst: 10.80.0.2  | Dst: 10250          |                 |
+-----------------+---------------------+-----------------+
```

2. Traitement réseau VPC

Le paquet quitte l'ENI et entre dans la couche réseau VPC, où il est dirigé vers la passerelle du sous-réseau pour un routage supplémentaire.

3. Recherche dans une table de routage VPC

La table de routage VPC pour le sous-réseau contenant l'ENI du plan de contrôle EKS dispose d'un itinéraire spécifique (le deuxième dans le diagramme) pour le CIDR du nœud distant. Sur la base de cette règle de routage, le paquet est dirigé vers la VPC-to-onprem passerelle.

4. Transit transfrontalier

La passerelle transfère le paquet à travers la limite du cloud via votre connexion établie (telle que Direct Connect ou VPN) vers votre réseau sur site.

5. Réception réseau sur site

Le paquet arrive à votre routeur local sur site qui gère le trafic pour le sous-réseau où se trouvent vos nœuds hybrides.

6. Livraison finale

Le routeur local identifie que l'adresse IP (`10.80.0.2`) de destination appartient à son réseau directement connecté et transfère le paquet directement au nœud hybride cible, où le `kubelet` reçoit et traite la requête.

Réponse

Une fois que le nœud hybride `kubelet` a traité la demande, il renvoie une réponse en suivant le même chemin en sens inverse :

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.80.0.2  | Src: 10250          |                 |
| Dst: 10.0.0.132 | Dst: 67493          |                 |
+-----------------+---------------------+-----------------+
```

6. **`kubelet`** Envoie une réponse

`kubelet`Le nœud hybride (`10.80.0.2`) crée un paquet de réponse avec l'adresse IP source d'origine comme destination. La destination n'appartenant pas au réseau local, elle est envoyée à la passerelle par défaut de l'hôte, qui est le routeur local.

5. Routage du routeur local

Le routeur détermine que l'adresse IP de destination (`10.0.0.132`) appartient à `10.0.0.0/16`, qui dispose d'une route pointant vers la passerelle connectée à AWS.

4. Retour transfrontalier

Le paquet retourne par la même connexion locale vers VPC (telle que Direct Connect ou VPN), franchissant la limite du cloud dans le sens inverse.

3. Routage VPC

Lorsque le paquet arrive dans le VPC, les tables de routage identifient que l'adresse IP de destination appartient à un CIDR VPC. Les acheminements de paquets au sein du VPC.

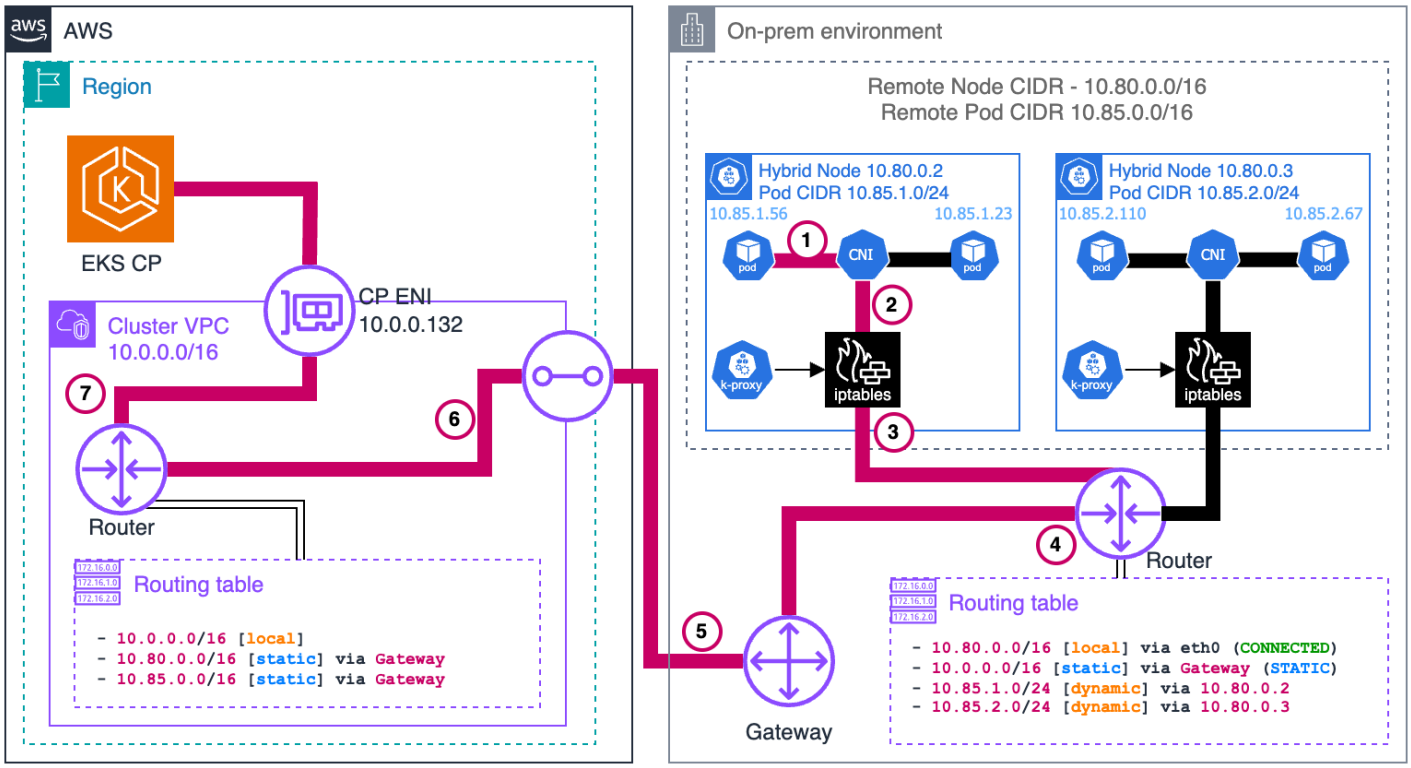
2. Livraison de réseaux VPC

La couche réseau VPC transmet le paquet au sous-réseau avec le plan de contrôle EKS ENI (`10.0.0.132`).

1. Réception ENI

Le paquet atteint le plan de contrôle EKS ENI connecté au serveur API Kubernetes, achevant ainsi son aller-retour.

Pods exécutés sur des nœuds hybrides vers le plan de contrôle EKS



Sans CNI NAT

Demande

Les pods communiquent généralement avec le serveur API Kubernetes via le service `kubernetes`. L'adresse IP du service est la première adresse IP du service CIDR du cluster. Cette convention permet aux pods qui doivent s'exécuter avant que CoreDNS soit disponible d'accéder au serveur API, par exemple le CNI. Les demandes quittent le pod avec l'adresse IP du service comme destination. Par exemple, si le CIDR du service est `172.16.0.0/16`, l'adresse IP du service sera `172.16.0.1`.

1. Le module lance une demande

Le pod envoie une requête à l'adresse IP `kubernetes` du service ()`172.16.0.1` sur le port (443) du serveur API à partir d'un port source aléatoire. Le paquet se présente comme suit :

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.85.1.56 | Src: 67493 (random) |                 |
| Dst: 172.16.0.1 | Dst: 443            |                 |
+-----------------+---------------------+-----------------+
```

2. Traitement CNI

Le CNI détecte que l'adresse IP de destination n'appartient à aucun des pods CIDR qu'il gère. Comme le NAT sortant est désactivé, le CNI transmet le paquet à la pile réseau hôte sans le modifier.

3. Traitement du réseau de nœuds

Le paquet entre dans la pile réseau du nœud où des hooks `netfilter` déclenchent les règles `iptables` définies par kube-proxy. Plusieurs règles s'appliquent dans l'ordre suivant :

1. Le paquet atteint d'abord la chaîne `KUBE-SERVICES`, qui contient les règles correspondant à l'adresse IP du cluster et au port de chaque service.
2. La règle de correspondance passe à la chaîne `KUBE-SVC-XXX` pour le service `kubernetes` (paquets destinés à `172.16.0.1:443`), qui contient les règles d'équilibrage de charge.
3. La règle d'équilibrage de charge sélectionne aléatoirement l'une des `KUBE-SEP-XXX` chaînes pour le plan de commande ENI IPs (`10.0.0.132`or`10.0.1.23`).
4. La chaîne `KUBE-SEP-XXX` sélectionnée possède la règle réelle qui change l'adresse IP de destination de l'adresse IP du service à l'adresse IP sélectionnée. C'est ce qu'on appelle la traduction d'adresses réseau de destination (DNAT).

Une fois ces règles appliquées, en supposant que l'adresse IP de l'ENI du plan de contrôle EKS sélectionné soit `10.0.0.132`, le paquet se présente comme suit :

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.85.1.56 | Src: 67493 (random) |                 |
| Dst: 10.0.0.132 | Dst: 443            |                 |
+-----------------+---------------------+-----------------+
```

Le nœud transmet le paquet à sa passerelle par défaut car l'adresse IP de destination ne se trouve pas sur le réseau local.

4. Routage du routeur local

Le routeur local détermine que l'adresse IP de destination (`10.0.0.132`) appartient au CIDR VPC (`10.0.0.0/16`) et la transfère vers la passerelle connectée à AWS.

5. Transit transfrontalier

Le paquet passe par votre connexion établie (telle que Direct Connect ou VPN) à travers la limite du cloud jusqu'au VPC.

6. Livraison de réseaux VPC

La couche réseau VPC achemine le paquet vers le sous-réseau approprié où se trouve l'ENI (`10.0.0.132`) du plan de contrôle EKS.

7. Réception ENI

Le paquet atteint le plan de contrôle EKS ENI attaché au serveur d'API Kubernetes.

Réponse

Une fois que le plan de contrôle EKS a traité la demande, il renvoie une réponse au pod :

7. Le serveur API envoie une réponse

Le serveur d'API EKS Kubernetes crée un paquet de réponse avec l'adresse IP source d'origine comme destination. Le paquet se présente comme suit :

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.0.0.132 | Src: 443            |                 |
```

```
| Dst: 10.85.1.56 | Dst: 67493          |                 |
+-----------------+---------------------+-----------------+
```

Comme l'adresse IP de destination appartient au pod distant CIDR (`10.85.0.0/16`) configuré, il l'envoie via son ENI dans le VPC avec le routeur du sous-réseau comme saut suivant.

6. Routage VPC

La table de routage VPC contient une entrée pour le pod distant CIDR (`10.85.0.0/16`), qui dirige ce trafic vers la passerelle. VPC-to-onprem

5. Transit transfrontalier

La passerelle transfère le paquet à travers la limite du cloud via votre connexion établie (telle que Direct Connect ou VPN) vers votre réseau sur site.

4. Réception réseau sur site

Le paquet arrive à votre routeur local sur site.

3. Livraison au nœud

La table du routeur contient une entrée pour `10.85.1.0/24` avec `10.80.0.2` comme prochain saut, livrant le paquet à notre nœud.

2. Traitement du réseau de nœuds

Lorsque le paquet est traité par la pile réseau du nœud, `conntrack` (une partie de `netfilter`) correspond au paquet avec la connexion initialement établie par le pod. Depuis l'application initiale du DNAT, `conntrack` inverse le DNAT en réécrivant l'adresse IP source de l'ENI du plan de contrôle EKS vers l'adresse IP `kubernetes` du service :

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 172.16.0.1 | Src: 443            |                 |
| Dst: 10.85.1.56 | Dst: 67493          |                 |
+-----------------+---------------------+-----------------+
```

1. Traitement CNI

Le CNI identifie que l'adresse IP de destination appartient à un pod de son réseau et achemine le paquet vers l'espace de noms réseau du pod approprié.

Ce flux montre pourquoi Remote Pod CIDRs doit être correctement routable depuis le VPC jusqu'au nœud spécifique hébergeant chaque pod. Le chemin de retour complet dépend du routage correct du pod sur les réseaux cloud et IPs sur site.

Avec CNI NAT

Ce flux est très similaire à celui sans CNI NAT, mais avec une différence essentielle : le CNI applique le NAT source (SNAT) au paquet avant de l'envoyer à la pile réseau du nœud. Cela modifie l'adresse IP source du paquet pour la remplacer par l'adresse IP du nœud, ce qui permet au paquet d'être renvoyé vers le nœud sans nécessiter de configuration de routage supplémentaire.

Demande

1. Le module lance une demande

Le pod envoie une requête à l'adresse IP `kubernetes` du service ()`172.16.0.1` sur le port (443) du serveur API EKS Kubernetes à partir d'un port source aléatoire. Le paquet se présente comme suit :

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.85.1.56 | Src: 67493 (random) |                 |
| Dst: 172.16.0.1 | Dst: 443            |                 |
+-----------------+---------------------+-----------------+
```

2. Traitement CNI

Le CNI détecte que l'adresse IP de destination n'appartient à aucun des pods CIDR qu'il gère. Comme le NAT sortant est activé, le CNI applique le SNAT au paquet, en remplaçant l'adresse IP source par celle du nœud avant de le transmettre à la pile réseau du nœud :

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.80.0.2  | Src: 67493 (random) |                 |
| Dst: 172.16.0.1 | Dst: 443            |                 |
+-----------------+---------------------+-----------------+
```

Remarque : CNI et CNI `iptables` sont présentés dans l'exemple sous forme de blocs séparés pour plus de clarté, mais dans la pratique, il est possible que certains CNIs utilisent le NAT `iptables` pour appliquer le NAT.

3. Traitement du réseau de nœuds

Ici, les `iptables` règles définies par `kube-proxy` se comportent de la même manière que dans l'exemple précédent, en équilibrant la charge du paquet sur l'un des plans de contrôle EKS ENIs. Le paquet se présente maintenant comme suit :

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.80.0.2  | Src: 67493 (random) |                 |
| Dst: 10.0.0.132 | Dst: 443            |                 |
+-----------------+---------------------+-----------------+
```

Le nœud transmet le paquet à sa passerelle par défaut car l'adresse IP de destination ne se trouve pas sur le réseau local.

4. Routage du routeur local

Le routeur local détermine que l'adresse IP de destination (`10.0.0.132`) appartient au CIDR VPC (`10.0.0.0/16`) et la transfère vers la passerelle connectée à AWS.

5. Transit transfrontalier

Le paquet passe par votre connexion établie (telle que Direct Connect ou VPN) à travers la limite du cloud jusqu'au VPC.

6. Livraison de réseaux VPC

La couche réseau VPC achemine le paquet vers le sous-réseau approprié où se trouve l'ENI (`10.0.0.132`) du plan de contrôle EKS.

7. Réception ENI

Le paquet atteint le plan de contrôle EKS ENI attaché au serveur d'API Kubernetes.

Réponse

Une fois que le plan de contrôle EKS a traité la demande, il renvoie une réponse au pod :

7. Le serveur API envoie une réponse

Le serveur d'API EKS Kubernetes crée un paquet de réponse avec l'adresse IP source d'origine comme destination. Le paquet se présente comme suit :

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.0.0.132 | Src: 443            |                 |
| Dst: 10.80.0.2  | Dst: 67493          |                 |
+-----------------+---------------------+-----------------+
```

Comme l'adresse IP de destination appartient au nœud distant CIDR (`10.80.0.0/16`) configuré, il l'envoie via son ENI dans le VPC avec le routeur du sous-réseau comme saut suivant.

6. Routage VPC

La table de routage VPC contient une entrée pour le nœud distant CIDR (`10.80.0.0/16`), qui dirige ce trafic vers la passerelle. VPC-to-onprem

5. Transit transfrontalier

La passerelle transfère le paquet à travers la limite du cloud via votre connexion établie (telle que Direct Connect ou VPN) vers votre réseau sur site.

4. Réception réseau sur site

Le paquet arrive à votre routeur local sur site.

3. Livraison au nœud

Le routeur local identifie que l'adresse IP (`10.80.0.2`) de destination appartient à son réseau directement connecté et transfère le paquet directement au nœud hybride cible.

2. Traitement du réseau de nœuds

Lorsque le paquet est traité par la pile réseau du nœud, `conntrack` (une partie de `netfilter`) correspond au paquet avec la connexion initialement établie par le pod et, comme le DNAT a été appliqué à l'origine, il inverse cette opération en réécrivant l'adresse IP source de l'ENI du plan de contrôle EKS vers l'adresse IP du service `kubernetes` :

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 172.16.0.1 | Src: 443            |                 |
| Dst: 10.80.0.2  | Dst: 67493          |                 |
+-----------------+---------------------+-----------------+
```

1. Traitement CNI

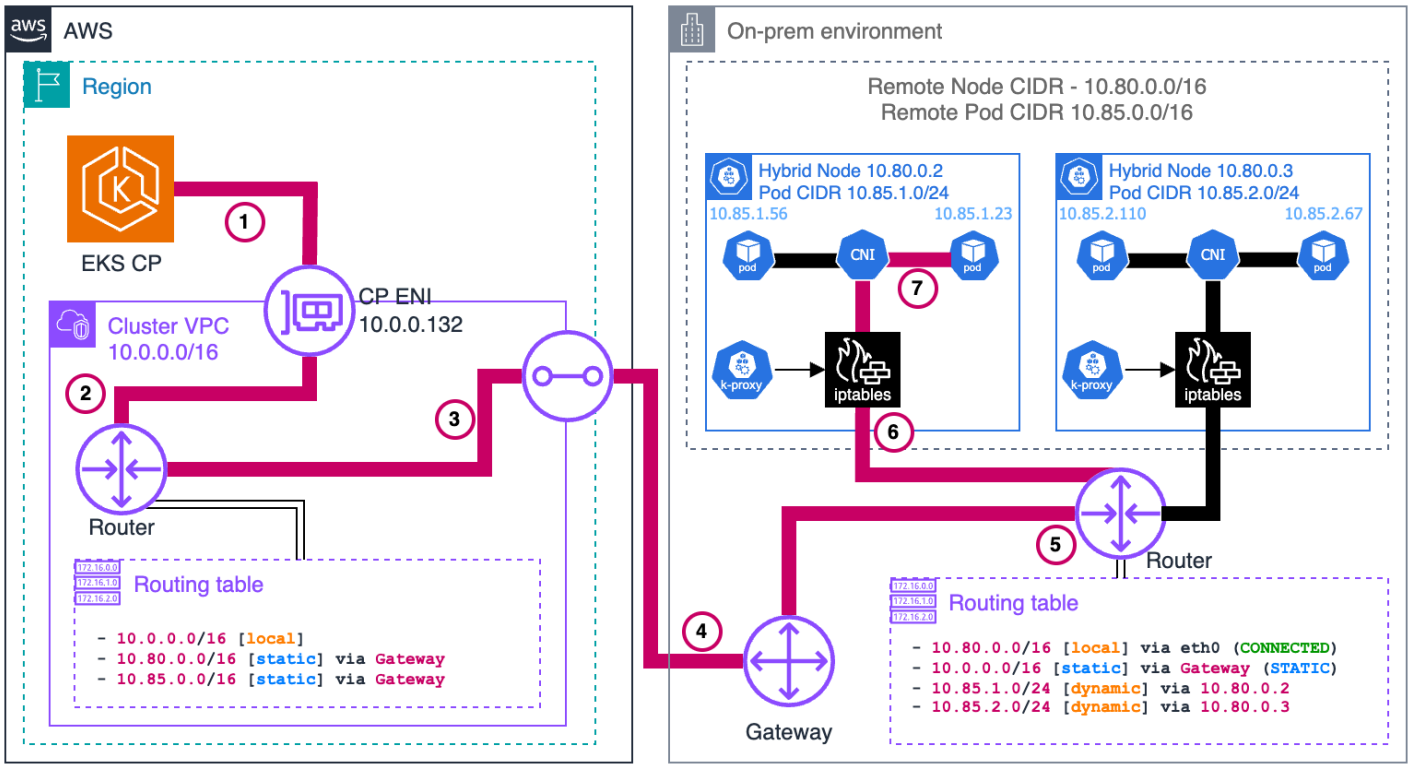
Le CNI identifie que ce paquet appartient à une connexion à laquelle il a précédemment appliqué le SNAT. Il inverse le SNAT, remplaçant l'adresse IP de destination par l'adresse IP du pod :

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 172.16.0.1 | Src: 443            |                 |
| Dst: 10.85.1.56 | Dst: 67493          |                 |
+-----------------+---------------------+-----------------+
```

Le CNI détecte que l'adresse IP de destination appartient à un pod de son réseau et achemine le paquet vers l'espace de noms réseau du pod approprié.

Ce flux montre comment CNI Nat-ing peut simplifier la configuration en permettant le renvoi des paquets vers le nœud sans nécessiter de routage supplémentaire pour le pod. CIDRs

Plan de contrôle EKS pour les pods exécutés sur un nœud hybride (webhooks)



Ce modèle de trafic est le plus souvent observé avec les webhooks, où le plan de contrôle EKS doit initier directement des connexions vers les serveurs webhooks s'exécutant dans des pods sur des nœuds hybrides. Parmi les exemples, citons la validation et la mutation des webhooks

d'admission, qui sont appelés par le serveur API pendant les processus de validation ou de mutation des ressources.

Demande

1. Le serveur d'API EKS Kubernetes lance une demande

Lorsqu'un webhook est configuré dans le cluster et qu'une opération API pertinente le déclenche, le serveur API EKS Kubernetes doit établir une connexion directe avec le pod du serveur webhook. Le serveur API recherche d'abord l'adresse IP du pod à partir de la ressource Service ou Endpoint associée au webhook.

En supposant que le pod webhook s'exécute sur un nœud hybride avec l'adresse IP `10.85.1.23`, le serveur API Kubernetes EKS crée une requête HTTPS vers le point de terminaison webhook. Le paquet initial est envoyé via l'ENI du plan de contrôle EKS dans votre VPC, car l'adresse IP de destination `10.85.1.23` appartient au CIDR du pod distant configuré (`10.85.0.0/16`). Le paquet se présente comme suit :

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.0.0.132 | Src: 41892 (random) |                 |
| Dst: 10.85.1.23 | Dst: 8443           |                 |
+-----------------+---------------------+-----------------+
```

2. Traitement réseau VPC

Le paquet quitte l'ENI du plan de contrôle EKS et entre dans la couche réseau VPC avec le routeur du sous-réseau comme prochain saut.

3. Recherche dans une table de routage VPC

La table de routage VPC pour le sous-réseau contenant l'ENI du plan de contrôle EKS contient une route spécifique pour le CIDR du pod distant (`10.85.0.0/16`). Cette règle de routage dirige le paquet vers la VPC-to-onprem passerelle (par exemple, une passerelle privée virtuelle pour les connexions Direct Connect ou VPN) :

```
Destination     Target
10.0.0.0/16     local
10.85.0.0/16    vgw-id (VPC-to-onprem gateway)
```

4. Transit transfrontalier

La passerelle transfère le paquet à travers la limite du cloud via votre connexion établie (telle que Direct Connect ou VPN) vers votre réseau sur site. Le paquet conserve ses adresses IP source et de destination d'origine lorsqu'il traverse cette connexion.

5. Réception réseau sur site

Le paquet arrive à votre routeur local sur site. Le routeur consulte sa table de routage pour déterminer comment atteindre l'adresse 10.85.1.23. Pour que cela fonctionne, votre réseau local doit disposer de routes pour le pod CIDRs qui dirigent le trafic vers le nœud hybride approprié.

Dans ce cas, la table de routage du routeur contient une entrée indiquant que le sous-réseau `10.85.1.0/24` est accessible via le nœud hybride avec l'adresse IP `10.80.0.2` :

```
Destination     Next Hop
10.85.1.0/24    10.80.0.2
```

6. Livraison au nœud

Sur la base de l'entrée de la table de routage, le routeur transmet le paquet au nœud hybride (`10.80.0.2`). Lorsque le paquet arrive au nœud, il a le même aspect que lorsqu'il a été envoyé par le serveur API EKS Kubernetes, l'adresse IP de destination étant toujours celle du pod.

7. Traitement CNI

La pile réseau du nœud reçoit le paquet et, constatant que l'adresse IP de destination n'est pas celle du nœud, le transmet au CNI pour traitement. Le CNI identifie que l'adresse IP de destination appartient à un pod s'exécutant localement sur ce nœud et transfère le paquet vers le pod correct via les interfaces virtuelles appropriées :

```
Original packet -> node routing -> CNI -> Pod's network namespace
```

Le serveur Webhook du pod reçoit la demande et la traite.

Réponse

Une fois que le pod webhook a traité la requête, il renvoie une réponse en suivant le même chemin en sens inverse :

7. Le pod envoie une réponse

Le pod webhook crée un paquet de réponse avec sa propre adresse IP comme source et le demandeur d'origine (l'ENI du plan de contrôle EKS) comme destination :

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.85.1.23 | Src: 8443           |                 |
| Dst: 10.0.0.132 | Dst: 41892          |                 |
+-----------------+---------------------+-----------------+
```

Le CNI identifie que ce paquet est destiné à un réseau externe (et non à un pod local) et le transmet à la pile réseau du nœud en conservant l'adresse IP source d'origine.

6. Traitement du réseau de nœuds

Le nœud détermine que l'adresse IP de destination (`10.0.0.132`) ne se trouve pas dans le réseau local et transfère le paquet vers sa passerelle par défaut (le routeur local).

5. Routage du routeur local

Le routeur local consulte sa table de routage et détermine que l'adresse IP de destination (`10.0.0.132`) appartient au CIDR VPC (`10.0.0.0/16`). Il transfère le paquet vers la passerelle connectée à AWS.

4. Transit transfrontalier

Le paquet retourne par la même connexion locale vers VPC, traversant la limite du cloud dans le sens inverse.

3. Routage VPC

Lorsque le paquet arrive dans le VPC, les tables de routage identifient que l'adresse IP de destination appartient à un sous-réseau au sein du VPC. Le paquet est acheminé en conséquence au sein du VPC.

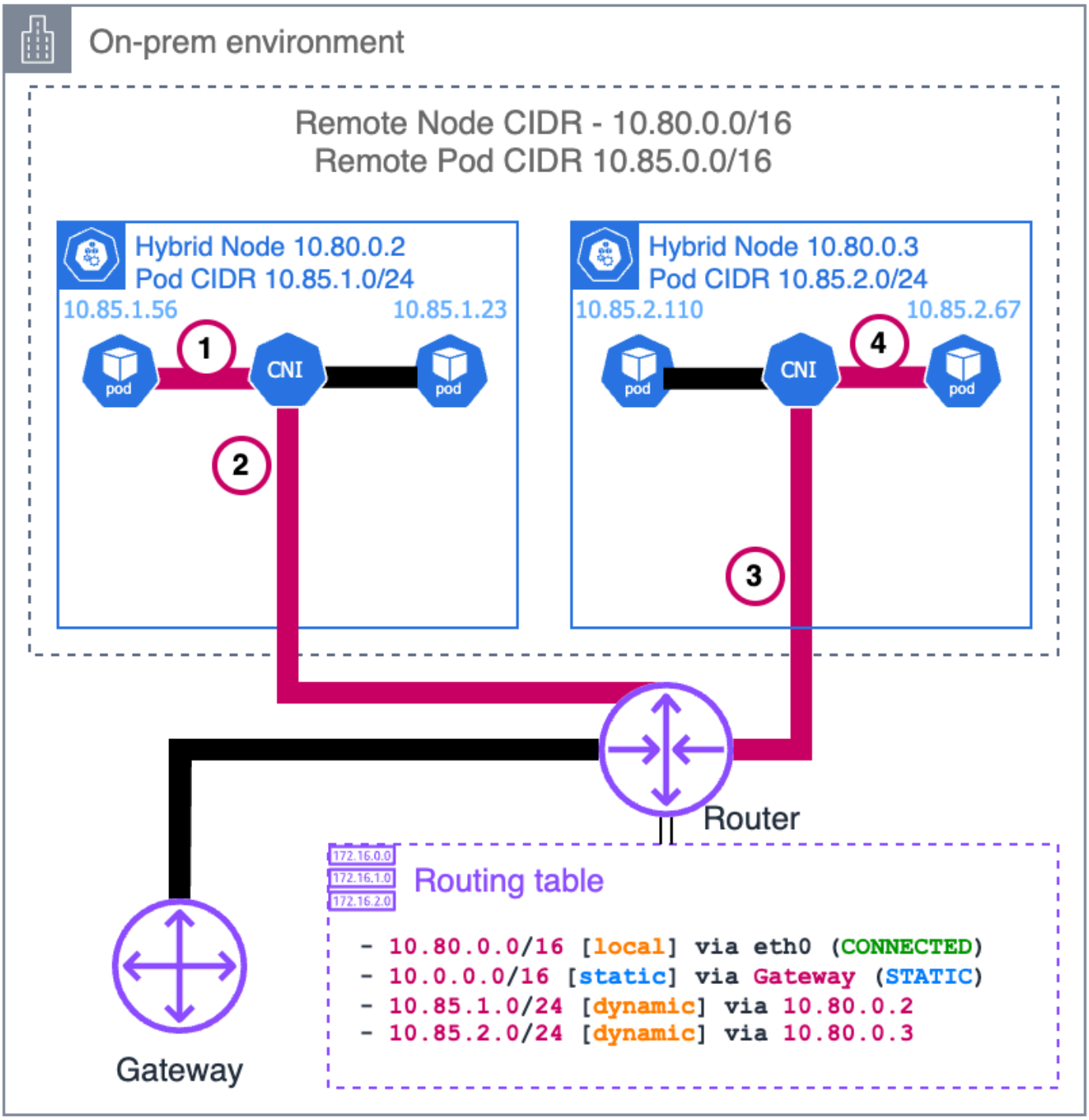
2. et 1. Plan de commande EKS ENI Receiption

Le paquet atteint l'ENI rattaché au serveur API EKS Kubernetes, complétant ainsi l'aller-retour. Le serveur API reçoit la réponse du webhook et continue à traiter la requête API initiale en fonction de cette réponse.

Ce flux de trafic montre pourquoi le module distant CIDRs doit être correctement configuré et routé :

- Le VPC doit disposer de routes pour le pod distant CIDRs pointant vers la passerelle sur site
- Votre réseau sur site doit disposer de routes pour les pods CIDRs qui dirigent le trafic vers les nœuds spécifiques hébergeant ces pods
- Sans cette configuration de routage, les webhooks et autres services similaires s'exécutant dans des pods sur des nœuds hybrides ne seraient pas accessibles depuis le plan de contrôle EKS.

Pod-to-Pod exécution sur des nœuds hybrides



Cette section explique comment les pods s'exécutant sur différents nœuds hybrides communiquent entre eux. Cet exemple suppose que votre CNI utilise VXLAN pour l'encapsulation, ce qui est

courant pour Cilium CNIs ou Calico. Le processus global est similaire pour d'autres protocoles d'encapsulation tels que Geneve ou. IP-in-IP

Demande

1. Le pod A initie la communication

Le Pod A (`10.85.1.56`) sur le Nœud 1 souhaite envoyer du trafic vers le Pod B (`10.85.2.67`) sur le Nœud 2. Le paquet initial ressemble à ceci :

```
+------------------+-----------------+-------------+-----------------+
| Ethernet Header  | IP Header       | TCP/UDP     | Payload         |
| Src: Pod A MAC   | Src: 10.85.1.56 | Src: 43721  |                 |
| Dst: Gateway MAC | Dst: 10.85.2.67 | Dst: 8080   |                 |
+------------------+-----------------+-------------+-----------------+
```

2. CNI intercepte et traite le paquet

Lorsque le paquet du pod A quitte son espace de noms réseau, le CNI l'intercepte. Le CNI consulte sa table de routage et détermine : – L'adresse IP de destination (`10.85.2.67`) appartient au CIDR du pod – Cette adresse IP ne se trouve pas sur le nœud local, mais appartient au nœud 2 (`10.80.0.3`) – Le paquet doit être encapsulé avec VXLAN.

La décision d'encapsuler est essentielle car le réseau physique sous-jacent ne sait pas comment acheminer CIDRs directement le pod. Il sait uniquement comment acheminer le trafic entre les nœuds IPs.

Le CNI encapsule l'intégralité du paquet d'origine dans une trame VXLAN. Cela crée effectivement un « paquet dans un paquet » avec de nouveaux en-têtes :

```
+-----------------+----------------+-------------+-----------
+--------------------------+
| Outer Ethernet  | Outer IP       | Outer UDP    | VXLAN      | Original Pod-to-Pod
    |
| Src: Node1 MAC  | Src: 10.80.0.2 | Src: Random  | VNI: 42    | Packet (unchanged
    |
| Dst: Router MAC | Dst: 10.80.0.3 | Dst: 8472    |            | from above)
    |
+-----------------+----------------+-------------+-----------
+--------------------------+
```

Points clés concernant cette encapsulation : – Le paquet externe est adressé du nœud 1 (`10.80.0.2`) au nœud 2 (`10.80.0.3`) – Le port UDP `8472` est le port VXLAN utilisé par défaut par Cilium – L'identifiant de réseau VXLAN (VNI) identifie le réseau superposé auquel appartient ce paquet – Le paquet d'origine complet (avec l'adresse IP du pod A comme source et l'adresse IP du pod B comme destination) est conservé intact à l'intérieur

Le paquet encapsulé entre alors dans la pile réseau normale du nœud 1 et est traité de la même manière que n'importe quel autre paquet :

1. Traitement du réseau de nœuds : la pile réseau du nœud 1 achemine le paquet en fonction de sa destination (`10.80.0.3`)
2. Livraison sur le réseau local :
   - Si les deux nœuds sont sur le même réseau de couche 2, le paquet est envoyé directement au nœud 2
   - S'ils se trouvent sur des sous-réseaux différents, le paquet est d'abord transféré au routeur local
3. Gestion du routeur : le routeur transmet le paquet en fonction de sa table de routage et le livre au nœud 2

3. Traitement des nœuds de réception

Lorsque le paquet encapsulé arrive au nœud 2 (`10.80.0.3`) :

1. La pile réseau du nœud le reçoit et l'identifie comme un paquet VXLAN (port UDP `4789`)
2. Le paquet est transmis à l'interface VXLAN du CNI pour traitement

4. Décapsulation VXLAN

Le CNI sur le nœud 2 traite le paquet VXLAN :

1. Il supprime les en-têtes extérieurs (Ethernet, IP, UDP et VXLAN)
2. Il extrait le paquet intérieur d'origine
3. Le paquet a maintenant retrouvé sa forme d'origine :

```
+------------------+-----------------+-------------+-----------------+
| Ethernet Header  | IP Header       | TCP/UDP     | Payload         |
| Src: Pod A MAC   | Src: 10.85.1.56 | Src: 43721  |                 |
| Dst: Gateway MAC | Dst: 10.85.2.67 | Dst: 8080   |                 |
```

```
+-----------------+----------------+------------+----------------+
```

Le CNI du nœud 2 examine l'adresse IP de destination (`10.85.2.67`) et :

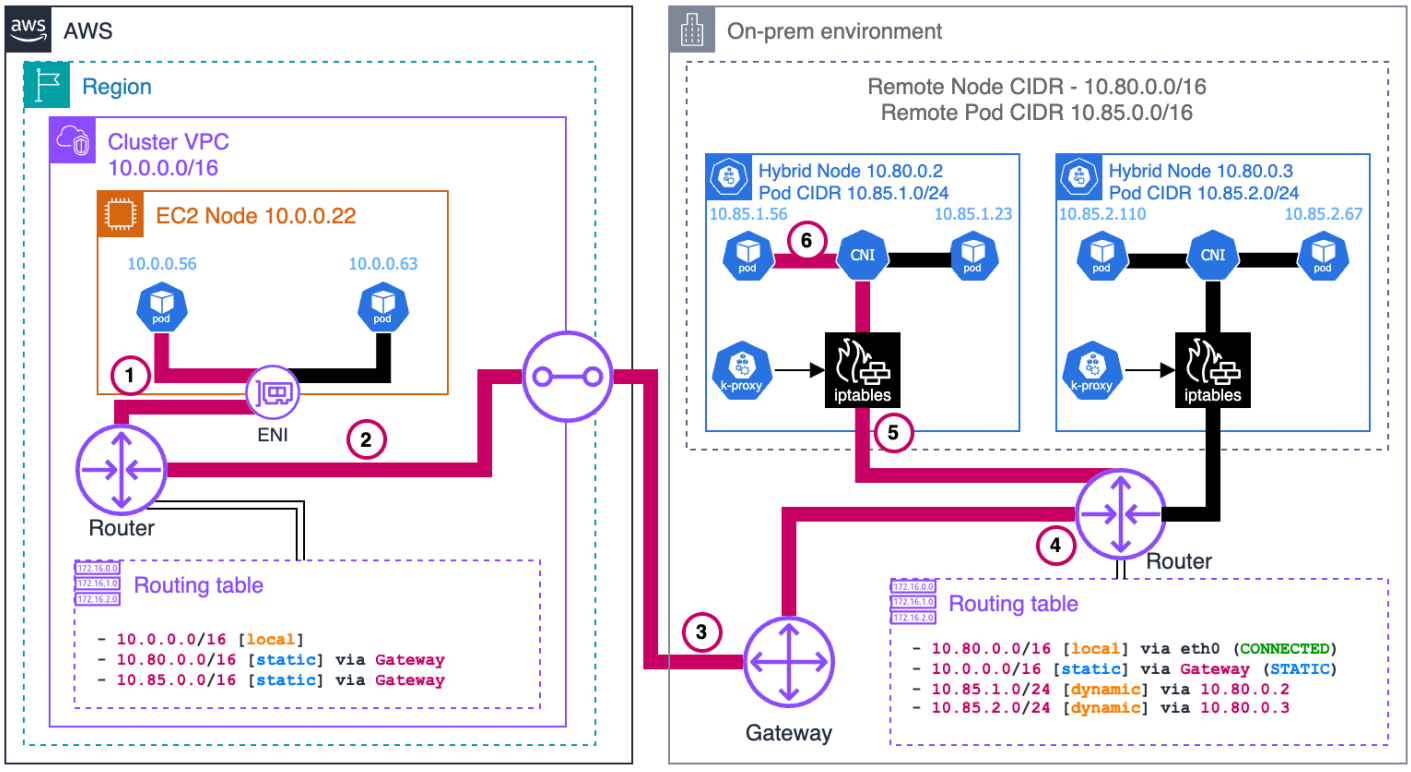
1. Identifie que cette adresse IP appartient à un pod local
2. Achemine le paquet via les interfaces virtuelles appropriées
3. Fournit le paquet à l'espace de noms réseau du Pod B

Réponse

Lorsque le Pod B répond au Pod A, l'ensemble du processus se déroule en sens inverse :

4. Le Pod B envoie un paquet au Pod A (`10.85.1.56`)
5. Le CNI du nœud 2 l'encapsule avec VXLAN, en définissant la destination sur le nœud 1 (`10.80.0.2`)
6. Le paquet encapsulé est livré au nœud 1
7. Le CNI du nœud 1 le décapsule et fournit la réponse initiale au module A

Des pods sur des nœuds cloud vers des pods sur des nœuds hybrides (trafic est-ouest)



Demande

1. Le module A initie la communication

Le pod A (`10.0.0.56`) sur le EC2 nœud souhaite envoyer du trafic vers le pod B (`10.85.1.56`) sur le nœud hybride. Le paquet initial ressemble à ceci :

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.0.0.56  | Src: 52390 (random) |                 |
| Dst: 10.85.1.56 | Dst: 8080           |                 |
+-----------------+---------------------+-----------------+
```

Avec le VPC CNI, le Pod A possède une adresse IP provenant du VPC CIDR et est directement attaché à un ENI sur l'instance. EC2 L'espace de noms réseau du pod est connecté au réseau VPC, de sorte que le paquet entre directement dans l'infrastructure de routage VPC.

2. Routage VPC

La table de routage VPC contient une route spécifique pour le Remote Pod CIDR (`10.85.0.0/16`), dirigeant ce trafic vers la passerelle : VPC-to-onprem

```
Destination     Target
10.0.0.0/16     local
10.85.0.0/16    vgw-id (VPC-to-onprem gateway)
```

Sur la base de cette règle de routage, le paquet est dirigé vers la passerelle qui se connecte à votre réseau sur site.

3. Transit transfrontalier

La passerelle transfère le paquet à travers la limite du cloud via votre connexion établie (telle que Direct Connect ou VPN) vers votre réseau sur site. Le paquet conserve ses adresses IP source et destination d'origine tout au long de ce transit.

4. Réception réseau sur site

Le paquet arrive à votre routeur local sur site. Le routeur consulte sa table de routage pour déterminer le prochain saut permettant d'atteindre l'adresse 10.85.1.56. Votre routeur local doit disposer de routes pour le pod CIDRs qui dirigent le trafic vers le nœud hybride approprié.

La table du routeur contient une entrée indiquant que le sous-réseau `10.85.1.0/24` est accessible via le nœud hybride avec l'adresse IP `10.80.0.2` :

```
Destination     Next Hop
10.85.1.0/24    10.80.0.2
```

5. Traitement du réseau de nœuds

Le routeur transmet le paquet au nœud hybride (`10.80.0.2`). Lorsque le paquet arrive au nœud, il a toujours l'adresse IP du pod A comme source et celle du pod B comme destination.

6. Traitement CNI

La pile réseau du nœud reçoit le paquet et, constatant que l'adresse IP de destination n'est pas la sienne, le transmet au CNI pour traitement. Le CNI identifie que l'adresse IP de destination appartient à un pod s'exécutant localement sur ce nœud et transfère le paquet vers le pod correct via les interfaces virtuelles appropriées :

```
Original packet -> node routing -> CNI -> Pod B's network namespace
```

Le pod B reçoit le paquet et le traite selon les besoins.

Réponse

6. Le pod B envoie une réponse

Le pod B crée un paquet de réponse avec sa propre adresse IP comme source et celle du pod A comme destination :

```
+-----------------+---------------------+-----------------+
| IP Header       | TCP Header          | Payload         |
| Src: 10.85.1.56 | Src: 8080           |                 |
| Dst: 10.0.0.56  | Dst: 52390          |                 |
+-----------------+---------------------+-----------------+
```

Le CNI identifie que ce paquet est destiné à un réseau externe et le transmet à la pile réseau du nœud.

5. Traitement du réseau de nœuds

Le nœud détermine que l'adresse IP de destination (`10.0.0.56`) n'appartient pas au réseau local et transfère le paquet vers sa passerelle par défaut (le routeur local).

4. Routage du routeur local

Le routeur local consulte sa table de routage et détermine que l'adresse IP de destination (`10.0.0.56`) appartient au CIDR VPC (`10.0.0.0/16`). Il transfère le paquet vers la passerelle connectée à AWS.

3. Transit transfrontalier

Le paquet retourne par la même connexion locale vers VPC, traversant la limite du cloud dans le sens inverse.

2. Routage VPC

Lorsque le paquet arrive dans le VPC, le système de routage identifie que l'adresse IP de destination appartient à un sous-réseau au sein du VPC. Le paquet est acheminé via le réseau VPC vers l'instance hébergeant EC2 le Pod A.

1. Le pod A reçoit une réponse

Le paquet arrive à l' EC2 instance et est livré directement au Pod A via son ENI attaché. Étant donné que le CNI VPC n'utilise pas de réseau superposé pour les pods dans le VPC, aucune décapsulation supplémentaire n'est nécessaire : le paquet arrive avec ses en-têtes d'origine intacts.

Ce flux de trafic est-ouest montre pourquoi le module distant CIDRs doit être correctement configuré et routable dans les deux sens :

- Le VPC doit disposer de routes pour le pod distant CIDRs pointant vers la passerelle sur site
- Votre réseau local doit disposer de routes pour les pods CIDRs qui dirigent le trafic vers les nœuds spécifiques hébergeant ces pods.

Référence des nœuds hybrides **`nodeadm`**

L'interface CLI nœuds hybrides Amazon EKS (`nodeadm`) simplifie l'installation, la configuration, l'enregistrement et la désinstallation des composants des nœuds hybrides. Vous pouvez inclure `nodeadm` dans votre système d'exploitation des images permettant d'automatiser le démarrage des nœuds hybrides. Pour plus d'informations, consultez the section called "Préparer le système d'exploitation".

La `nodeadm` version pour les nœuds hybrides est différente de la `nodeadm` version utilisée pour démarrer les EC2 instances Amazon en tant que nœuds dans des clusters Amazon EKS. Suivez

la documentation et les références correspondant à la version nodeadm appropriée. Cette page de documentation concerne la version nodeadm des nœuds hybrides.

Le code source des nœuds hybrides nodeadm est publié dans le référentiel https://github.com/aws/eks-hybridGitHub .

> **Important**
>
> Vous devez exécuter nodeadm avec un utilisateur qui possède root/sudo des privilèges.

Télécharger **nodeadm**

La version des nœuds hybrides de nodeadm est hébergée sur Amazon S3, géré par Amazon CloudFront. Pour installer nodeadm sur chaque hôte sur site, vous pouvez exécuter la commande suivante à partir de vos hôtes sur site.

Pour les hôtes x86_64

```
curl -OL 'https://hybrid-assets.eks.amazonaws.com/releases/latest/bin/linux/amd64/nodeadm'
```

Pour les hôtes ARM

```
curl -OL 'https://hybrid-assets.eks.amazonaws.com/releases/latest/bin/linux/arm64/nodeadm'
```

Ajoutez les permissions d'exécution au fichier binaire téléchargé sur chaque hôte.

```
chmod +x nodeadm
```

`nodeadm install`

La commande `nodeadm install` permet d'installer les artefacts et les dépendances nécessaires pour exécuter et joindre des nœuds hybrides à un cluster Amazon EKS. La commande `nodeadm install` peut être exécutée individuellement sur chaque nœud hybride ou pendant les pipelines de création d'images afin de préinstaller les dépendances des nœuds hybrides dans les images du système d'exploitation.

Utilisation

```
nodeadm install [KUBERNETES_VERSION] [flags]
```

Arguments positionnels

(Obligatoire) `KUBERNETES_VERSION` La version majeure.mineure d'EKS Kubernetes à installer, par exemple `1.32`

Indicateurs

Name	Obligatoire	Description
`-p`, `--credential-provider`	TRUE	Fournisseur d'informations d'identification à installer. Les valeurs prises en charge sont `iam-ra` et `ssm`. Pour plus d'informations, consultez the section called "Préparation des informations d'identification".
`-s`, `--containerd-source`	FALSE	Source pour `containerd` . `nodeadm` prend en charge l'installation de `containerd` à partir de la distribution du système d'exploitation, des paquets Docker et le saute l'installation de `containerd` . Valeurs `distro`- Il s'agit de la valeur par défaut. `nodeadm`installer a le dernier `containerd` package distribué par le système d'exploitation du nœud compatible avec la version EKS Kubernetes.

Name	Obligatoire	Description
		`distro`n'est pas une valeur prise en charge pour les systèmes d'exploitation Red Hat Enterprise Linux (RHEL). `docker`- `nodeadm` installer a le dernier `containerd` package créé et distribué par Docker compatible avec la version EKS Kubernetes. `dockern`'est pas une valeur prise en charge pour Amazon Linux 2023. `none` : `nodeadm` n'installera pas le paquet `containerd` . Vous devez installer `containerd` manuellement avant d'exécuter `nodeadm init`.
`-r`, `--region`	FALSE	Spécifie la AWS région pour le téléchargement d'artefacts tels que l'agent SSM. La valeur par défaut est `us-west-2` .
`-t`, `--timeout`	FALSE	Durée maximale de la commande d'installation. La saisie suit le format de durée. Par exemple `1h23m`. Le délai d'expiration par défaut pour le téléchargement de la commande d'installation est fixé à 20 minutes.

Name	Obligatoire	Description
`-h, --help`	FALSE	Affiche un message d'aide avec les paramètres disponibles pour les métriques, les sous-commandes et les valeurs positionnelles.

Exemples

Installez la version Kubernetes avec AWS Systems `1.32` Manager (SSM) comme fournisseur d'informations d'identification

```
nodeadm install 1.32 --credential-provider ssm
```

Installez la version Kubernetes avec AWS Systems `1.32` Manager (SSM) comme fournisseur d'informations d'identification, Docker comme source containerd, avec un délai de téléchargement de 20 minutes.

```
nodeadm install 1.32 --credential-provider ssm --containerd-source docker --timeout 20m
```

Installez la version Kubernetes `1.32` avec AWS IAM Roles Anywhere comme fournisseur d'informations d'identification

```
nodeadm install 1.32 --credential-provider iam-ra
```

`nodeadm config check`

La commande `nodeadm config check` vérifie la configuration du nœud fourni afin de détecter d'éventuelles erreurs. Cette commande peut être utilisée pour vérifier et valider l'exactitude d'un fichier de configuration de nœud hybride.

Utilisation

```
nodeadm config check [flags]
```

Indicateurs

Name	Obligatoire	Description
`-c`, `--config-source`	TRUE	Source de la configuration nodeadm. Pour les nœuds hybrides, l'entrée doit suivre un URI avec un schéma de fichier.
`-h`, `--help`	FALSE	Affiche un message d'aide avec les paramètres disponibl es pour les métriques, les sous-commandes et les valeurs positionnelles.

Exemples

```
nodeadm config check -c file://nodeConfig.yaml
```

`nodeadm init`

La commande `nodeadm init` démarre et connecte le nœud hybride au cluster Amazon EKS configuré. Consultez the section called "Configuration des nœuds pour les activations hybrides SSM" ou the section called "Configuration des nœuds pour Rôles Anywhere IAM" pour plus de détails sur la configuration du fichier `nodeConfig.yaml`.

Utilisation

```
nodeadm init [flags]
```

Indicateurs

Name	Obligatoire	Description
`-c`, `--config-source`	TRUE	Source de la configuration `nodeadm`. Pour les nœuds hybrides, l'entrée doit suivre

Name	Obligatoire	Description
		un URI avec un schéma de fichier.
`-s`, `--skip`	FALSE	Phases de `init` à ignorer. Il n'est pas recommandé de sauter l'une des phases, sauf si cela permet de résoudre un problème. Valeurs `install-validation` ignore la vérification du bon déroulement de la commande d'installation précédente. `cni-validation` ignore la vérification de l'ouverture des ports VXLAN de Cilium ou Calico CNI si le pare-feu est activé sur le nœud `node-ip-validation` ignore la vérification si l'adresse IP du nœud se trouve dans un CIDR dans les réseaux de nœuds distants
`-h`, `--help`	FALSE	Affiche un message d'aide avec les paramètres disponibl es pour les métriques, les sous-commandes et les valeurs positionnelles.

Exemples

```
nodeadm init -c file://nodeConfig.yaml
```

nodeadm upgrade

La commande `nodeadm upgrade` met à niveau tous les artefacts installés vers la dernière version et amorce le nœud pour configurer les artefacts mis à niveau et rejoindre le cluster EKS sur AWS. La mise à niveau est une commande perturbatrice pour les charges de travail exécutées sur le nœud. Veuillez déplacer vos charges de travail vers un autre nœud avant de lancer la mise à niveau.

Utilisation

```
nodeadm upgrade [KUBERNETES_VERSION] [flags]
```

Arguments positionnels

(Obligatoire) `KUBERNETES_VERSION` La version majeure.mineure d'EKS Kubernetes à installer, par exemple `1.32`

Indicateurs

Name	Obligatoire	Description
`-c`, `--config-source`	TRUE	Source de la configuration `nodeadm`. Pour les nœuds hybrides, l'entrée doit suivre un URI avec un schéma de fichier.
`-t`, `--timeout`	FALSE	Délai d'attente pour le téléchargement des artefacts . La saisie suit le format de durée. Par exemple 1 h 23 min. Le délai d'expiration par défaut pour le téléchargement de la commande de mise à niveau est défini sur 10 minutes.

Name	Obligatoire	Description
`-s`, `--skip`	FALSE	Phases de mise à niveau à ignorer. Il n'est pas recommandé de sauter l'une des phases, sauf si cela permet de résoudre un problème. Valeurs `pod-validation` ignore la vérification si tous les pods ne sont pas en cours d'exécution sur le nœud, à l'exception des ensembles de démons et des pods statiques. `node-validation` ignore la vérification si le nœud a été bouclé. `init-validation` ignore la vérification de l'initialisation réussie du nœud avant d'exécuter la mise à niveau. `containerd-major-version-upgrade` empêche les mises à niveau des versions majeures de containerd lors de la mise à niveau du nœud.

Name	Obligatoire	Description
`-h, --help`	FALSE	Affiche un message d'aide avec les paramètres disponibl es pour les métriques, les sous-commandes et les valeurs positionnelles.

Exemples

```
nodeadm upgrade 1.32 -c file://nodeConfig.yaml
```

```
nodeadm upgrade 1.32 -c file://nodeConfig.yaml --timeout 20m
```

`nodeadm uninstall`

La commande `nodeadm uninstall` arrête et supprime les artefacts que `nodeadm` installe pendant `nodeadm install`, y compris kubelet et containerd. Remarque : La commande de désinstallation ne vide pas et ne supprime pas vos nœuds hybrides de votre cluster. Vous devez exécuter les opérations de vidage et de suppression séparément. Pour plus d'informations, consultez the section called "Supprimer des nœuds hybrides". Par défaut, `nodeadm uninstall` ne continuera pas s'il reste des pods sur le nœud. De même, `nodeadm uninstall` ne supprime pas les dépendances CNI ni les dépendances d'autres modules complémentaires Kubernetes que vous exécutez sur votre cluster. Pour supprimer complètement l'installation CNI de votre hôte, consultez les instructions à l'adresse the section called "Configurer CNI". Si vous utilisez des activations hybrides AWS SSM comme fournisseur d'informations d'identification sur site, la `nodeadm uninstall` commande annule l'enregistrement de vos hôtes en tant qu'instances gérées par SSM. AWS

Utilisation

```
nodeadm uninstall [flags]
```

Indicateurs

Name	Obligatoire	Description
`-s`, `--skip`	FALSE	Phases de désinstallation à ignorer. Il n'est pas recommandé de sauter l'une des phases, sauf si cela permet de résoudre un problème. Valeurs `pod-validation` ignore la vérification si tous les pods ne sont pas en cours d'exécution sur le nœud, à l'exception des ensembles de démons et des pods statiques. `node-validation` ignore la vérification si le nœud a été bouclé. `init-validation` ignore la vérification de l'initialisation réussie du nœud avant d'exécuter la désinstallation.
`-h`, `--help`	FALSE	Affiche un message d'aide avec les paramètres disponibl es pour les métriques, les sous-commandes et les valeurs positionnelles.
`-f`, `--force`	FALSE	Supprimer de force les répertoires supplémentaires qui pourraient contenir des fichiers restants provenant des

Name	Obligatoire	Description
		composants Kubernetes et CNI. WARNING Cela supprimera tout le contenu des répertoires Kubernetes et CNI par défaut (`/var/lib/cni` , `/etc/cni/net.d` , etc.). N'utilise z pas cet indicateur si vous stockez vos propres données à ces emplacements. À partir de nodeadm `v1.0.9`, la commande `./nodeadm uninstall --skip node-validation,pod-validation --force` ne supprime plus le répertoir e `/var/lib/kubelet` . En effet, il peut contenir des volumes Pod et des répertoires volume-subpath qui incluent parfois le système de fichiers du nœud monté. Conseils pour une manipulat ion en toute sécurité – La suppression des chemins montés peut entraîner la suppression accidentelle du système de fichiers du nœud monté. Avant de supprimer manuellement le répertoir

Name	Obligatoire	Description
		e `/var/lib/kubelet` , inspectez soigneusement tous les montages actifs et démontez les volumes en toute sécurité afin d'éviter toute perte de données.

Exemples

```
nodeadm uninstall
```

```
nodeadm uninstall --skip node-validation,pod-validation
```

`nodeadm debug`

La commande `nodeadm debug` peut être utilisée pour dépanner les nœuds hybrides défectueux ou mal configurés. Il vérifie que les exigences suivantes sont respectées.

- Le nœud dispose d'un accès réseau aux informations requises AWS APIs pour obtenir les informations d'identification,
- Le nœud est en mesure d'obtenir des AWS informations d'identification pour le rôle IAM des nœuds hybrides configuré,
- Le nœud dispose d'un accès réseau au point de terminaison de l'API EKS Kubernetes et la validité du certificat du point de terminaison de l'API EKS Kubernetes,
- Le nœud est capable de s'authentifier auprès du cluster EKS, son identité dans le cluster est valide et le nœud a accès au cluster EKS via le VPC configuré pour le cluster EKS.

Si des erreurs sont détectées, la sortie de la commande suggère des étapes de dépannage. Certaines étapes de validation affichent les processus enfants. Si celles-ci échouent, le résultat s'affiche dans une section stderr sous l'erreur de validation.

Utilisation

```
nodeadm debug [flags]
```

Indicateurs

Name	Obligatoire	Description
`-c`, `--config-source`	TRUE	Source de la configuration `nodeadm`. Pour les nœuds hybrides, l'entrée doit suivre un URI avec un schéma de fichier.
`--no-color`	FALSE	Désactive la sortie couleur. Utile pour l'automatisation.
`-h`, `--help`	FALSE	Affiche un message d'aide avec les paramètres disponibl es pour les métriques, les sous-commandes et les valeurs positionnelles.

Exemples

```
nodeadm debug -c file://nodeConfig.yaml
```

Emplacements des fichiers Nodeadm

nodeadm installer

Lors de l'exécution `nodeadm install`, les fichiers et emplacements de fichiers suivants sont configurés.

Artefact	Chemin
CLI Rôles Anywhere IAM	/usr/local/bin/aws_assistant à la signature
Kubelet binaire	/usr/bin/kubelet
Kubectl binaire	usr/local/bin/kubectl

Artefact	Chemin
Fournisseur d'informations d'identification ECR	/etc/eks/image-credential-provider/ecr-fourni sseur d'informations d'identification
AWS Authentificateur IAM	/usr/local/bin/aws-iam-authentificateur
Configuration SSM CLI	/opt/ssm/ssm-setup-cli
SSM Agent	Sur Ubuntu -/snap/amazon-ssm-agent/current/ amazon-ssm-agent Sur RHEL et AL2 023 -/-ssm-agent usr/bin/a mazon
Containerd	Sur Ubuntu et AL2 023 -/usr/bin/containerd Sur RHEL : /bin/containerd
iptables	Sur Ubuntu et AL2 023 -/usr/sbin/iptables Sur RHEL : /sbin/iptables
Plug-ins CNI	/opt/cni/bin
suivi des artefacts installés	/opt/nodeadm/tracker

nodeadm init

Lors de l'exécution de `nodeadm init`, les fichiers et emplacements de fichiers suivants sont configurés.

Name	Chemin
Kubelet kubeconfig	/var/lib/kubelet/kubeconfig
Configuration Kubelet	/etc/kubernetes/kubelet/config.json
Unité de système Kubelet	/etc/systemd/system/kubelet.service

Name	Chemin
Configuration du fournisseur d'informations d'identification d'image	/etc/eks/image-credential-provider/config.json
Fichier d'environnement Kubelet	/etc/eks/kubelet/environment
Certificats Kubelet	/etc/kubernetes/pki/ca.crt
Config de Containerd	/etc/containerd/config.toml
Configuration des modules du noyau Containerd	/etc/modules-load.d/contianerd.conf
AWS fichier de configuration	/etc/aws/hybrid/config
AWS fichier d'informations d'identification (si le fichier d'informations d'identification est activé)	/eks-hybrid/.aws/identifiants
AWS unité système d'aide à la signature	/etc/systemd/system/aws_signing_helper_update.service
fichier de configuration sysctl	/etc/sysctl.d/99-nodeadm.conf
Ca-certificates	/etc/ssl/certs/ca-certificates.crt
Fichier de clé Gpg	/etc/apt/keyrings/docker.asc
Fichier source du dépôt Docker	/etc/apt/sources.list.d/docker.liste

Configuration des nœuds pour les activations hybrides SSM

Voici un exemple `nodeConfig.yaml` d'utilisation des activations hybrides AWS SSM pour les informations d'identification des nœuds hybrides.

```
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
    name:             # Name of the EKS cluster
```

```
    region:           # AWS Region where the EKS cluster resides
  hybrid:
    ssm:
      activationCode: # SSM hybrid activation code
      activationId:   # SSM hybrid activation id
```

Configuration des nœuds pour Rôles Anywhere IAM

Voici un `nodeConfig.yaml` exemple d'identifiant AWS IAM Roles Anywhere pour les nœuds hybrides.

Lorsque vous utilisez AWS IAM Roles Anywhere comme fournisseur d'informations d'identification sur site, celles que `nodeName` vous utilisez dans votre `nodeadm` configuration doivent correspondre aux autorisations que vous avez définies pour votre rôle IAM Hybrid Nodes. Par exemple, si vos autorisations pour le rôle IAM de Hybrid Nodes autorisent uniquement AWS IAM Roles Anywhere à assumer le rôle lorsque le nom de session du rôle est égal au CN du certificat hôte, le `nodeName` CN de votre `nodeadm` configuration doit être le même que le CN de vos certificats. Le `nodeName` que vous utilisez ne peut pas dépasser 64 caractères. Pour de plus amples informations, veuillez consulter the section called “Préparation des informations d'identification”.

```
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
    name:              # Name of the EKS cluster
    region:            # AWS Region where the EKS cluster resides
  hybrid:
    iamRolesAnywhere:
      nodeName:        # Name of the node
      trustAnchorArn:  # ARN of the IAM Roles Anywhere trust anchor
      profileArn:      # ARN of the IAM Roles Anywhere profile
      roleArn:         # ARN of the Hybrid Nodes IAM role
      certificatePath: # Path to the certificate file to authenticate with the IAM
 Roles Anywhere trust anchor
      privateKeyPath:  # Path to the private key file for the certificate
```

Configuration du nœud pour personnaliser kubelet (facultatif)

Vous pouvez transmettre la configuration et les métriques kubelet dans votre configuration `nodeadm`. Consultez l'exemple ci-dessous pour savoir comment ajouter une étiquette de nœud

`abc.amazonaws.com/test-label` supplémentaire et configurer `shutdownGracePeriod` sur 30 secondes.

```
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
    name:             # Name of the EKS cluster
    region:           # AWS Region where the EKS cluster resides
  kubelet:
    config:           # Map of kubelet config and values
       shutdownGracePeriod: 30s
    flags:            # List of kubelet flags
       - --node-labels=abc.company.com/test-label=true
  hybrid:
    ssm:
      activationCode: # SSM hybrid activation code
      activationId:   # SSM hybrid activation id
```

Configuration du nœud pour personnaliser containerd (facultatif)

Vous pouvez passer une configuration containerd personnalisée dans votre configuration `nodeadm`. La configuration containerd pour `nodeadm` accepte TOML en ligne. Consultez l'exemple ci-dessous pour savoir comment configurer containerd afin de désactiver la suppression des couches d'image décompressées dans le magasin de contenu containerd.

```
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
    name:             # Name of the EKS cluster
    region:           # AWS Region where the EKS cluster resides
  containerd:
    config: |         # Inline TOML containerd additional configuration
       [plugins."io.containerd.grpc.v1.cri".containerd]
       discard_unpacked_layers = false
  hybrid:
    ssm:
      activationCode: # SSM hybrid activation code
      activationId:   # SSM hybrid activation id
```

> **Note**
>
> Les versions 1.x et 2.x de Containerd utilisent différents formats de configuration. Containerd 1.x utilise la version de configuration 2, tandis que containerd 2.x utilise la version de configuration 3. Bien que containerd 2.x reste rétrocompatible avec la version 2 de configuration, la version 3 de configuration est recommandée pour des performances optimales. Vérifiez la version de votre containerd `containerd --version` ou consultez les journaux d'`nodeadm`installation. Pour plus de détails sur la gestion des versions de configuration, consultez https://containerd.io/releases/

Vous pouvez également utiliser la configuration containerd pour activer SELinux le support. SELinux Lorsque cette option est activée sur containerd, assurez-vous que les pods planifiés sur le nœud disposent du SecurityContext approprié et sont activés. seLinuxOptions Vous trouverez plus d'informations sur la configuration d'un contexte de sécurité dans la documentation Kubernetes.

> **Note**
>
> Red Hat Enterprise Linux (RHEL) 8 et RHEL 9 sont SELinux activés par défaut et définis sur strict sur l'hôte. Amazon Linux 2023 est SELinux activé par défaut et est réglé sur le mode permissif. Lorsqu'il SELinux est défini sur le mode permissif sur l'hôte, son activation sur containerd ne bloquera pas les demandes mais les enregistrera conformément à la SELinux configuration de l'hôte.

```
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
    name:             # Name of the EKS cluster
    region:           # AWS Region where the EKS cluster resides
  containerd:
    config: |         # Inline TOML containerd additional configuration
       [plugins."io.containerd.grpc.v1.cri"]
       enable_selinux = true
  hybrid:
    ssm:
      activationCode: # SSM hybrid activation code
      activationId:   # SSM hybrid activation id
```

Dépannage des nœuds hybrides

Cette rubrique traite de certaines erreurs courantes que vous pouvez rencontrer lors de l'utilisation des nœuds hybrides Amazon EKS et explique comment les résoudre. Pour d'autres informations de dépannage, consultez le Dépannage tag Knowledge Center pour Amazon EKS sur AWS Re:Post. Si vous ne parvenez pas à résoudre le problème, contactez AWS le Support.

Dépannage des nœuds avec **`nodeadm debug`**. Vous pouvez exécuter la commande `nodeadm debug` à partir de vos nœuds hybrides pour vérifier que les exigences en matière de réseau et d'informations d'identification sont respectées. Pour plus d'informations sur la commande `nodeadm debug`, voir the section called "Nœuds hybrides nodeadm".

Détectez les problèmes liés à vos nœuds hybrides grâce aux informations sur les clusters Les informations sur les clusters Amazon EKS comprennent des vérifications qui détectent les problèmes courants liés à la configuration des nœuds hybrides EKS dans votre cluster. Vous pouvez consulter les résultats de toutes les vérifications d'aperçu à partir de la AWS Management Console AWS CLI et du AWS SDKs. Pour plus d'informations sur les informations relatives aux clusters, consultez the section called "Informations sur le cluster".

Dépannage de l'installation de nœuds hybrides

Les rubriques de dépannage suivantes concernent l'installation des dépendances des nœuds hybrides sur les hôtes à l'aide de la commande `nodeadm install`.

`nodeadm` échec de la commande **`must run as root`**

La commande `nodeadm install` doit être exécutée avec un utilisateur disposant de droits root ou de `sudo` privilèges sur votre hôte. Si vous exécutez `nodeadm install` avec un utilisateur qui ne dispose pas des droits root ou des privilèges `sudo`, vous verrez l'erreur suivante dans la sortie `nodeadm`.

```
"msg":"Command failed","error":"must run as root"
```

Impossible de se connecter aux dépendances

La commande `nodeadm install` installe les dépendances requises pour les nœuds hybrides. Les dépendances des nœuds hybrides incluent`containerd`, `kubelet``kubectl`, et les composants AWS SSM ou AWS IAM Roles Anywhere. Vous devez avoir accès à partir de l'endroit où vous exécutez `nodeadm install` pour télécharger ces dépendances. Pour plus d'informations sur la liste des emplacements auxquels vous devez pouvoir accéder, consultez the section called "Préparez

votre réseau". Si vous n'y avez pas accès, vous verrez des erreurs similaires à celles-ci dans la sortie `nodeadm install`.

```
"msg":"Command failed","error":"failed reading file from url: ...: max retries achieved
 for http request"
```

Impossible de mettre à jour le gestionnaire de paquets

La commande `nodeadm install` exécute `apt update`, `yum update` ou `dnf update` avant l'installation des dépendances des nœuds hybrides. Si cette étape échoue, vous pourriez voir apparaître des erreurs similaires à celles ci-dessous. Pour remédier à cette situation, vous pouvez exécuter `apt update`, `yum update` ou `dnf update` avant l'exécution de `nodeadm install` ou essayer de réexécuter `nodeadm install`.

```
failed to run update using package manager
```

Délai d'expiration ou délai de mise en contexte dépassé

Lors de l'exécution de `nodeadm install`, si vous constatez des problèmes à différentes étapes du processus d'installation avec des erreurs indiquant un délai d'attente ou une limite de contexte dépassée, il se peut que votre connexion soit trop lente et empêche l'installation des dépendances des nœuds hybrides avant l'expiration des délais d'attente. Pour contourner ces problèmes, vous pouvez essayer d'utiliser la balise `--timeout` dans `nodeadm` afin de prolonger la durée des délais d'attente pour le téléchargement des dépendances.

```
nodeadm install K8S_VERSION --credential-provider CREDS_PROVIDER --timeout 20m0s
```

Dépannage des problèmes liés à la connexion

Les rubriques de résolution des problèmes de cette section concernent le processus de connexion des nœuds hybrides aux clusters EKS à l'aide de la commande `nodeadm init`.

Erreurs de fonctionnement ou schéma non pris en charge

Lors de l'exécution `nodeadm init`, si vous voyez des erreurs liées à `operation error` ou `unsupported scheme`, vérifiez votre `nodeConfig.yaml` pour vous assurer qu'il est correctement formaté et transmis à `nodeadm`. Pour plus d'informations sur le format et les options disponibles pour `nodeConfig.yaml`, consultez the section called "Nœuds hybrides nodeadm".

```
"msg":"Command failed","error":"operation error ec2imds: GetRegion, request canceled,
 context deadline exceeded"
```

Nœuds hybrides (rôle IAM) : autorisations manquantes pour l'action **`eks:DescribeCluster`**

Lors de l'exécution`nodeadm init`, `nodeadm` tente de recueillir des informations sur votre cluster EKS en appelant l'`DescribeCluster`action EKS. Si votre rôle IAM Hybrid Nodes n'est pas autorisé à effectuer cette `eks:DescribeCluster` action, vous devez transmettre le point de terminaison de l'API Kubernetes, le bundle CA du cluster et le IPv4 CIDR du service dans la configuration du nœud à laquelle vous passez lors de l'exécution. `nodeadm` `nodeadm init` Pour plus d'informations sur les autorisations requises pour le rôle IAM des nœuds hybrides, consultez the section called "Préparation des informations d'identification".

```
"msg":"Command failed","error":"operation error EKS: DescribeCluster, https response
 error StatusCode: 403 ... AccessDeniedException"
```

Nœuds hybrides (rôle IAM) : autorisations manquantes pour l'action **`eks:ListAccessEntries`**

Lors de l'exécution`nodeadm init`, `nodeadm` tente de valider si votre cluster EKS possède une entrée d'accès de type `HYBRID_LINUX` associée au rôle IAM des nœuds hybrides en appelant l'`ListAccessEntries`action EKS. Si votre rôle IAM Hybrid Nodes n'est pas autorisé à `eks:ListAccessEntries` effectuer cette action, vous devez passer le `--skip cluster-access-validation` drapeau lorsque vous exécutez la `nodeadm init` commande. Pour plus d'informations sur les autorisations requises pour le rôle IAM des nœuds hybrides, consultez the section called "Préparation des informations d'identification".

```
"msg":"Command failed","error":"operation error EKS: ListAccessEntries, https response
 error StatusCode: 403 ... AccessDeniedException"
```

L'adresse IP du nœud ne figure pas dans le réseau de nœuds distants CIDR

Lors de l'exécution`nodeadm init`, vous pouvez rencontrer une erreur si l'adresse IP du nœud ne se trouve pas dans le réseau de nœuds distants spécifié CIDRs. L'erreur ressemblera à l'exemple suivant :

```
node IP 10.18.0.1 is not in any of the remote network CIDR blocks [10.0.0.0/16
 192.168.0.0/16]
```

Cet exemple montre un nœud dont l'adresse IP est 10.18.0.1 qui tente de rejoindre un cluster doté du réseau distant CIDRs 10.0.0.0/16 et 192.168.0.0/16. L'erreur se produit car 10.18.0.1 ne se situe dans aucune des plages.

Vérifiez que vous avez correctement configuré votre adresse IP `RemoteNodeNetworks` pour inclure toutes les adresses IP des nœuds. Pour plus d'informations sur la configuration réseau, voir the section called "Préparez votre réseau".

- Exécutez la commande suivante dans la région où se trouve votre cluster pour vérifier vos configurations `RemoteNodeNetwork`. Vérifiez que les blocs CIDR répertoriés dans la sortie incluent la plage IP de votre nœud et correspondent aux blocs CIDR répertoriés dans le message d'erreur. Si elles ne correspondent pas, vérifiez que le nom du cluster et la région dans votre `nodeConfig.yaml` correspondent bien au cluster souhaité.

```
aws eks describe-cluster --name CLUSTER_NAME --region REGION_NAME --query
 cluster.remoteNetworkConfig.remoteNodeNetworks
```

- Vérifiez que vous travaillez avec le nœud prévu :
  - Vérifiez que vous êtes sur le bon nœud en vérifiant que son nom d'hôte et son adresse IP correspondent à ceux que vous souhaitez enregistrer auprès du cluster.
  - Vérifiez que ce nœud se trouve dans le bon réseau sur site (celui dont la plage CIDR a été enregistrée en tant que `RemoteNodeNetwork` lors de la configuration du cluster).

Si l'adresse IP de votre nœud ne correspond toujours pas à ce que vous attendiez, vérifiez les points suivants :

- Si vous utilisez Rôles Anywhere IAM, `kubelet` effectue une recherche DNS sur Rôles Anywhere IAM `nodeName` et utilise une adresse IP associée au nom du nœud, le cas échéant. Si vous conservez des entrées DNS pour vos nœuds, vérifiez qu'elles pointent vers l' IPs intérieur de votre réseau de nœuds distants CIDRs.
- Si votre nœud possède plusieurs interfaces réseau, vous `kubelet` pouvez sélectionner une interface avec une adresse IP en dehors du réseau CIDRs de votre nœud distant par défaut. Pour utiliser une interface différente, spécifiez son adresse IP à l'aide de la balise `--node-ip` `kubelet` dans votre fichier `nodeConfig.yaml`. Pour de plus amples informations, veuillez consulter the section called "Nœuds hybrides nodeadm". Vous pouvez consulter les interfaces réseau et les adresses IP de votre nœud en exécutant la commande suivante sur votre nœud :

```
ip addr show
```

Les nœuds hybrides n'apparaissent pas dans le cluster EKS

Si vous avez exécuté `nodeadm init` et qu'il s'est terminé, mais que vos nœuds hybrides n'apparaissent pas dans votre cluster, il se peut qu'il y ait des problèmes avec la connexion réseau entre vos nœuds hybrides et le plan de contrôle EKS, que vous n'ayez pas configuré les autorisations requises pour le groupe de sécurité ou que vous n'ayez pas effectué le mappage requis entre le rôle IAM de vos nœuds hybrides et le contrôle d'accès basé sur les rôles (RBAC) de Kubernetes. Vous pouvez lancer le processus de débogage en vérifiant l'état de `kubelet` et des journaux `kubelet` à l'aide des commandes suivantes. Exécutez les commandes suivantes à partir des nœuds hybrides qui n'ont pas réussi à rejoindre votre cluster.

```
systemctl status kubelet
```

```
journalctl -u kubelet -f
```

Impossible de communiquer avec le cluster

Si votre nœud hybride n'a pas pu communiquer avec le plan de contrôle du cluster, des journaux similaires aux suivants peuvent s'afficher.

```
"Failed to ensure lease exists, will retry" err="Get ..."
```

```
"Unable to register node with API server" err="Post ..."
```

```
Failed to contact API server when waiting for CSINode publishing ... dial tcp <ip
 address>: i/o timeout
```

Si vous voyez ces messages, vérifiez les points suivants pour vous assurer qu'ils répondent aux exigences relatives aux nœuds hybrides détaillées dans the section called "Préparez votre réseau".

- Vérifiez que le VPC transmis au cluster EKS possède des itinéraires vers votre Transit Gateway (TGW) ou votre passerelle privée virtuelle (VGW) pour votre nœud sur site et éventuellement votre pod. CIDRs
- Vérifiez que vous disposez d'un groupe de sécurité supplémentaire pour votre cluster EKS et que des règles d'entrée s'appliquent à votre nœud local CIDRs et, éventuellement, à votre pod. CIDRs

- Vérifiez que votre routeur sur site est configuré pour autoriser le trafic vers et depuis le plan de contrôle EKS.

Non autorisé

Si votre nœud hybride a pu communiquer avec le plan de contrôle EKS mais n'a pas pu s'enregistrer, vous pouvez voir des journaux similaires à ceux ci-dessous. Notez que la principale différence dans les messages du journal ci-dessous réside dans l'erreur `Unauthorized`. Cela indique que le nœud n'a pas pu exécuter ses tâches car il ne dispose pas des autorisations requises.

```
"Failed to ensure lease exists, will retry" err="Unauthorized"
```

```
"Unable to register node with API server" err="Unauthorized"
```

```
Failed to contact API server when waiting for CSINode publishing: Unauthorized
```

Si vous voyez ces messages, vérifiez les éléments suivants pour vous assurer qu'ils répondent aux exigences relatives aux nœuds hybrides détaillées dans the section called "Préparation des informations d'identification" et the section called "Préparation de l'accès aux clusters".

- Vérifiez que l'identité des nœuds hybrides correspond au rôle IAM des nœuds hybrides que vous attendiez. Pour ce faire, exécutez `sudo aws sts get-caller-identity` depuis vos nœuds hybrides.
- Vérifiez que votre rôle IAM des nœuds hybrides dispose des autorisations requises.
- Vérifiez que dans votre cluster, vous disposez d'une entrée d'accès EKS pour votre rôle IAM Hybrid Nodes ou confirmez que vous disposez d'`aws-auth` ConfigMap une entrée pour votre rôle IAM Hybrid Nodes. Si vous utilisez des entrées d'accès EKS, vérifiez que votre entrée d'accès pour votre rôle IAM des nœuds hybrides dispose du type d'accès `HYBRID_LINUX`. Si vous utilisez le `aws-auth` ConfigMap, confirmez que votre entrée pour le rôle Hybrid Nodes IAM répond aux exigences et au format détaillés dansthe section called "Préparation de l'accès aux clusters".

Nœuds hybrides enregistrés dans le cluster EKS mais affichant l'état **`Not Ready`**

Si vos nœuds hybrides se sont correctement enregistrés auprès de votre cluster EKS, mais qu'ils affichent l'état `Not Ready`, la première chose à vérifier est l'état de votre interface réseau de conteneur (CNI). Si vous n'avez pas installé de CNI, vos nœuds hybrides devraient avoir le statut

`Not Ready`. Une fois qu'un CNI est installé et fonctionne correctement, les nœuds sont mis à jour vers le statut `Ready`. Si vous avez essayé d'installer un CNI mais qu'il ne fonctionne pas correctement, consultez the section called "Dépannage des nœuds hybrides CNI" sur cette page.

Les demandes de signature de certificat (CSRs) sont bloquées en attente

Après avoir connecté des nœuds hybrides à votre cluster EKS, si vous constatez que CSRs des nœuds hybrides sont en attente, cela signifie que ceux-ci ne répondent pas aux exigences d'approbation automatique. CSRs pour les nœuds hybrides sont automatiquement approuvés et émis si les CSRs nœuds hybrides ont été créés par un nœud portant une `eks.amazonaws.com/compute-type: hybrid` étiquette, et que le CSR possède les noms alternatifs de sujet suivants (SANs) : au moins un SAN DNS égal au nom du nœud et l'adresse IP SANs appartient au réseau de nœuds distants CIDRs.

Le profil hybride existe déjà

Si vous avez modifié votre configuration `nodeadm` et que vous essayez de réenregistrer le nœud avec la nouvelle configuration, vous pouvez obtenir une erreur indiquant que le profil hybride existe déjà, mais que son contenu a été modifié. Au lieu d'exécuter `nodeadm init` entre les changements de configuration, exécutez `nodeadm uninstall` suivi d'un `nodeadm install` et `nodeadm init`. Cela garantit un nettoyage correct lors des changements de configuration.

```
"msg":"Command failed","error":"hybrid profile already exists at /etc/aws/hybrid/config
 but its contents do not align with the expected configuration"
```

Le nœud hybride n'a pas réussi à résoudre l'API privée

Après l'exécution `nodeadm init`, si vous constatez une erreur dans les journaux `kubelet` indiquant des échecs de connexion au serveur API EKS Kubernetes en raison de la présence de `no such host`, vous devrez peut-être modifier votre entrée DNS pour le point de terminaison API EKS Kubernetes dans votre réseau sur site ou au niveau de l'hôte. Consultez la section Transfert de requêtes DNS entrantes vers votre VPC dans AWS la documentation de Route53.

```
Failed to contact API server when waiting for CSINode publishing: Get ... no such host
```

Impossible d'afficher les nœuds hybrides dans la console EKS

Si vous avez enregistré vos nœuds hybrides mais que vous ne parvenez pas à les afficher dans la console EKS, vérifiez les autorisations du principal IAM que vous utilisez pour afficher la console. Le principal IAM que vous utilisez doit disposer d'autorisations IAM et Kubernetes minimales spécifiques

pour afficher les ressources dans la console. Pour de plus amples informations, veuillez consulter the section called "Accéder aux ressources du cluster".

Résolution des problèmes d'exécution de nœuds hybrides

Si vos nœuds hybrides enregistrés dans votre cluster EKS avaient le statut `Ready`, puis sont passés au statut `Not Ready`, plusieurs problèmes peuvent avoir contribué à ce statut non fonctionnel, par exemple un manque de ressources CPU, de mémoire ou d'espace disque disponible, ou une déconnexion du nœud du plan de contrôle EKS. Vous pouvez suivre les étapes ci-dessous pour dépanner vos nœuds, et si vous ne parvenez pas à résoudre le problème, contactez le AWS Support.

Exécutez `nodeadm debug` à partir de vos nœuds hybrides pour vérifier que les exigences en matière de réseau et d'informations d'identification sont satisfaites. Pour plus d'informations sur la commande `nodeadm debug` , consultez the section called "Nœuds hybrides nodeadm".

Obtenir le statut du nœud

```
kubectl get nodes -o wide
```

Vérifiez les conditions et les événements du nœud

```
kubectl describe node NODE_NAME
```

Obtenir le statut du pod

```
kubectl get pods -A -o wide
```

Vérifiez l'état et les événements du pod

```
kubectl describe pod POD_NAME
```

Vérifiez les journaux du pod

```
kubectl logs POD_NAME
```

Vérifiez les journaux **`kubectl`**

```
systemctl status kubelet
```

```
journalctl -u kubelet -f
```

Les sondes de durée de vie du pod échouent ou les webhooks ne fonctionnent pas

Si les applications, les modules complémentaires ou les webhooks exécutés sur vos nœuds hybrides ne démarrent pas correctement, vous rencontrez peut-être des problèmes de réseau qui bloquent la communication avec les pods. Pour que le plan de contrôle EKS puisse contacter les webhooks exécutés sur des nœuds hybrides, vous devez configurer votre cluster EKS avec un réseau de pods distant et disposer de routes pour votre CIDR de pod sur site dans votre table de routage VPC, avec comme cible votre Transit Gateway (TGW), votre passerelle privée virtuelle (VPW) ou toute autre passerelle que vous utilisez pour connecter votre VPC à votre réseau sur site. Pour plus d'informations sur les exigences réseau pour les nœuds hybrides, voir the section called "Préparez votre réseau". Vous devez également autoriser ce trafic dans votre pare-feu local et vous assurer que votre routeur peut acheminer correctement le trafic vers vos pods. Pour plus d'informations sur les conditions requises pour exécuter des webhooks sur des nœuds hybrides, consultez the section called "Configuration de webhooks".

Un message de journal de pod courant pour ce scénario est présenté ci-dessous, où l'adresse IP correspond à l'adresse IP du cluster pour le service Kubernetes.

```
dial tcp <ip-address>:443: connect: no route to host
```

`kubectl logs`ou **`kubectl exec`** les commandes ne fonctionnent pas (commandes **`kubelet`** API)

Si`kubectl attach`,`kubectl cp`, `kubectl exec`kubectl `logs`, et `kubectl port-forward` les commandes expirent alors que `kubectl` les autres commandes réussissent, le problème est probablement lié à la configuration du réseau distant. Ces commandes se connectent via le cluster au point de terminaison `kubelet` sur le nœud. Pour de plus amples informations, veuillez consulter the section called "Point de terminaison `kubelet`".

Vérifiez que votre nœud IPs et votre pod font IPs partie du réseau de nœuds distants et du réseau de nœuds distants CIDRs configurés pour votre cluster. Utilisez les commandes ci-dessous pour examiner les attributions d'adresses IP.

```
kubectl get nodes -o wide
```

```
kubectl get pods -A -o wide
```

IPs Comparez-les avec votre réseau distant configuré CIDRs pour garantir un routage correct. Pour les exigences de configuration réseau, voir the section called "Préparez votre réseau".

Dépannage des nœuds hybrides CNI

Si vous rencontrez des problèmes lors du démarrage initial de Cilium ou Calico avec des nœuds hybrides, cela est le plus souvent dû à des problèmes de réseau entre les nœuds hybrides ou les pods CNI s'exécutant sur les nœuds hybrides et le plan de contrôle EKS. Assurez-vous que votre environnement répond aux exigences de la section Préparer le réseau pour les nœuds hybrides. Il est utile de décomposer le problème en plusieurs parties.

Configuration du cluster EKS

- Les RemotePodNetwork configurations RemoteNodeNetwork et sont-elles correctes ?

Configuration VPC

- Existe-t-il des itinéraires pour RemoteNodeNetwork et RemotePodNetwork dans la table de routage VPC avec la cible du Transit Gateway ou de la Virtual Private Gateway ?

Configuration du groupe de sécurité

- Existe-t-il des règles d'entrée et de sortie pour le RemoteNodeNetwork et ? RemotePodNetwork

Réseau local

- Existe-t-il des routes et des accès vers et depuis le plan de contrôle EKS, ainsi que vers et depuis les nœuds hybrides et les pods fonctionnant sur des nœuds hybrides ?

Configuration CNI

- Si vous utilisez un réseau superposé, la configuration du pool IP pour le CNI correspond-elle à celle RemotePodNetwork configurée pour le cluster EKS si vous utilisez des webhooks ?

Le nœud hybride a un statut **`Ready`** sans CNI installé

Si vos nœuds hybrides affichent le statut `Ready`, mais que vous n'avez pas installé de CNI sur votre cluster, il est possible que d'anciens artefacts CNI soient présents sur vos nœuds hybrides. Par défaut, lorsque vous désinstallez Cilium et Calico à l'aide d'outils tels que Helm, les ressources sur

disque ne sont pas supprimées de vos machines physiques ou virtuelles. En outre, les définitions de ressources personnalisées (CRDs) correspondantes CNIs peuvent toujours être présentes sur votre cluster à partir d'une ancienne installation. Pour plus d'informations, consultez les sections Supprimer Cilium et Supprimer Calico de the section called "Configurer CNI".

Résolution des problèmes liés à Cilium

Si vous rencontrez des problèmes lors de l'exécution de Cilium sur des nœuds hybrides, consultez les étapes de dépannage dans la documentation Cilium. Les sections ci-dessous traitent des problèmes qui peuvent être propres au déploiement de Cilium sur des nœuds hybrides.

Cilium ne démarre pas

Si les agents Cilium qui s'exécutent sur chaque nœud hybride ne démarrent pas, vérifiez les journaux des pods de l'agent Cilium pour détecter d'éventuelles erreurs. L'agent Cilium nécessite une connexion au point de terminaison API EKS Kubernetes pour démarrer. Le démarrage de l'agent Cilium échouera si cette connectivité n'est pas correctement configurée. Dans ce cas, vous verrez des messages de journal similaires aux suivants dans les journaux du pod de l'agent Cilium.

```
msg="Unable to contact k8s api-server"
level=fatal msg="failed to start: Get \"https://<k8s-cluster-ip>:443/api/v1/namespaces/
kube-system\": dial tcp <k8s-cluster-ip>:443: i/o timeout"
```

L'agent Cilium s'exécute sur le réseau hôte. Votre cluster EKS doit être configuré avec `RemoteNodeNetwork` pour la connectivité Cilium. Vérifiez que vous disposez d'un groupe de sécurité supplémentaire pour votre cluster EKS avec une règle entrante pour votre `RemoteNodeNetwork`, que vous disposez de routes dans votre VPC pour votre `RemoteNodeNetwork`, et que votre réseau sur site est correctement configuré pour permettre la connectivité au plan de contrôle EKS.

Si l'opérateur Cilium est en cours d'exécution et que certains de vos agents Cilium sont en cours d'exécution, mais pas tous, vérifiez que vous disposez d'un pod disponible IPs à allouer pour tous les nœuds de votre cluster. Vous configurez la taille de votre pod allouable CIDRs lorsque vous utilisez le pool de clusters IPAM `clusterPoolIPv4PodCIDRList` dans votre configuration Cilium. La taille CIDR par nœud est configurée à l'aide du paramètre `clusterPoolIPv4MaskSize` dans votre configuration Cilium. Voir Extension du pool de clusters dans la documentation de Cilium pour plus d'informations.

Cilium BGP ne fonctionne pas

Si vous utilisez le plan de contrôle Cilium BGP pour annoncer les adresses de vos pods ou services à votre réseau sur site, vous pouvez utiliser les commandes CLI Cilium suivantes pour vérifier si BGP annonce les routes vers vos ressources. Pour connaître les étapes d'installation de la CLI Cilium, consultez la section Installer la CLI Cilium dans la documentation de Cilium.

Si BGP fonctionne correctement, vous devez indiquer l'état de session de vos nœuds hybrides `established` dans la sortie. Vous devrez peut-être collaborer avec votre équipe réseau afin d'identifier les valeurs correctes pour l'AS local, l'AS pair et l'adresse pair de votre environnement.

```
cilium bgp peers
```

```
cilium bgp routes
```

Si vous utilisez Cilium BGP pour faire IPs de la publicité pour des Services par type`LoadBalancer`, vous devez avoir la même étiquette sur votre produit `CiliumLoadBalancerIPPool` et sur le Service, qui doit être utilisée dans le sélecteur de votre. `CiliumBGPAdvertisement` Un exemple est illustré ci-dessous. Notez que si vous utilisez Cilium BGP pour promouvoir les services avec type LoadBalancer, les IPs routes BGP peuvent être perturbées lors du redémarrage de l'agent Cilium. Pour plus d'informations, consultez la section Scénarios de défaillance dans la documentation Cilium.

Service

```
kind: Service
apiVersion: v1
metadata:
  name: guestbook
  labels:
    app: guestbook
spec:
  ports:
  - port: 3000
    targetPort: http-server
  selector:
    app: guestbook
  type: LoadBalancer
```

CiliumLoadBalancerIPPool

```
apiVersion: cilium.io/v2alpha1
```

```
kind: CiliumLoadBalancerIPPool
metadata:
  name: guestbook-pool
  labels:
    app: guestbook
spec:
  blocks:
  - cidr: <CIDR to advertise>
  serviceSelector:
    matchExpressions:
      - { key: app, operator: In, values: [ guestbook ] }
```

CiliumBGPAdvertisement

```
apiVersion: cilium.io/v2alpha1
kind: CiliumBGPAdvertisement
metadata:
  name: bgp-advertisements-guestbook
  labels:
    advertise: bgp
spec:
  advertisements:
    - advertisementType: "Service"
      service:
        addresses:
          - ExternalIP
          - LoadBalancerIP
      selector:
        matchExpressions:
          - { key: app, operator: In, values: [ guestbook ] }
```

Résolution des problèmes liés à Calico

Si vous rencontrez des problèmes lors de l'exécution de Calico sur des nœuds hybrides, consultez les étapes de dépannage dans la documentation Calico. Les sections ci-dessous traitent des problèmes qui peuvent être propres au déploiement de Calico sur des nœuds hybrides.

Le tableau ci-dessous récapitule les composants Calico et indique s'ils s'exécutent par défaut sur le réseau du nœud ou du pod. Si vous avez configuré Calico pour utiliser NAT pour le trafic sortant des pods, votre réseau sur site doit être configuré pour acheminer le trafic vers votre CIDR de nœud sur site et vos tables de routage VPC doivent être configurées avec une route pour votre CIDR de nœud sur site avec votre passerelle de transit (TGW) ou votre passerelle privée virtuelle (VGW) comme

cible. Si vous ne configurez pas Calico pour utiliser NAT pour le trafic sortant des pods, votre réseau sur site doit être configuré pour acheminer le trafic vers votre CIDR de pod sur site et vos tables de routage VPC doivent être configurées avec une route pour votre CIDR de pod sur site avec votre passerelle de transit (TGW) ou votre passerelle privée virtuelle (VGW) comme cible.

Composant	Réseau
Serveur API Calico	Nœud
Contrôleurs Calico pour Kubernetes	pod
Agent de nœud Calico	Nœud
Calico `typha`	Nœud
Pilote de nœud Calico CSI	pod
Opérateur Calico	Nœud

Les ressources Calico sont planifiées ou en cours d'exécution sur des nœuds cordonnés

Les ressources Calico qui ne s'exécutent pas en tant que telles DaemonSet sont soumises à des tolérances flexibles par défaut qui leur permettent d'être planifiées sur des nœuds cordonés qui ne sont pas prêts à être planifiés ou à exécuter des pods. Vous pouvez renforcer les tolérances pour les ressources autres que DaemonSet Calico en modifiant l'installation de votre opérateur pour inclure les éléments suivants.

```
installation:
  ...
  controlPlaneTolerations:
  - effect: NoExecute
    key: node.kubernetes.io/unreachable
    operator: Exists
    tolerationSeconds: 300
  - effect: NoExecute
    key: node.kubernetes.io/not-ready
    operator: Exists
    tolerationSeconds: 300
  calicoKubeControllersDeployment:
    spec:
```

```
      template:
        spec:
          tolerations:
          - effect: NoExecute
            key: node.kubernetes.io/unreachable
            operator: Exists
            tolerationSeconds: 300
          - effect: NoExecute
            key: node.kubernetes.io/not-ready
            operator: Exists
            tolerationSeconds: 300
  typhaDeployment:
    spec:
      template:
        spec:
          tolerations:
          - effect: NoExecute
            key: node.kubernetes.io/unreachable
            operator: Exists
            tolerationSeconds: 300
          - effect: NoExecute
            key: node.kubernetes.io/not-ready
            operator: Exists
            tolerationSeconds: 300
```

Dépannage des informations d'identification

Pour les activations hybrides AWS SSM et pour AWS IAM Roles Anywhere, vous pouvez vérifier que les informations d'identification du rôle Hybrid Nodes IAM sont correctement configurées sur vos nœuds hybrides en exécutant la commande suivante depuis vos nœuds hybrides. Vérifiez que le nom du nœud et le nom du rôle IAM des nœuds hybrides correspondent à vos attentes.

```
sudo aws sts get-caller-identity
```

```
{
    "UserId": "ABCDEFGHIJKLM12345678910:<node-name>",
    "Account": "<aws-account-id>",
    "Arn": "arn:aws: sts::<aws-account-id>:assumed-role/<hybrid-nodes-iam-role/<node-
name>"
}
```

Dépannage du gestionnaire de systèmes (SSM) AWS

Si vous utilisez des activations hybrides AWS SSM pour les informations d'identification de vos nœuds hybrides, tenez compte des répertoires et artefacts SSM suivants qui sont installés sur vos nœuds hybrides par. `nodeadm` Pour plus d'informations sur l'agent SSM, consultez la section Utilisation de l'agent SSM dans le Guide de l'utilisateur de AWS Systems Manager.

Description	Location
Agent SSM	Ubuntu - `/snap/amazon-ssm-agent/current/amazon-ssm-agent` RHEL et AL2 023 - `/usr/bin/amazon-ssm-agent`
Journaux SSM Agent	`/var/log/amazon/ssm`
AWS informations d'identification	`/root/.aws/credentials`
Configuration SSM CLI	`/opt/ssm/ssm-setup-cli`

Redémarrage de l'agent SSM

Certains problèmes peuvent être résolus en redémarrant l'agent SSM. Vous pouvez utiliser les commandes ci-dessous pour le redémarrer.

AL2023 et autres systèmes d'exploitation

```
systemctl restart amazon-ssm-agent
```

Ubuntu

```
systemctl restart snap.amazon-ssm-agent.amazon-ssm-agent
```

Vérifiez la connectivité aux points de terminaison SSM

Vérifiez que vous pouvez vous connecter aux points de terminaison SSM à partir de vos nœuds hybrides. Pour obtenir la liste des points de terminaison SSM, consultez la section Points de terminaison et quotas Systems Manager AWS. Remplacez `us-west-2` la commande ci-dessous par la AWS région pour votre activation hybride AWS SSM.

```
ping ssm.us-west-2.amazonaws.com
```

Afficher l'état de connexion des instances SSM enregistrées

Vous pouvez vérifier l'état de connexion des instances enregistrées avec les activations hybrides SSM à l'aide de la commande AWS CLI suivante. Remplacez l'ID de machine par l'ID de machine de votre instance.

```
aws ssm get-connection-status --target mi-012345678abcdefgh
```

Incompatibilité de la somme de contrôle de la CLI de configuration SSM

Si vous constatez un problème de discordance de somme de contrôle `ssm-setup-cli` lors de l'exécution de `nodeadm install`, vous devez vérifier qu'il n'existe pas d'anciennes installations SSM sur votre hôte. Si votre hôte contient d'anciennes installations SSM, supprimez-les et relancez `nodeadm install` pour résoudre le problème.

```
Failed to perform agent-installation/on-prem registration: error while verifying
 installed ssm-setup-cli checksum: checksum mismatch with latest ssm-setup-cli.
```

SSM **`InvalidActivation`**

Si vous constatez une erreur lors de l'enregistrement de votre instance auprès de AWS SSM, confirmez que le `regionactivationCode`, et `activationId` dans votre nom `nodeConfig.yaml` sont corrects. La AWS région de votre cluster EKS doit correspondre à la région de votre activation hybride SSM. Si ces valeurs sont mal configurées, vous pourriez voir une erreur similaire à celle-ci.

```
ERROR Registration failed due to error registering the instance with AWS SSM.
 InvalidActivation
```

SSM **`ExpiredTokenException`** : le jeton de sécurité inclus dans la demande a expiré

Si l'agent SSM n'est pas en mesure d'actualiser les informations d'identification, vous pouvez voir apparaître un `ExpiredTokenException`. Dans ce scénario, si vous parvenez à vous connecter aux points de terminaison SSM à partir de vos nœuds hybrides, vous devrez peut-être redémarrer l'agent SSM pour forcer l'actualisation des informations d'identification.

```
"msg":"Command failed","error":"operation error SSM: DescribeInstanceInformation, https
 response error StatusCode: 400, RequestID: eee03a9e-f7cc-470a-9647-73d47e4cf0be, api
 error ExpiredTokenException: The security token included in the request is expired"
```

Erreur SSM lors de l'exécution de la commande register machine

Si vous constatez une erreur lors de l'enregistrement de la machine auprès de SSM, vous devrez peut-être relancer `nodeadm install` afin de vous assurer que toutes les dépendances SSM sont correctement installées.

```
"error":"running register machine command: , error: fork/exec /opt/aws/ssm-setup-cli:
 no such file or directory"
```

SSM **`ActivationExpired`**

Lors de l'exécution de `nodeadm init`, si vous constatez une erreur lors de l'enregistrement de l'instance auprès de SSM en raison d'une activation expirée, vous devez créer une nouvelle activation hybride SSM, mettre à jour votre `nodeConfig.yaml` avec le `activationCode` et le `activationId` de votre nouvelle activation hybride SSM, puis relancer `nodeadm init`.

```
"msg":"Command failed","error":"SSM activation expired. Please use a valid activation"
```

```
ERROR Registration failed due to error registering the instance with AWS SSM.
 ActivationExpired
```

SSM n'a pas réussi à actualiser les informations d'identification mises en cache

Si vous ne parvenez pas à actualiser les informations d'identification mises en cache, le fichier `/root/.aws/credentials` a peut-être été supprimé sur votre hôte. Vérifiez d'abord l'activation hybride SSM et assurez-vous qu'elle est active et que vos nœuds hybrides sont correctement configurés pour utiliser l'activation. Vérifiez les journaux de l'agent SSM à `/var/log/amazon/ssm` et réexécutez la commande `nodeadm init` une fois que vous avez résolu le problème côté SSM.

```
"Command failed","error":"operation error SSM: DescribeInstanceInformation, get
 identity: get credentials: failed to refresh cached credentials"
```

Nettoyez SSM

Pour supprimer l'agent SSM de votre hôte, vous pouvez exécuter les commandes suivantes.

```
dnf remove -y amazon-ssm-agent
sudo apt remove --purge amazon-ssm-agent
snap remove amazon-ssm-agent
```

```
rm -rf /var/lib/amazon/ssm/Vault/Store/RegistrationKey
```

Résolution des problèmes liés à Rôles Anywhere IAM AWS

Si vous utilisez AWS IAM Roles Anywhere pour les informations d'identification de vos nœuds hybrides, tenez compte des répertoires et artefacts suivants qui sont installés sur vos nœuds hybrides par`nodeadm`. Pour plus d'informations sur la résolution des problèmes liés à IAM Roles Anywhere, consultez la section Résolution des problèmes d'identité et d'accès à AWS IAM Roles Anywhere dans le guide de l'utilisateur d' AWS IAM Roles Anywhere.

Description	Location
CLI Rôles Anywhere IAM	`/usr/local/bin/aws_signing_helper`
Emplacement et nom du certificat par défaut	`/etc/iam/pki/server.pem`
Emplacement et nom de la clé par défaut	`/etc/iam/pki/server.key`

Rôles Anywhere IAM n'a pas pu actualiser les informations d'identification mises en cache

Si vous constatez un échec de l'actualisation des informations d'identification mises en cache, vérifiez le contenu de `/etc/aws/hybrid/config` et assurez-vous que Rôles Anywhere IAM a été correctement configuré dans votre configuration `nodeadm`. Confirmez que `/etc/iam/pki` existe. Chaque nœud doit disposer d'un certificat et d'une clé uniques. Par défaut, lorsque vous utilisez Rôles Anywhere IAM comme fournisseur d'informations d'identification, `nodeadm` utilise `/etc/iam/pki/server.pem` pour l'emplacement et le nom du certificat, et `/etc/iam/pki/server.key` pour la clé privée. Vous devrez peut-être créer les répertoires avant d'y placer les certificats et les clés avec `sudo mkdir -p /etc/iam/pki`. Vous pouvez vérifier le contenu de votre certificat à l'aide de la commande ci-dessous.

```
openssl x509 -text -noout -in server.pem
```

```
open /etc/iam/pki/server.pem: no such file or directory
could not parse PEM data
Command failed {"error": "... get identity: get credentials: failed to refresh cached
 credentials, process provider error: error in credential_process: exit status 1"}
```

Rôles Anywhere IAM n'a pas autorisé à exécuter **`sts:AssumeRole`**

Dans les journaux `kubelet`, si vous constatez un problème d'accès refusé pour l'opération `sts:AssumeRole` lors de l'utilisation de Rôles Anywhere IAM, vérifiez la politique de confiance de votre rôle IAM des nœuds hybrides afin de confirmer que le principal de service Rôles Anywhere IAM est autorisé à assumer le rôle IAM des nœuds hybrides. Vérifiez également que l'ARN de l'ancre d'approbation est correctement configurée dans votre stratégie de confiance du rôle IAM des nœuds hybrides et que votre rôle IAM des nœuds hybrides est ajouté à votre profil Rôles Anywhere IAM.

```
could not get token: AccessDenied: User: ... is not authorized to perform:
 sts:AssumeRole on resource: ...
```

Rôles Anywhere IAM n'est pas autorisé à définir **`roleSessionName`**

Dans les journaux `kubelet`, si vous constatez un problème d'accès refusé pour la configuration du paramètre `roleSessionName`, vérifiez que vous avez défini la valeur `acceptRoleSessionName` sur true pour votre profil Rôles Anywhere IAM.

```
AccessDeniedException: Not authorized to set roleSessionName
```

Résolution des problèmes de système d'exploitation

RHEL

Échec de l'enregistrement du gestionnaire d'autorisations ou d'abonnements

Lorsque vous exécutez `nodeadm install`, si vous rencontrez un problème lors de l'installation des dépendances des nœuds hybrides en raison d'un problème d'enregistrement des droits, assurez-vous d'avoir correctement défini votre nom d'utilisateur et votre mot de passe Red Hat sur votre hôte.

```
This system is not registered with an entitlement server
```

Ubuntu

GLIBC introuvable

Si vous utilisez Ubuntu comme système d'exploitation et Rôles Anywhere IAM comme fournisseur d'informations d'identification avec des nœuds hybrides et que vous rencontrez un problème lié

à l'absence de GLIBC, vous pouvez installer cette dépendance manuellement pour résoudre le problème.

```
GLIBC_2.32 not found (required by /usr/local/bin/aws_signing_helper)
```

Exécutez les commandes suivantes pour installer la dépendance :

```
ldd --version
sudo apt update && apt install libc6
sudo apt install glibc-source
```

Flacon Rocket

Si vous avez activé le conteneur d'administration Bottlerocket, vous pouvez y accéder via SSH pour effectuer un débogage avancé et un dépannage avec des privilèges élevés. Les sections suivantes contiennent des commandes qui doivent être exécutées dans le contexte de l'hôte Bottlerocket. Une fois que vous êtes dans le conteneur d'administration, vous pouvez exécuter `sheltie` pour obtenir un shell root complet dans l'hôte Bottlerocket.

```
sheltie
```

Vous pouvez également exécuter les commandes des sections suivantes à partir du shell du conteneur admin en préfixant chaque commande avec `sudo chroot /.bottlerocket/rootfs`.

```
sudo chroot /.bottlerocket/rootfs <command>
```

Utilisation de logdog pour la collecte de logs

Bottlerocket fournit l'utilitaire `logdog` permettant de collecter efficacement les journaux et les informations système à des fins de dépannage.

```
logdog
```

L'utilitaire `logdog` rassemble les journaux provenant de divers emplacements sur un hôte Bottlerocket et les combine dans un fichier tar. Par défaut, l'archive tar sera créée à l'emplacement `/var/log/support/bottlerocket-logs.tar.gz` et sera accessible depuis les conteneurs hôtes à l'emplacement `/.bottlerocket/support/bottlerocket-logs.tar.gz`.

Accès aux journaux du système avec journalctl

Vous pouvez vérifier l'état des différents services système tels que `kubelet`, `containerd`, etc., et consulter leurs journaux à l'aide des commandes suivantes. La balise `-f` suivra les bûches en temps réel.

Pour vérifier l'état du service `kubelet` et récupérer les journaux `kubelet`, vous pouvez exécuter :

```
systemctl status kubelet
journalctl -u kubelet -f
```

Pour vérifier l'état du service `containerd` et récupérer les journaux `containerd` de l'instance orchestrée, vous pouvez exécuter :

```
systemctl status containerd
journalctl -u containerd -f
```

Pour vérifier l'état du service `host-containerd` et récupérer les journaux `containerd` de l'instance hôte, vous pouvez exécuter :

```
systemctl status host-containerd
journalctl -u host-containerd -f
```

Pour récupérer les journaux des conteneurs bootstrap et des conteneurs hôtes, vous pouvez exécuter :

```
journalctl _COMM=host-ctr -f
```

Utiliser le stockage de données d'application pour votre cluster

Vous pouvez utiliser une gamme de services de AWS stockage avec Amazon EKS pour répondre aux besoins de stockage de vos applications. Grâce à un large éventail de pilotes d'interface de stockage de conteneurs (CSI) pris en AWS charge, vous pouvez facilement utiliser Amazon EBS, Amazon S3, Amazon EFS, Amazon FSX et Amazon File Cache pour répondre aux besoins de stockage de vos applications exécutées sur Amazon EKS. Pour gérer les sauvegardes de votre cluster Amazon EKS, consultez la section Support AWS de sauvegarde pour Amazon EKS.

Ce chapitre couvre les options de stockage pour les clusters Amazon EKS.

Rubriques

- Utilisez le stockage de volumes Kubernetes avec Amazon EBS
- Utilisation du stockage en système de fichiers Elastic avec Amazon EFS
- Utilisez un stockage d'applications haute performance avec Amazon FSx pour Lustre
- Utilisez un stockage d'applications haute performance avec FSx pour NetApp ONTAP
- Utilisez le stockage de données avec Amazon FSx pour OpenZFS
- Minimisez le temps de latence avec Amazon File Cache
- Accéder aux objets Amazon S3 avec le pilote CSI Mountpoint pour Amazon S3
- Activer la fonctionnalité d'instantané pour les volumes CSI

Utilisez le stockage de volumes Kubernetes avec Amazon EBS

> **Note**
>
> Nouveau : le mode automatique Amazon EKS automatise les tâches de routine pour le stockage par blocs. Découvrez comment the section called "Déployer une charge de travail avec état".

Le pilote CSI (Container Storage Interface) Amazon Elastic Block Store (Amazon EBS) gère le cycle de vie des volumes Amazon EBS en tant que stockage pour les volumes Kubernetes que vous créez.

Le pilote CSI Amazon EBS crée des volumes Amazon EBS pour ces types de volumes Kubernetes : volumes éphémères génériques et volumes persistants.

Considérations

- Il n'est pas nécessaire d'installer le contrôleur CSI Amazon EBS sur les clusters du mode automatique EKS.
- Vous ne pouvez pas monter des volumes Amazon EBS sur des pods Fargate.
- Vous pouvez exécuter le contrôleur Amazon EBS CSI sur des nœuds Fargate, mais le nœud `DaemonSet` Amazon EBS CSI ne peut fonctionner que sur des instances Amazon. EC2
- Les volumes Amazon EBS et le pilote CSI Amazon EBS ne sont pas compatibles avec les nœuds hybrides Amazon EKS.
- Une assistance sera fournie pour la dernière version complémentaire et la version précédente. Les bogues ou vulnérabilités détectés dans la dernière version seront rétroportés vers la version précédente dans une nouvelle version mineure.
- Le mode automatique EKS nécessite des classes de stockage pour utiliser `ebs.csi.eks.amazonaws.com` comme provisionneur. Le pilote Amazon EBS CSI standard (`ebs.csi.aws.com`) gère ses propres volumes séparément. Pour utiliser les volumes existants avec le mode automatique EKS, migrez-les à l'aide de snapshots de volume vers une classe de stockage qui utilise le provisionneur en mode automatique.

> **Important**
>
> Pour utiliser la fonctionnalité de snapshot du pilote CSI Amazon EBS, vous devez d'abord installer le contrôleur de snapshot CSI. Pour de plus amples informations, veuillez consulter the section called "Contrôleur d'instantané CSI".

Conditions préalables

- Un cluster existant. Pour afficher la version requise de la plateforme, exécutez la commande suivante.

```
aws eks describe-addon-versions --addon-name aws-ebs-csi-driver
```

- Le pilote EBS CSI a besoin d'autorisations AWS IAM.

- AWS suggère d'utiliser EKS Pod Identities. Pour de plus amples informations, veuillez consulter the section called “Présentation de la configuration des identités du pod EKS”.
- Pour plus d'informations sur les rôles IAM pour les comptes de service, consultez the section called “Fournisseur IAM OIDC”.

Étape 1 : Créer un rôle IAM

Le plug-in Amazon EBS CSI nécessite des autorisations IAM pour passer des appels en votre AWS APIs nom. Si vous ne suivez pas ces étapes, la tentative d'installation et d'exécution de `kubectl describe pvc` affichera `failed to provision volume with StorageClass` et s'accompagnera d'une erreur `could not create volume in EC2: UnauthorizedOperation`. Pour plus d'informations, consultez Configuration de l'autorisation du pilote sur GitHub.

> **Note**
>
> Les pods auront accès aux autorisations attribuées au rôle IAM, sauf si vous bloquez l'accès à IMDS. Pour de plus amples informations, veuillez consulter the section called “Bonnes pratiques”.

La procédure suivante explique comment créer un rôle IAM et y associer la politique AWS gérée. Pour mettre en œuvre cette procédure, vous pouvez utiliser l'un des outils suivants :

- the section called “`eksctl`”
- the section called “AWS Management Console”
- the section called “AWS CLI”

> **Note**
>
> Vous pouvez créer une politique autogérée avec des autorisations limitées. Vérifiez `AmazonEBSCSIDriverPolicy` et créez une politique IAM personnalisée avec des autorisations réduites.

> **Note**
>
> Les étapes spécifiques de cette procédure sont destinées à l'utilisation du pilote en tant que module complémentaire d'Amazon EKS. Différentes étapes sont nécessaires pour utiliser le pilote en tant que module complémentaire autogéré. Pour plus d'informations, voir Configurer les autorisations du conducteur sur GitHub.

`eksctl`

1. Créez un rôle IAM et associez une politique. AWS gère une politique AWS gérée ou vous pouvez créer votre propre politique personnalisée. Vous pouvez créer un rôle IAM et associer la politique AWS gérée à l'aide de la commande suivante. Remplacez *my-cluster* par le nom de votre cluster. La commande déploie une AWS CloudFormation pile qui crée un rôle IAM et y attache la politique IAM.

   ```
   eksctl create iamserviceaccount \
           --name ebs-csi-controller-sa \
           --namespace kube-system \
           --cluster my-cluster \
           --role-name AmazonEKS_EBS_CSI_DriverRole \
           --role-only \
           --attach-policy-arn arn:aws: iam::aws:policy/service-role/
   AmazonEBSCSIDriverPolicy \
           --approve
   ```

2. Vous pouvez ignorer cette étape si vous n'utilisez pas de clé KMS personnalisée. Si vous en utilisez un pour le chiffrement sur vos volumes Amazon EBS, personnalisez le rôle IAM selon vos besoins. Par exemple, procédez comme suit :

   a. Copiez et collez le code suivant dans un nouveau fichier `kms-key-for-encryption-on-ebs.json`. Remplacez *custom-key-arn* par l'ARN de clé KMS personnalisé.

   ```
   {
         "Version":"2012-10-17",
         "Statement": [
           {
             "Effect": "Allow",
             "Action": [
               "kms:CreateGrant",
               "kms:ListGrants",
   ```

```
            "kms:RevokeGrant"
          ],
          "Resource": ["arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab"],
          "Condition": {
            "Bool": {
              "kms:GrantIsForAWSResource": "true"
            }
          }
        },
        {
          "Effect": "Allow",
          "Action": [
            "kms:Encrypt",
            "kms:Decrypt",
            "kms:ReEncrypt*",
            "kms:GenerateDataKey*",
            "kms:DescribeKey"
          ],
          "Resource": ["arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab"]
        }
      ]
    }
```

b. Créez la politique. Vous pouvez remplacer *KMS_Key_For_Encryption_On_EBS_Policy* par un autre nom. Cependant, si vous le remplacez, assurez-vous de le remplacer également dans les étapes suivantes.

```
aws iam create-policy \
      --policy-name KMS_Key_For_Encryption_On_EBS_Policy \
      --policy-document file://kms-key-for-encryption-on-ebs.json
```

c. Attachez la politique IAM au rôle à l'aide de la commande suivante. Remplacez *111122223333* par votre ID de compte.

```
aws iam attach-role-policy \
      --policy-arn arn:aws: iam::111122223333:policy/
KMS_Key_For_Encryption_On_EBS_Policy \
      --role-name AmazonEKS_EBS_CSI_DriverRole
```

AWS Management Console

1. Ouvrez la console IAM à https://console.aws.amazon.com/iam/ l'adresse.
2. Dans le volet de navigation de gauche, choisissez Rôles.
3. Sur la page Rôles, choisissez Créer un rôle.
4. Sur la page Select trusted entity (Sélectionnez une entité de confiance), procédez comme suit :
   a. Dans la section Trusted entity type (Type d'entité de confiance), sélectionnez Web identity (Identité web).
   b. Pour Fournisseur d'identité, choisissez l'URL du fournisseur OpenID Connect pour votre cluster (comme indiqué sous Présentation dans Amazon EKS).
   c. Pour Audience, choisissez `sts.amazonaws.com`.
   d. Choisissez Suivant.
5. Sur la page Add permissions (Ajouter des autorisations), procédez comme suit :
   a. Dans la zone Filter policies (Politiques de filtre), saisissez `AmazonEBSCSIDriverPolicy`.
   b. Cochez la case à gauche du `AmazonEBSCSIDriverPolicy` renvoyé dans la recherche.
   c. Choisissez Suivant.
6. Sur la page Name, review, and create (Nommer, vérifier et créer), procédez comme suit :
   a. Pour Role name (Nom de rôle), saisissez un nom unique pour votre rôle, par exemple, *`AmazonEKS_EBS_CSI_DriverRole`*.
   b. Sous Ajouter des balises (Facultatif), ajoutez des métadonnées au rôle en attachant les identifications sous forme de paires clé-valeur. Pour plus d'informations sur l'utilisation des balises dans IAM, consultez la rubrique Balisage des ressources IAM dans le Guide de l'utilisateur IAM.
   c. Choisissez Créer un rôle.
7. Une fois le rôle créé, choisissez le rôle dans la console pour l'ouvrir et le modifier.
8. Sélectionnez l'onglet Trust relationships (Relations d'approbation), puis Edit trust policy (Modifier la politique d'approbation).
9. Trouvez la ligne qui ressemble à la ligne suivante :

```
"oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE:aud":
 "sts.amazonaws.com"
```

Ajoutez une virgule à la fin de la ligne précédente, puis ajoutez la ligne suivante après la ligne précédente. Remplacez *region-code* par la AWS région dans laquelle se trouve votre cluster. Remplacez *EXAMPLED539D4633E53DE1B71EXAMPLE* par l'ID du fournisseur OIDC de votre cluster.

```
"oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE:sub":
 "system:serviceaccount:kube-system:ebs-csi-controller-sa"
```

10. Sélectionnez Update Trust Policy (Mettre à jour la politique d'approbation) pour terminer.
11. Si vous utilisez une clé KMS personnalisée pour le chiffrement de vos volumes Amazon EBS, personnalisez le rôle IAM en fonction des besoins. Par exemple, procédez comme suit :
    a. Dans le volet de navigation de gauche, choisissez Politiques.
    b. Sur la page Politiques, choisissez Créer une politique.
    c. Sur la page Créer une politique, choisissez l'onglet JSON.
    d. Copiez et collez le code suivant dans l'éditeur, en remplaçant *custom-key-arn* par l'ARN de clé KMS personnalisée.

```
{
      "Version":"2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Action": [
            "kms:CreateGrant",
            "kms:ListGrants",
            "kms:RevokeGrant"
          ],
          "Resource": ["arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab"],
          "Condition": {
            "Bool": {
              "kms:GrantIsForAWSResource": "true"
            }
          }
        },
        {
          "Effect": "Allow",
          "Action": [
            "kms:Encrypt",
```

```
            "kms:Decrypt",
            "kms:ReEncrypt*",
            "kms:GenerateDataKey*",
            "kms:DescribeKey"
          ],
          "Resource": ["arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab"]
        }
      ]
    }
```

e. Choisissez Suivant : Balises.

f. Sur la page Ajouter des balises (facultatif), choisissez Suivant : Vérifier.

g. Dans Nom, attribuez un nom unique à votre politique (par exemple *KMS_Key_For_Encryption_On_EBS_Policy*).

h. Choisissez Create Policy (Créer une politique).

i. Dans le volet de navigation de gauche, choisissez Rôles.

j. Choisissez le ***AmazonEKS_EBS_CSI_DriverRole***dans la console pour l'ouvrir en vue de le modifier.

k. Dans la liste déroulante Ajouter des autorisations, choisissez Associer des politiques.

l. Dans la zone Filter policies (Politiques de filtre), saisissez *KMS_Key_For_Encryption_On_EBS_Policy*.

m. Cochez la case à gauche du *KMS_Key_For_Encryption_On_EBS_Policy* renvoyé dans la recherche.

n. Choisissez Attacher des politiques.

AWS CLI

1. Affichez l'URL du fournisseur OIDC de votre cluster. Remplacez *my-cluster* par le nom de votre cluster. Si la sortie de la commande est `None`, consultez Prérequis.

```
aws eks describe-cluster --name my-cluster --query "cluster.identity.oidc.issuer" --
output text
```

L'exemple qui suit illustre un résultat.

```
https://oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE
```

2. Créez le rôle IAM et attachez-y l'action `AssumeRoleWithWebIdentity`.

   a. Copiez le contenu suivant dans un fichier nommé `aws-ebs-csi-driver-trust-policy.json`. Remplacez *`111122223333`* par votre ID de compte. Remplacez *`EXAMPLED539D4633E53DE1B71EXAMPLE`* et *`region-code`* par les valeurs renvoyées par l'étape précédente.

   ```
   {
         "Version":"2012-10-17",
         "Statement": [
           {
             "Effect": "Allow",
             "Principal": {
               "Federated": "arn:aws:iam::111122223333:oidc-provider/oidc.eks.us-
   east-1.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
             },
             "Action": "sts:AssumeRoleWithWebIdentity",
             "Condition": {
               "StringEquals": {
                 "oidc.eks.us-east-1.amazonaws.com/id/
   EXAMPLED539D4633E53DE1B71EXAMPLE:aud": "sts.amazonaws.com",
                 "oidc.eks.us-east-1.amazonaws.com/id/
   EXAMPLED539D4633E53DE1B71EXAMPLE:sub": "system:serviceaccount:kube-system:ebs-csi-
   controller-sa"
               }
             }
           }
         ]
       }
   ```

   b. Créez le rôle. Vous pouvez remplacer *`AmazonEKS_EBS_CSI_DriverRole`* par un autre nom. Si vous le remplacez, assurez-vous de le remplacer dans les étapes suivantes.

   ```
   aws iam create-role \
         --role-name AmazonEKS_EBS_CSI_DriverRole \
         --assume-role-policy-document file://"aws-ebs-csi-driver-trust-policy.json"
   ```

3. Joignez une politique. AWS gère une politique AWS gérée ou vous pouvez créer votre propre politique personnalisée. Attachez la politique AWS gérée au rôle à l'aide de la commande suivante.

```
aws iam attach-role-policy \
      --policy-arn arn:aws: iam::aws:policy/service-role/AmazonEBSCSIDriverPolicy \
      --role-name AmazonEKS_EBS_CSI_DriverRole
```

4. Si vous utilisez une clé KMS personnalisée pour le chiffrement de vos volumes Amazon EBS, personnalisez le rôle IAM en fonction des besoins. Par exemple, procédez comme suit :

   a. Copiez et collez le code suivant dans un nouveau fichier `kms-key-for-encryption-on-ebs.json`. Remplacez *`custom-key-arn`* par l'ARN de clé KMS personnalisé.

```
{
      "Version":"2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Action": [
            "kms:CreateGrant",
            "kms:ListGrants",
            "kms:RevokeGrant"
          ],
          "Resource": ["arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab"],
          "Condition": {
            "Bool": {
              "kms:GrantIsForAWSResource": "true"
            }
          }
        },
        {
          "Effect": "Allow",
          "Action": [
            "kms:Encrypt",
            "kms:Decrypt",
            "kms:ReEncrypt*",
            "kms:GenerateDataKey*",
            "kms:DescribeKey"
          ],
          "Resource": ["arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab"]
        }
      ]
    }
```

b. Créez la politique. Vous pouvez remplacer *KMS_Key_For_Encryption_On_EBS_Policy* par un autre nom. Cependant, si vous le remplacez, assurez-vous de le remplacer également dans les étapes suivantes.

```
aws iam create-policy \
      --policy-name KMS_Key_For_Encryption_On_EBS_Policy \
      --policy-document file://kms-key-for-encryption-on-ebs.json
```

c. Attachez la politique IAM au rôle à l'aide de la commande suivante. Remplacez *111122223333* par votre ID de compte.

```
aws iam attach-role-policy \
      --policy-arn arn:aws: iam::111122223333:policy/
KMS_Key_For_Encryption_On_EBS_Policy \
      --role-name AmazonEKS_EBS_CSI_DriverRole
```

Maintenant que vous avez créé le rôle IAM du pilote CSI Amazon EBS, vous pouvez passer à la section suivante. Lorsque vous déployez le module complémentaire avec ce rôle IAM, il crée et est configuré pour utiliser un compte de service nommé `ebs-csi-controller-sa`. Le compte de service est lié à un Kubernetes `clusterrole` auquel sont attribuées les autorisations Kubernetes requises.

Étape 2 : obtenir le pilote CSI Amazon EBS

Nous vous recommandons d'installer le pilote Amazon EBS CSI via le module complémentaire Amazon EKS afin d'améliorer la sécurité et de réduire la charge de travail. Pour ajouter un module complémentaire Amazon EKS à votre cluster, voir the section called "Créer un module complémentaire". Pour plus d'informations sur les modules complémentaires, voir the section called "Modules complémentaires Amazon EKS".

> **Important**
>
> Avant d'ajouter le pilote Amazon EBS en tant que module complémentaire Amazon EKS, vérifiez qu'aucune version autogérée du pilote n'est déjà installée sur votre cluster. Si c'est le cas, voir Désinstallation d'un pilote CSI Amazon EBS autogéré sur GitHub.

> **Note**
>
> Par défaut, le rôle RBAC utilisé par le CSI EBS est autorisé à muter les nœuds afin de prendre en charge sa fonction de suppression des taches. En raison des limites de Kubernetes RBAC, cela lui permet également de muter n'importe quel autre nœud du cluster. Le graphique Helm comporte un paramètre (`node.serviceAccount.disableMutation`) qui désactive la mutation des autorisations RBAC des nœuds pour le ebs-csi-node compte de service. Lorsqu'elles sont activées, les fonctionnalités du pilote telles que la suppression des taches ne fonctionneront pas.

Sinon, si vous souhaitez une installation autogérée du pilote Amazon EBS CSI, consultez la section Installation sur. GitHub

Étape 3 : déployer un exemple d'application

Vous pouvez déployer une variété d'applications types et les modifier selon vos besoins. Pour plus d'informations, consultez les exemples de Kubernetes sur. GitHub

Utilisation du stockage en système de fichiers Elastic avec Amazon EFS

Amazon Elastic File System (Amazon EFS) fournit un stockage de fichiers entièrement élastique sans serveur pour vous permettre de partager des données de fichiers sans provisionner ni gérer la capacité et les performances de stockage. Le pilote Amazon EFS Container Storage Interface (CSI) fournit une interface CSI qui permet aux clusters Kubernetes exécutés de gérer le cycle de vie des AWS systèmes de fichiers Amazon EFS. Cette rubrique vous montre comment déployer le pilote Amazon EFS CSI dans votre cluster Amazon EKS.

Considérations

- Le pilote Amazon EFS CSI n'est pas compatible avec les images de conteneurs basées sur Windows.
- Vous ne pouvez pas utiliser l'approvisionnement dynamique pour les volumes persistants avec les nœuds Fargate, mais vous pouvez utiliser l'approvisionnement statique.
- L'approvisionnement dynamique nécessite la version 1.2 ou ultérieure du pilote. Vous pouvez utiliser l'approvisionnement statique pour les volumes persistants à l'aide de la version `1.1`

du pilote sur n'importe quelle version de cluster Amazon EKS prise en charge (voir Versions Amazon EKS prises en charge).

- La version 1.3.2 ou ultérieure de ce pilote prend en charge l'architecture Arm64, y compris les instances basées sur Amazon EC2 Graviton.
- Les versions 1.4.2 et ultérieures de ce pilote prennent en charge l'utilisation de FIPS pour le montage de systèmes de fichiers.
- Prenez note des quotas de ressources pour Amazon EFS. Par exemple, un quota de 1 000 points d'accès peut être créé pour chaque système de fichiers Amazon EFS. Pour plus d'informations, consultez les quotas de ressources Amazon EFS que vous ne pouvez pas modifier.
- À partir de la version 2.0.0, ce pilote est passé de l'utilisation de `stunnel` à celle de `efs-proxy` pour les connexions TLS. Lorsque `efs-proxy` est utilisé, il ouvre un nombre de threads égal à un plus le nombre de cœurs pour le nœud sur lequel il s'exécute.
- Le pilote CSI Amazon EFS n'est pas compatible avec les nœuds Amazon EKS.

Conditions préalables

- Le pilote Amazon EFS CSI nécessite des autorisations AWS Identity and Access Management (IAM).
  - AWS suggère d'utiliser EKS Pod Identities. Pour de plus amples informations, veuillez consulter the section called "Présentation de la configuration des identités du pod EKS".
  - Pour plus d'informations sur les rôles IAM pour les comptes de service et la configuration d'un fournisseur IAM OpenID Connect (OIDC) pour votre cluster, consultez the section called "Fournisseur IAM OIDC".
- Version `2.12.3` ou version ultérieure `1.27.160` ou version ultérieure de l'interface de ligne de AWS commande (AWS CLI) installée et configurée sur votre appareil ou AWS CloudShell. Pour vérifier votre version actuelle, utilisez `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Les gestionnaires de packages tels que `yum` Homebrew pour macOS ont souvent plusieurs versions de retard sur la dernière version de la AWS CLI. `apt-get` Pour installer la dernière version, consultez la section Installation et configuration rapide avec aws configure dans le Guide de l'utilisateur de l'interface de ligne de AWS commande. La version de la AWS CLI installée AWS CloudShell peut également avoir plusieurs versions de retard par rapport à la dernière version. Pour le mettre à jour, consultez la section Installation de la AWS CLI dans votre répertoire de base dans le guide de AWS CloudShell l'utilisateur.

- L'outil de ligne de commande `kubectl` est installé sur votre appareil ou AWS CloudShell. La version peut correspondre à celle utilisée par votre cluster Kubernetes, ou différer d'au plus une version mineure, qu'elle soit antérieure ou plus récente. Par exemple, si la version de votre cluster est `1.29`, vous pouvez utiliser la version `kubectl` `1.28`, `1.29` ou `1.30`. Pour installer ou mettre à niveau `kubectl`, veuillez consulter the section called "Configurer kubectl et eksctl".

> **Note**
>
> Un Pod fonctionnant sur Fargate monte automatiquement un système de fichiers Amazon EFS, sans nécessiter d'étapes manuelles d'installation de pilotes.

Étape 1 : Créer un rôle IAM

Le pilote CSI Amazon EFS nécessite des autorisations IAM pour interagir avec votre système de fichiers. Créez un rôle IAM et associez-y la politique AWS gérée requise. Pour mettre en œuvre cette procédure, vous pouvez utiliser l'un des outils suivants :

- the section called "`eksctl`"
- the section called "AWS Management Console"
- the section called "AWS CLI"

> **Note**
>
> Les étapes spécifiques de cette procédure sont destinées à l'utilisation du pilote en tant que module complémentaire d'Amazon EKS. Pour plus de détails sur les installations autogérées, voir Configurer l'autorisation du pilote sur GitHub.

`eksctl`

Si vous utilisez les identités du pod

Exécutez les commandes suivantes pour créer un rôle IAM et une association d'identité de pod avec `eksctl`. Remplacez `my-cluster` par le nom de votre cluster. Vous pouvez aussi remplacer `AmazonEKS_EFS_CSI_DriverRole` par un nom différent.

```
export cluster_name=my-cluster
export role_name=AmazonEKS_EFS_CSI_DriverRole
eksctl create podidentityassociation \
    --service-account-name efs-csi-controller-sa \
    --namespace kube-system \
    --cluster $cluster_name \
    --role-name $role_name \
    --permission-policy-arns arn:aws: iam::aws:policy/service-role/
AmazonEFSCSIDriverPolicy
```

Si vous utilisez des rôles IAM pour des comptes de service

Exécutez les commandes suivantes pour créer un rôle IAM avec `eksctl`. Remplacez `my-cluster` par le nom de votre cluster. Vous pouvez aussi remplacer `AmazonEKS_EFS_CSI_DriverRole` par un nom différent.

```
export cluster_name=my-cluster
export role_name=AmazonEKS_EFS_CSI_DriverRole
eksctl create iamserviceaccount \
    --name efs-csi-controller-sa \
    --namespace kube-system \
    --cluster $cluster_name \
    --role-name $role_name \
    --role-only \
    --attach-policy-arn arn:aws: iam::aws:policy/service-role/AmazonEFSCSIDriverPolicy
 \
    --approve
TRUST_POLICY=$(aws iam get-role --output json --role-name $role_name --query
 'Role.AssumeRolePolicyDocument' | \
    sed -e 's/efs-csi-controller-sa/efs-csi-*/' -e 's/StringEquals/StringLike/')
aws iam update-assume-role-policy --role-name $role_name --policy-document
 "$TRUST_POLICY"
```

AWS Management Console

Exécutez la commande suivante pour créer un rôle IAM avec AWS Management Console.

1. Ouvrez la console IAM à https://console.aws.amazon.com/iam/ l'adresse.
2. Dans le volet de navigation de gauche, choisissez Rôles.
3. Sur la page Rôles, choisissez Créer un rôle.

4. Sur la page Select trusted entity (Sélectionner une entité de confiance), procédez comme suit :
   a. Si vous utilisez les identités de pod EKS :
      i. Dans la section Type d'entité fiable, sélectionnez AWS service.
      ii. Dans le menu déroulant Service ou cas d'utilisation, sélectionnez EKS.
      iii. Dans la section Cas d'utilisation, sélectionnez EKS – Identité du pod.
      iv. Choisissez Suivant.
   b. Si vous utilisez des rôles IAM pour des comptes de service :
      i. Dans la section Trusted entity type (Type d'entité de confiance), sélectionnez Web identity (Identité web).
      ii. Pour Fournisseur d'identité, choisissez l'URL du fournisseur OpenID Connect pour votre cluster (comme indiqué sous Présentation dans Amazon EKS).
      iii. Pour Audience, choisissez `sts.amazonaws.com`.
      iv. Choisissez Suivant.
5. Sur la page Add permissions (Ajouter des autorisations), procédez comme suit :
   a. Dans la zone Filter policies (Politiques de filtre), saisissez `AmazonEFSCSIDriverPolicy`.
   b. Cochez la case à gauche du `AmazonEFSCSIDriverPolicy` renvoyé dans la recherche.
   c. Choisissez Suivant.
6. Sur la page Name, review, and create (Nommer, vérifier et créer), procédez comme suit :
   a. Pour Role name (Nom de rôle), saisissez un nom unique pour votre rôle, par exemple, `AmazonEKS_EFS_CSI_DriverRole`.
   b. Sous Ajouter des balises (Facultatif), ajoutez des métadonnées au rôle en attachant les identifications sous forme de paires clé-valeur. Pour plus d'informations sur l'utilisation des balises dans IAM, consultez la rubrique Balisage des ressources IAM dans le Guide de l'utilisateur IAM.
   c. Choisissez Créer un rôle.
7. Une fois le rôle créé :
   a. Si vous utilisez les identités de pod EKS :
      i. Ouvrez la console Amazon EKS.
      ii. Dans le volet de navigation gauche, sélectionnez Clusters, puis sélectionnez le nom du cluster pour lequel vous souhaitez configurer l'association d'identité de pod EKS.
      iii. Choisissez l'onglet Access.
      iv. Dans Associations d'identité du pod, sélectionnez Créer.

v. Choisissez la liste déroulante du rôle IAM et sélectionnez le rôle que vous venez de créer.

vi. Sélectionnez le champ Espace de noms Kubernetes et saisissez `kube-system`.

vii. Sélectionnez le champ Compte de service Kubernetes et saisissez `efs-csi-controller-sa`.

viii. Choisissez Créer.

ix. Pour plus d'informations sur la création d'associations d'identités de pod, consultez the section called "Création d'une association Pod Identity (AWS console)".

b. Si vous utilisez des rôles IAM pour des comptes de service :

i. Choisissez le rôle pour l'ouvrir en vue de le modifier.

ii. Sélectionnez l'onglet Trust relationships (Relations d'approbation), puis Edit trust policy (Modifier la politique d'approbation).

iii. Trouvez la ligne qui ressemble à la ligne suivante :

```
"oidc.eks.region-code.amazonaws.com/id/<EXAMPLED539D4633E53DE1B71EXAMPLE>:aud":
 "sts.amazonaws.com"
```

Ajoutez la ligne suivante au-dessus de la ligne précédente. Remplacez `<region-code>` par la AWS région dans laquelle se trouve votre cluster. Remplacez `<EXAMPLED539D4633E53DE1B71EXAMPLE>` par l'ID du fournisseur OIDC de votre cluster.

```
"oidc.eks.<region-code>.amazonaws.com/id/
<EXAMPLED539D4633E53DE1B71EXAMPLE>:sub": "system:serviceaccount:kube-system:efs-
csi-*",
```

iv. Modifiez l'opérateur `Condition` de `"StringEquals"` en `"StringLike"`.

v. Sélectionnez Update Trust Policy (Mettre à jour la politique d'approbation) pour terminer.

AWS CLI

Exécutez les commandes suivantes pour créer un rôle IAM avec la AWS CLI.

Si vous utilisez les identités du pod

1. Créez le rôle IAM qui accorde les actions `AssumeRole` et `TagSession` au service `pods.eks.amazonaws.com`.

a. Copiez le contenu suivant dans un fichier nommé `aws-efs-csi-driver-trust-policy-pod-identity.json`.

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

b. Créez le rôle. Remplacez `my-cluster` par le nom de votre cluster. Vous pouvez aussi remplacer `AmazonEKS_EFS_CSI_DriverRole` par un nom différent.

```
export cluster_name=my-cluster
export role_name=AmazonEKS_EFS_CSI_DriverRole
aws iam create-role \
  --role-name $role_name \
  --assume-role-policy-document file://"aws-efs-csi-driver-trust-policy-pod-identity.json"
```

2. Associez la politique AWS gérée requise au rôle à l'aide de la commande suivante.

```
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::aws:policy/service-role/AmazonEFSCSIDriverPolicy \
  --role-name $role_name
```

3. Exécutez la commande suivante pour créer l'association d'identité de pod. Remplacer `arn:aws:iam::<111122223333>:role/my-role` par le rôle créé lors des étapes précédentes.

```
aws eks create-pod-identity-association --cluster-name $cluster_name --role-arn {arn-aws}iam::<111122223333>:role/my-role --namespace kube-system --service-account efs-csi-controller-sa
```

4. Pour plus d'informations sur la création d'associations d'identités de pod, consultez the section called "Création d'une association Pod Identity (AWS console)".

Si vous utilisez des rôles IAM pour des comptes de service

1. Affichez l'URL du fournisseur OIDC de votre cluster. Remplacez `my-cluster` par le nom de votre cluster. Vous pouvez aussi remplacer `AmazonEKS_EFS_CSI_DriverRole` par un nom différent.

```
export cluster_name=my-cluster
export role_name=AmazonEKS_EFS_CSI_DriverRole
aws eks describe-cluster --name $cluster_name --query "cluster.identity.oidc.issuer"
 --output text
```

   L'exemple qui suit illustre un résultat.

```
https://oidc.eks.<region-code>.amazonaws.com/id/<EXAMPLED539D4633E53DE1B71EXAMPLE>
```

   Si la sortie de la commande est `None`, consultez Prérequis.

2. Créez le rôle IAM qui accorde l'action `AssumeRoleWithWebIdentity`.
   a. Copiez le contenu suivant dans un fichier nommé `aws-efs-csi-driver-trust-policy.json`. Remplacez `<111122223333>` par votre ID de compte. Remplacez `<EXAMPLED539D4633E53DE1B71EXAMPLE>` et `<region-code>` par les valeurs renvoyées par l'étape précédente.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::123456789012:oidc-provider/oidc.eks.us-east-1.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
```

```
      "Condition": {
        "StringLike": {
          "oidc.eks.us-east-1.amazonaws.com/id/
EXAMPLED539D4633E53DE1B71EXAMPLE:sub": "system:serviceaccount:kube-system:efs-csi-
*",
          "oidc.eks.us-east-1.amazonaws.com/id/
EXAMPLED539D4633E53DE1B71EXAMPLE:aud": "sts.amazonaws.com"
        }
      }
    }
  ]
}
```

b. Créez le rôle.

```
aws iam create-role \
  --role-name $role_name \
  --assume-role-policy-document file://"aws-efs-csi-driver-trust-policy.json"
```

3. Associez la politique AWS gérée requise au rôle à l'aide de la commande suivante.

```
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::aws:policy/service-role/AmazonEFSCSIDriverPolicy \
  --role-name $role_name
```

Étape 2 : obtenir le pilote CSI Amazon EFS

Nous vous recommandons d'installer le pilote CSI Amazon EFS en utilisant le module complémentaire Amazon EKS. Pour ajouter un module complémentaire Amazon EKS à votre cluster, voir the section called “Créer un module complémentaire”. Pour plus d'informations sur les modules complémentaires, voir the section called “Modules complémentaires Amazon EKS”. Si vous ne parvenez pas à utiliser le module complémentaire Amazon EKS, nous vous encourageons à signaler les raisons pour lesquelles vous ne pouvez pas utiliser le module complémentaire Amazon EKS dans le GitHub référentiel de feuilles de route pour les conteneurs.

> **Important**
>
> Avant d'ajouter le pilote Amazon EFS en tant que module complémentaire Amazon EKS, vérifiez qu'aucune version autogérée du pilote n'est déjà installée sur votre cluster. Dans ce cas, consultez la section Désinstallation du pilote Amazon EFS CSI sur. GitHub

Sinon, si vous souhaitez une installation autogérée du pilote CSI Amazon EFS, voir Installation sur GitHub.

Étape 3 : Créer un système de fichiers Amazon EFS

Pour créer un système de fichiers Amazon EFS, voir Création d'un système de fichiers Amazon EFS pour Amazon EKS sur GitHub.

Étape 4 : déployer un exemple d'application

Vous pouvez déployer une variété d'applications types et les modifier selon vos besoins. Pour plus d'informations, consultez la section Exemples sur GitHub.

Utilisez un stockage d'applications haute performance avec Amazon FSx pour Lustre

Le pilote Container Storage Interface (CSI) Amazon FSx pour Lustre fournit une interface CSI qui permet aux clusters Amazon EKS de gérer le cycle de vie des systèmes de fichiers Amazon FSx pour Lustre. Pour plus d'informations, consultez le Guide de l'utilisateur Amazon FSx pour Lustre.

Pour plus d'informations sur le déploiement du pilote CSI Amazon FSx pour Lustre sur votre cluster Amazon EKS et la vérification de son bon fonctionnement, consultez the section called "Déploiement du pilote".

Déployez le pilote FSx for Lustre

Cette rubrique explique comment déployer le pilote CSI FSx for Lustre sur votre cluster Amazon EKS et vérifier qu'il fonctionne. Nous vous recommandons d'utiliser la dernière version du pilote. Pour les versions disponibles, consultez la matrice de compatibilité des spécifications CSI sur GitHub.

> **Note**
>
> Le pilote n'est pas pris en charge sur Fargate ou sur les nœuds hybrides Amazon EKS.

Pour une description détaillée des paramètres disponibles et des exemples complets illustrant les fonctionnalités du pilote, consultez le projet de pilote FSx pour Lustre Container Storage Interface (CSI) sur GitHub.

Conditions préalables

- Un cluster existant.
- Le module complémentaire Amazon FSx CSI Driver EKS nécessite l'agent EKS Pod Identity pour l'authentification. Sans ce composant, le module complémentaire échouera avec l'erreur `Amazon EKS Pod Identity agent is not installed in the cluster`, empêchant les opérations sur le volume. Installez l'agent Pod Identity avant ou après le déploiement du module complémentaire FSx CSI Driver. Pour de plus amples informations, veuillez consulter the section called "Configurer l'agent".
- Version `2.12.3` ou version ultérieure `1.27.160` ou version ultérieure de l'interface de ligne de AWS commande (AWS CLI) installée et configurée sur votre appareil ou AWS CloudShell. Pour vérifier votre version actuelle, utilisez `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Les gestionnaires de packages tels que `yum` Homebrew pour macOS ont souvent plusieurs versions de retard sur la dernière version de la AWS CLI. `apt-get` Pour installer la dernière version, consultez la section Installation et configuration rapide avec aws configure dans le Guide de l'utilisateur de l'interface de ligne de AWS commande. La version de la AWS CLI installée AWS CloudShell peut également avoir plusieurs versions de retard par rapport à la dernière version. Pour le mettre à jour, consultez la section Installation de la AWS CLI dans votre répertoire de base dans le guide de AWS CloudShell l'utilisateur.
- Version `0.215.0` ou ultérieure de l'outil de ligne de commande `eksctl` installée sur votre appareil ou AWS CloudShell. Pour installer ou mettre à jour `eksctl`, veuillez consulter Installation dans la documentation de `eksctl`.
- L'outil de ligne de commande `kubectl` est installé sur votre appareil ou AWS CloudShell. La version peut correspondre à celle utilisée par votre cluster Kubernetes, ou différer d'au plus une version mineure, qu'elle soit antérieure ou plus récente. Par exemple, si la version de votre cluster est `1.29`, vous pouvez utiliser la version `kubectl` `1.28`, `1.29` ou `1.30`. Pour installer ou mettre à niveau `kubectl`, veuillez consulter the section called "Configurer kubectl et eksctl".

Étape 1 : Créer un rôle IAM

Le plug-in Amazon FSx CSI nécessite des autorisations IAM pour passer des appels en votre AWS APIs nom.

> **Note**
>
> Les pods auront accès aux autorisations attribuées au rôle IAM, sauf si vous bloquez l'accès à IMDS. Pour de plus amples informations, veuillez consulter the section called "Bonnes pratiques".

La procédure suivante explique comment créer un rôle IAM et y associer la politique AWS gérée.

1. Créez un rôle IAM et associez la politique AWS gérée à l'aide de la commande suivante. Remplacez `my-cluster` par le nom du cluster que vous souhaitez utiliser. La commande déploie une AWS CloudFormation pile qui crée un rôle IAM et y attache la politique IAM.

   ```
   eksctl create iamserviceaccount \
       --name fsx-csi-controller-sa \
       --namespace kube-system \
       --cluster my-cluster \
       --role-name AmazonEKS_FSx_CSI_DriverRole \
       --role-only \
       --attach-policy-arn arn:aws: iam::aws:policy/AmazonFSxFullAccess \
       --approve
   ```

   Vous verrez plusieurs lignes de sortie lorsque le compte de service sera créé. Les dernières lignes de sortie sont similaires à celle de l'exemple suivant.

   ```
   [#]  1 task: {
       2 sequential sub-tasks: {
           create IAM role for serviceaccount "kube-system/fsx-csi-controller-sa",
           create serviceaccount "kube-system/fsx-csi-controller-sa",
       } }
   [#]  building iamserviceaccount stack "eksctl-my-cluster-addon-iamserviceaccount-
   kube-system-fsx-csi-controller-sa"
   [#]  deploying stack "eksctl-my-cluster-addon-iamserviceaccount-kube-system-fsx-csi-
   controller-sa"
   [#]  waiting for CloudFormation stack "eksctl-my-cluster-addon-iamserviceaccount-
   kube-system-fsx-csi-controller-sa"
   ```

```
[#]  created serviceaccount "kube-system/fsx-csi-controller-sa"
```

Notez le nom de la AWS CloudFormation pile qui a été déployée. Dans l'exemple de sortie ci-dessus, la pile est nommée `eksctl-my-cluster-addon-iamserviceaccount-kube-system-fsx-csi-controller-sa`.

Maintenant que vous avez créé le rôle IAM du pilote Amazon FSx CSI, vous pouvez passer à la section suivante. Lorsque vous déployez le module complémentaire avec ce rôle IAM, il crée et est configuré pour utiliser un compte de service nommé `fsx-csi-controller-sa`. Le compte de service est lié à un Kubernetes `clusterrole` auquel sont attribuées les autorisations Kubernetes requises.

Étape 2 : Installation du pilote Amazon FSx CSI

Nous vous recommandons d'installer le pilote Amazon FSx CSI via le module complémentaire Amazon EKS afin d'améliorer la sécurité et de réduire la charge de travail. Pour ajouter un module complémentaire Amazon EKS à votre cluster, voir the section called “Créer un module complémentaire”. Pour plus d'informations sur les modules complémentaires, voir the section called “Modules complémentaires Amazon EKS”.

Important

Les installations préexistantes du pilote Amazon FSx CSI dans le cluster peuvent provoquer l'échec de l'installation des modules complémentaires. Lorsque vous tentez d'installer la version complémentaire Amazon EKS alors qu'un pilote FSx CSI autre qu'EKS existe, l'installation échoue en raison de conflits de ressources. Utilisez le drapeau `OVERWRITE` lors de l'installation pour résoudre ce problème.

```
aws eks create-addon --addon-name aws-fsx-csi-driver --cluster-name my-cluster
 --resolve-conflicts OVERWRITE
```

Sinon, si vous souhaitez une installation autogérée du pilote Amazon FSx CSI, consultez la section Installation sur. GitHub

Étape 3 : déployer une classe de stockage, une demande de volume persistant et un exemple d'application

Cette procédure utilise le GitHub référentiel de pilotes CSI (FSx for Lustre Container Storage Interface) pour utiliser un volume configuré dynamiquement FSx pour Lustre.

1. Notez le groupe de sécurité de votre cluster. Vous pouvez le voir dans la AWS Management Console section Mise en réseau ci-dessous ou en utilisant la commande AWS CLI suivante. Remplacez `my-cluster` par le nom du cluster que vous souhaitez utiliser.

```
aws eks describe-cluster --name my-cluster --query
 cluster.resourcesVpcConfig.clusterSecurityGroupId
```

2. Créez un groupe de sécurité pour votre système de FSx fichiers Amazon conformément aux critères indiqués dans la section Groupes de sécurité Amazon VPC du guide de l'utilisateur Amazon FSx for Lustre. Pour le VPC, sélectionnez le VPC de votre cluster, comme indiqué dans la section Mise en réseau. Pour « les groupes de sécurité associés à vos clients Lustre », utilisez le groupe de sécurité de votre cluster. Vous pouvez juste laisser les règles sortantes pour autoriser Tout le trafic.
3. Téléchargez le manifeste de la classe de stockage à l'aide de la commande suivante.

```
curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-fsx-csi-driver/master/
examples/kubernetes/dynamic_provisioning/specs/storageclass.yaml
```

4. Modifiez la section des paramètres du fichier `storageclass.yaml`. Remplacez chaque valeur d'exemple par vos propres valeurs.

```
parameters:
  subnetId: subnet-0eabfaa81fb22bcaf
  securityGroupIds: sg-068000ccf82dfba88
  deploymentType: PERSISTENT_1
  automaticBackupRetentionDays: "1"
  dailyAutomaticBackupStartTime: "00:00"
  copyTagsToBackups: "true"
  perUnitStorageThroughput: "200"
  dataCompressionType: "NONE"
  weeklyMaintenanceStartTime: "7:09:00"
  fileSystemTypeVersion: "2.12"
```

- **`subnetId`**— L'ID de sous-réseau dans lequel le système de fichiers Amazon FSx for Lustre doit être créé. Amazon FSx for Lustre n'est pas pris en charge dans toutes les zones de disponibilité. Ouvrez la console Amazon FSx for Lustre https://console.aws.amazon.com/fsx/ à l'adresse pour vérifier que le sous-réseau que vous souhaitez utiliser se trouve dans une zone de disponibilité prise en charge. Le sous-réseau peut inclure vos nœuds, ou peut être un sous-réseau ou un VPC différent.
  - Vous pouvez vérifier la présence des sous-réseaux de nœuds dans le en AWS Management Console sélectionnant le groupe de nœuds dans la section Calculer.
  - Si le sous-réseau que vous spécifiez n'est pas le même que celui dans lequel vous avez des nœuds, vous VPCs devez être connecté et vous assurer que les ports nécessaires sont ouverts dans vos groupes de sécurité.
- **`securityGroupIds`** : l'ID du groupe de sécurité que vous avez créé pour le système de fichiers.
- **`deploymentType`** (facultatif) : le type de déploiement du système de fichiers. Les valeurs valides sont `SCRATCH_1`, `SCRATCH_2`, `PERSISTENT_1` et `PERSISTENT_2`. Pour plus d'informations sur les types de déploiement, consultez Créer votre système de fichiers Amazon FSx for Lustre.
- autres paramètres (facultatif) — Pour plus d'informations sur les autres paramètres, voir Modifier StorageClass sur GitHub.

5. Créez le manifeste de la classe de stockage.

```
kubectl apply -f storageclass.yaml
```

L'exemple qui suit illustre un résultat.

```
storageclass.storage.k8s.io/fsx-sc created
```

6. Téléchargez le manifeste de revendication de volume persistant.

```
curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-fsx-csi-driver/master/
examples/kubernetes/dynamic_provisioning/specs/claim.yaml
```

7. (Facultatif) Modifiez le fichier `claim.yaml`. Remplacez `1200Gi` par l'une des valeurs d'incrément ci-dessous, en fonction de vos besoins de stockage et du `deploymentType` que vous avez sélectionné à l'étape précédente.

```
storage: 1200Gi
```

- SCRATCH_2 et PERSISTENT – `1.2 TiB`, `2.4 TiB` ou des incréments de 2,4 Tio au-dessus de 2,4 Tio.
- SCRATCH_1 – `1.2 TiB`, `2.4 TiB`, `3.6 TiB` ou des incréments de 3,6 Tio au-dessus de 3,6 Tio.

8. Créez la revendication de volume persistant.

```
kubectl apply -f claim.yaml
```

L'exemple qui suit illustre un résultat.

```
persistentvolumeclaim/fsx-claim created
```

9. Vérifiez que le système de fichiers est approvisionné.

```
kubectl describe pvc
```

L'exemple qui suit illustre un résultat.

```
Name:          fsx-claim
Namespace:     default
StorageClass:  fsx-sc
Status:        Bound
[...]
```

> **Note**
>
> Le `Status` peut indiquer `Pending` pendant 5-10 minutes avant de devenir `Bound`. Ne passez pas à l'étape suivante tant que le `Status` n'est pas `Bound`. Si le `Status` affiche `Pending` pendant plus de 10 minutes, utilisez des messages d'avertissement dans les `Events` comme référence afin de résoudre tout problème.

10. Déployez un exemple d'application

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-sigs/aws-fsx-csi-
driver/master/examples/kubernetes/dynamic_provisioning/specs/pod.yaml
```

11. Vérifiez que l'exemple d'application est en cours d'exécution.

```
kubectl get pods
```

L'exemple qui suit illustre un résultat.

```
NAME      READY   STATUS              RESTARTS   AGE
fsx-app   1/1     Running             0          8s
```

12. Vérifiez que le système de fichiers est correctement monté par l'application.

```
kubectl exec -ti fsx-app -- df -h
```

L'exemple qui suit illustre un résultat.

```
Filesystem                  Size  Used Avail Use% Mounted on
overlay                      80G  4.0G   77G   5% /
tmpfs                        64M     0   64M   0% /dev
tmpfs                       3.8G     0  3.8G   0% /sys/fs/cgroup
192.0.2.0@tcp:/abcdef01     1.1T  7.8M  1.1T   1% /data
/dev/nvme0n1p1               80G  4.0G   77G   5% /etc/hosts
shm                          64M     0   64M   0% /dev/shm
tmpfs                       6.9G   12K  6.9G   1% /run/secrets/kubernetes.io/
serviceaccount
tmpfs                       3.8G     0  3.8G   0% /proc/acpi
tmpfs                       3.8G     0  3.8G   0% /sys/firmware
```

13. Vérifiez que les données ont été écrites dans le système de fichiers FSx for Lustre par l'exemple d'application.

```
kubectl exec -it fsx-app -- ls /data
```

L'exemple qui suit illustre un résultat.

```
out.txt
```

Cet exemple de sortie signifie que l'exemple d'application a réussi à écrire le fichier `out.txt` dans le système de fichiers.

> **Note**
>
> Avant de supprimer le cluster, assurez-vous de supprimer le système de fichiers FSx for Lustre. Pour plus d'informations, consultez les ressources de nettoyage dans le guide de l'utilisateur de FSx for Lustre.

Optimisation des performances FSx pour Lustre

Lorsque vous utilisez FSx for Lustre avec Amazon EKS, vous pouvez optimiser les performances en appliquant les réglages Lustre lors de l'initialisation du nœud. L'approche recommandée consiste à utiliser les données utilisateur du modèle de lancement afin de garantir une configuration cohérente sur tous les nœuds.

Ces accordages comprennent :

- Optimisations du réseau et du RPC
- Gestion du module Lustre
- Réglages LRU (Lock Resource Unit)
- Paramètres de contrôle du cache client
- Commandes RPC pour OST et MDC

Pour obtenir des instructions détaillées sur la mise en œuvre de ces réglages de performance :

- Pour optimiser les performances des nœuds standard (non EFA), consultez the section called "Optimize (non-EFA)" pour obtenir le script complet qui peut être ajouté aux données utilisateur de votre modèle de lancement.
- Pour optimiser les performances des nœuds compatibles EFA, consultez the section called "Optimiser (EFA)".

Optimisez les performances FSx d'Amazon for Lustre sur les nœuds (EFA)

Cette rubrique explique comment configurer le réglage d'Elastic Fabric Adapter (EFA) avec Amazon EKS et FSx Amazon for Lustre.

> **Note**
>
> - Pour plus d'informations sur la création et le déploiement du pilote CSI FSx for Lustre, consultezthe section called "Déploiement du pilote".
> - Pour optimiser les nœuds standard sans EFA, voir the section called "Optimize (non-EFA)".

Étape 1. Créer un cluster EKS

Créez un cluster à l'aide du fichier de configuration fourni :

```
# Create cluster using efa-cluster.yaml
eksctl create cluster -f efa-cluster.yaml
```

Exemple `efa-cluster.yaml` :

```
#efa-cluster.yaml

apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: csi-fsx
  region: us-east-1
  version: "1.30"

iam:
  withOIDC: true

availabilityZones: ["us-east-1a", "us-east-1d"]

managedNodeGroups:
  - name: my-efa-ng
    instanceType: c6gn.16xlarge
    minSize: 1
    desiredCapacity: 1
```

```
    maxSize: 1
    availabilityZones: ["us-east-1b"]
    volumeSize: 300
    privateNetworking: true
    amiFamily: Ubuntu2204
    efaEnabled: true
    preBootstrapCommands:
      - |
        #!/bin/bash
        eth_intf="$(/sbin/ip -br -4 a sh | grep $(hostname -i)/ | awk '{print $1}')"
        efa_version=$(modinfo efa | awk '/^version:/ {print $2}' | sed 's/[^0-9.]//g')
        min_efa_version="2.12.1"

        if [[ "$(printf '%s\n' "$min_efa_version" "$efa_version" | sort -V | head -
n1)" != "$min_efa_version" ]]; then
            sudo curl -O https://efa-installer.amazonaws.com/aws-efa-
installer-1.37.0.tar.gz
            tar -xf aws-efa-installer-1.37.0.tar.gz && cd aws-efa-installer
            echo "Installing EFA driver"
            sudo apt-get update && apt-get upgrade -y
            sudo apt install -y pciutils environment-modules libnl-3-dev libnl-
route-3-200 libnl-route-3-dev dkms
            sudo ./efa_installer.sh -y
            modinfo efa
        else
            echo "Using EFA driver version $efa_version"
        fi

        echo "Installing Lustre client"
        sudo wget -O - https://fsx-lustre-client-repo-public-keys.s3.amazonaws.com/fsx-
ubuntu-public-key.asc | gpg --dearmor | sudo tee /usr/share/keyrings/fsx-ubuntu-public-
key.gpg > /dev/null
        sudo echo "deb [signed-by=/usr/share/keyrings/fsx-ubuntu-public-key.gpg]
 https://fsx-lustre-client-repo.s3.amazonaws.com/ubuntu jammy main" > /etc/apt/
sources.list.d/fsxlustreclientrepo.list
        sudo apt update | tail
        sudo apt install -y lustre-client-modules-$(uname -r) amazon-ec2-utils | tail
        modinfo lustre

        echo "Loading Lustre/EFA modules..."
        sudo /sbin/modprobe lnet
        sudo /sbin/modprobe kefalnd ipif_name="$eth_intf"
        sudo /sbin/modprobe ksocklnd
        sudo lnetctl lnet configure
```

```
        echo "Configuring TCP interface..."
        sudo lnetctl net del --net tcp 2> /dev/null
        sudo lnetctl net add --net tcp --if $eth_intf

        # For P5 instance type which supports 32 network cards,
        # by default add 8 EFA interfaces selecting every 4th device (1 per PCI bus)
        echo "Configuring EFA interface(s)..."
        instance_type="$(ec2-metadata --instance-type | awk '{ print $2 }')"
        num_efa_devices="$(ls -1 /sys/class/infiniband | wc -l)"
        echo "Found $num_efa_devices available EFA device(s)"

        if [[ "$instance_type" == "p5.48xlarge" || "$instance_type" ==
 "p5e.48xlarge" ]]; then
           for intf in $(ls -1 /sys/class/infiniband | awk 'NR % 4 == 1'); do
               sudo lnetctl net add --net efa --if $intf --peer-credits 32
          done
        else
        # Other instances: Configure 2 EFA interfaces by default if the instance
 supports multiple network cards,
        # or 1 interface for single network card instances
        # Can be modified to add more interfaces if instance type supports it
            sudo lnetctl net add --net efa --if $(ls -1 /sys/class/infiniband | head -
n1) --peer-credits 32
            if [[ $num_efa_devices -gt 1 ]]; then
               sudo lnetctl net add --net efa --if $(ls -1 /sys/class/infiniband | tail
 -n1) --peer-credits 32
            fi
        fi

        echo "Setting discovery and UDSP rule"
        sudo lnetctl set discovery 1
        sudo lnetctl udsp add --src efa --priority 0
        sudo /sbin/modprobe lustre

        sudo lnetctl net show
        echo "Added $(sudo lnetctl net show | grep -c '@efa') EFA interface(s)"
```

Étape 2. Créer un groupe de nœuds

Créer un groupe de nœuds activée pour EFA :

```
# Create node group using efa-ng.yaml
```

```
eksctl create nodegroup -f efa-ng.yaml
```

> **Important**
>
> === Ajustez ces valeurs en fonction de votre environnement dans la section `# 5. Mount FSx filesystem`.

```
FSX_DNS="<your-fsx-filesystem-dns>" # Needs to be adjusted.
MOUNT_NAME="<your-mount-name>" # Needs to be adjusted.
MOUNT_POINT="</your/mount/point>" # Needs to be adjusted.
```

===

Exemple `efa-ng.yaml` :

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: final-efa
  region: us-east-1

managedNodeGroups:
  - name: ng-1
    instanceType: c6gn.16xlarge
    minSize: 1
    desiredCapacity: 1
    maxSize: 1
    availabilityZones: ["us-east-1a"]
    volumeSize: 300
    privateNetworking: true
    amiFamily: Ubuntu2204
    efaEnabled: true
    preBootstrapCommands:
      - |
        #!/bin/bash
        exec 1> >(logger -s -t $(basename $0)) 2>&1


 ##############################################################################################
```

```
        #                                       Configuration Section
         #

 ##############################################################################################

        # File System Configuration
        FSX_DNS="<your-fsx-filesystem-dns>" # Needs to be adjusted.
        MOUNT_NAME="<your-mount-name>" # Needs to be adjusted.
        MOUNT_POINT="</your/mount/point>" # Needs to be adjusted.

        # Lustre Tuning Parameters
        LUSTRE_LRU_MAX_AGE=600000
        LUSTRE_MAX_CACHED_MB=64
        LUSTRE_OST_MAX_RPC=32
        LUSTRE_MDC_MAX_RPC=64
        LUSTRE_MDC_MOD_RPC=50

        # File paths
        FUNCTIONS_SCRIPT="/usr/local/bin/lustre_functions.sh"
        TUNINGS_SCRIPT="/usr/local/bin/apply_lustre_tunings.sh"
        SERVICE_FILE="/etc/systemd/system/lustre-tunings.service"

        #EFA
        MIN_EFA_VERSION="2.12.1"

        # Function to check if a command was successful
        check_success() {
            if [ $? -eq 0 ]; then
                echo "SUCCESS: $1"
            else
                echo "FAILED: $1"
                return 1
            fi
        }

        echo "********Starting FSx for Lustre configuration********"

        # 1. Install Lustre client
        if grep -q '^ID=ubuntu' /etc/os-release; then
            echo "Detected Ubuntu, proceeding with Lustre setup..."
            # Add Lustre repository
            sudo wget -O - https://fsx-lustre-client-repo-public-keys.s3.amazonaws.com/
fsx-ubuntu-public-key.asc | sudo gpg --dearmor | sudo tee /usr/share/keyrings/fsx-
ubuntu-public-key.gpg > /dev/null
```

```
            echo "deb [signed-by=/usr/share/keyrings/fsx-ubuntu-public-key.gpg]
 https://fsx-lustre-client-repo.s3.amazonaws.com/ubuntu jammy main" | sudo tee /etc/
apt/sources.list.d/fsxlustreclientrepo.list

            sudo apt-get update
            sudo apt-get install -y lustre-client-modules-$(uname -r)
            sudo apt-get install -y lustre-client
        else
            echo "Not Ubuntu, exiting"
            exit 1
        fi

        check_success "Install Lustre client"

        # Ensure Lustre tools are in the PATH
        export PATH=$PATH:/usr/sbin

        # 2. Apply network and RPC tunings
        echo "********Applying network and RPC tunings********"
        if ! grep -q "options ptlrpc ptlrpcd_per_cpt_max" /etc/modprobe.d/
modprobe.conf; then
            echo "options ptlrpc ptlrpcd_per_cpt_max=64" | sudo tee -a /etc/modprobe.d/
modprobe.conf
            check_success "Set ptlrpcd_per_cpt_max"
        else
            echo "ptlrpcd_per_cpt_max already set in modprobe.conf"
        fi

        if ! grep -q "options ksocklnd credits" /etc/modprobe.d/modprobe.conf; then
            echo "options ksocklnd credits=2560" | sudo tee -a /etc/modprobe.d/
modprobe.conf
            check_success "Set ksocklnd credits"
        else
            echo "ksocklnd credits already set in modprobe.conf"
        fi

        # 3. Load Lustre modules
        manage_lustre_modules() {
            echo "Checking for existing Lustre modules..."
            if lsmod | grep -q lustre; then
                echo "Existing Lustre modules found."

                # Check for mounted Lustre filesystems
```

```
                echo "Checking for mounted Lustre filesystems..."
                if mount | grep -q "type lustre"; then
                    echo "Found mounted Lustre filesystems. Attempting to unmount..."
                    mounted_fs=$(mount | grep "type lustre" | awk '{print $3}')
                    for fs in $mounted_fs; do
                        echo "Unmounting $fs"
                        sudo umount $fs
                        check_success "Unmount filesystem $fs"
                    done
                else
                    echo "No Lustre filesystems mounted."
                fi

                # After unmounting, try to remove modules
                echo "Attempting to remove Lustre modules..."
                sudo lustre_rmmod
                if [ $? -eq 0 ]; then
                    echo "SUCCESS: Removed existing Lustre modules"
                else
                    echo "WARNING: Could not remove Lustre modules. They may still be
in use."
                    echo "Please check for any remaining Lustre processes or mounts."
                    return 1
                fi
            else
                echo "No existing Lustre modules found."
            fi

            echo "Loading Lustre modules..."
            sudo modprobe lustre
            check_success "Load Lustre modules" || exit 1

            echo "Checking loaded Lustre modules:"
            lsmod | grep lustre
        }

        # Managing Lustre kernel modules
        echo "********Managing Lustre kernel modules********"
        manage_lustre_modules

        # 4. Initializing Lustre networking
        echo "********Initializing Lustre networking********"
        sudo lctl network up
        check_success "Initialize Lustre networking" || exit 1
```

```
        # 4.5 EFA Setup and Configuration
        setup_efa() {
            echo "********Starting EFA Setup********"

            # Get interface and version information
            eth_intf="$(/sbin/ip -br -4 a sh | grep $(hostname -i)/ | awk '{print
 $1}')"
            efa_version=$(modinfo efa | awk '/^version:/ {print $2}' | sed 's/[^0-9.]//
g')
            min_efa_version=$MIN_EFA_VERSION

            # Install or verify EFA driver
            if [[ "$(printf '%s\n' "$min_efa_version" "$efa_version" | sort -V | head -
n1)" != "$min_efa_version" ]]; then
                echo "Installing EFA driver..."
                sudo curl -O https://efa-installer.amazonaws.com/aws-efa-
installer-1.37.0.tar.gz
                tar -xf aws-efa-installer-1.37.0.tar.gz && cd aws-efa-installer

                # Install dependencies
                sudo apt-get update && apt-get upgrade -y
                sudo apt install -y pciutils environment-modules libnl-3-dev libnl-
route-3-200 libnl-route-3-dev dkms

                # Install EFA
                sudo ./efa_installer.sh -y
                modinfo efa
            else
                echo "Using existing EFA driver version $efa_version"
            fi
        }

        configure_efa_network() {
            echo "********Configuring EFA Network********"

            # Load required modules
            echo "Loading network modules..."
            sudo /sbin/modprobe lnet
            sudo /sbin/modprobe kefalnd ipif_name="$eth_intf"
            sudo /sbin/modprobe ksocklnd

            # Initialize LNet
            echo "Initializing LNet..."
```

```
            sudo lnetctl lnet configure

            # Configure TCP interface
            echo "Configuring TCP interface..."
            sudo lnetctl net del --net tcp 2> /dev/null
            sudo lnetctl net add --net tcp --if $eth_intf

            # For P5 instance type which supports 32 network cards,
            # by default add 8 EFA interfaces selecting every 4th device (1 per PCI
bus)
            echo "Configuring EFA interface(s)..."
            instance_type="$(ec2-metadata --instance-type | awk '{ print $2 }')"
            num_efa_devices="$(ls -1 /sys/class/infiniband | wc -l)"
            echo "Found $num_efa_devices available EFA device(s)"

            if [[ "$instance_type" == "p5.48xlarge" || "$instance_type" ==
"p5e.48xlarge" ]]; then
                # P5 instance configuration
                for intf in $(ls -1 /sys/class/infiniband | awk 'NR % 4 == 1'); do
                    sudo lnetctl net add --net efa --if $intf --peer-credits 32
                done
            else
                # Standard configuration
                # Other instances: Configure 2 EFA interfaces by default if the
instance supports multiple network cards,
                # or 1 interface for single network card instances
                # Can be modified to add more interfaces if instance type supports it
                sudo lnetctl net add --net efa --if $(ls -1 /sys/class/infiniband |
head -n1) --peer-credits 32
                if [[ $num_efa_devices -gt 1 ]]; then
                    sudo lnetctl net add --net efa --if $(ls -1 /sys/class/infiniband |
tail -n1) --peer-credits 32
                fi
            fi

            # Configure discovery and UDSP
            echo "Setting up discovery and UDSP rules..."
            sudo lnetctl set discovery 1
            sudo lnetctl udsp add --src efa --priority 0
            sudo /sbin/modprobe lustre

            # Verify configuration
            echo "Verifying EFA network configuration..."
            sudo lnetctl net show
```

```
        echo "Added $(sudo lnetctl net show | grep -c '@efa') EFA interface(s)"
    }

    # Main execution
    setup_efa
    configure_efa_network

    # 5. Mount FSx filesystem
    if [ ! -z "$FSX_DNS" ] && [ ! -z "$MOUNT_NAME" ]; then
        echo "********Creating mount point********"
        sudo mkdir -p $MOUNT_POINT
        check_success "Create mount point"

        echo "Mounting FSx filesystem..."
        sudo mount -t lustre ${FSX_DNS}@tcp:/${MOUNT_NAME} ${MOUNT_POINT}
        check_success "Mount FSx filesystem"
    else
        echo "Skipping FSx mount as DNS or mount name is not provided"
    fi

    # 6. Applying Lustre performance tunings
    echo "********Applying Lustre performance tunings********"

    # Get number of CPUs
    NUM_CPUS=$(nproc)

    # Calculate LRU size (100 * number of CPUs)
    LRU_SIZE=$((100 * NUM_CPUS))

    #Apply LRU tunings
    echo "Apply LRU tunings"
    sudo lctl set_param ldlm.namespaces.*.lru_max_age=${LUSTRE_LRU_MAX_AGE}
    check_success "Set lru_max_age"
    sudo lctl set_param ldlm.namespaces.*.lru_size=$LRU_SIZE
    check_success "Set lru_size"

    # Client Cache Control
    sudo lctl set_param llite.*.max_cached_mb=${LUSTRE_MAX_CACHED_MB}
    check_success "Set max_cached_mb"

    # RPC Controls
    sudo lctl set_param osc.*OST*.max_rpcs_in_flight=${LUSTRE_OST_MAX_RPC}
    check_success "Set OST max_rpcs_in_flight"
```

```
        sudo lctl set_param mdc.*.max_rpcs_in_flight=${LUSTRE_MDC_MAX_RPC}
        check_success "Set MDC max_rpcs_in_flight"

        sudo lctl set_param mdc.*.max_mod_rpcs_in_flight=${LUSTRE_MDC_MOD_RPC}
        check_success "Set MDC max_mod_rpcs_in_flight"

        # 7. Verify all tunings
        echo "********Verifying all tunings********"

        # Function to verify parameter value
        verify_param() {
            local param=$1
            local expected=$2
            local actual=$3

            if [ "$actual" == "$expected" ]; then
                echo "SUCCESS: $param is correctly set to $expected"
            else
                echo "WARNING: $param is set to $actual (expected $expected)"
            fi
        }

        echo "Verifying all parameters:"

        # LRU tunings
        actual_lru_max_age=$(lctl get_param -n ldlm.namespaces.*.lru_max_age | head -1)
        verify_param "lru_max_age" "600000" "$actual_lru_max_age"

        actual_lru_size=$(lctl get_param -n ldlm.namespaces.*.lru_size | head -1)
        verify_param "lru_size" "$LRU_SIZE" "$actual_lru_size"

        # Client Cache
        actual_max_cached_mb=$(lctl get_param -n llite.*.max_cached_mb | grep
"max_cached_mb:" | awk '{print $2}')
        verify_param "max_cached_mb" "64" "$actual_max_cached_mb"

        # RPC Controls
        actual_ost_rpcs=$(lctl get_param -n osc.*OST*.max_rpcs_in_flight | head -1)
        verify_param "OST max_rpcs_in_flight" "32" "$actual_ost_rpcs"

        actual_mdc_rpcs=$(lctl get_param -n mdc.*.max_rpcs_in_flight | head -1)
        verify_param "MDC max_rpcs_in_flight" "64" "$actual_mdc_rpcs"

        actual_mdc_mod_rpcs=$(lctl get_param -n mdc.*.max_mod_rpcs_in_flight | head -1)
```

```
        verify_param "MDC max_mod_rpcs_in_flight" "50" "$actual_mdc_mod_rpcs"

        # Network and RPC configurations from modprobe.conf
        actual_ptlrpc=$(grep "ptlrpc ptlrpcd_per_cpt_max" /etc/modprobe.d/modprobe.conf
| awk '{print $3}')
        verify_param "ptlrpcd_per_cpt_max" "ptlrpcd_per_cpt_max=64" "$actual_ptlrpc"

        actual_ksocklnd=$(grep "ksocklnd credits" /etc/modprobe.d/modprobe.conf | awk
'{print $3}')
        verify_param "ksocklnd credits" "credits=2560" "$actual_ksocklnd"

        # 8. Setup persistence
        setup_persistence() {
            # Create functions file
            cat << EOF > $FUNCTIONS_SCRIPT
        #!/bin/bash

        apply_lustre_tunings() {
            local NUM_CPUS=\$(nproc)
            local LRU_SIZE=\$((100 * NUM_CPUS))

            echo "Applying Lustre performance tunings..."
            lctl set_param ldlm.namespaces.*.lru_max_age=$LUSTRE_LRU_MAX_AGE
            lctl set_param ldlm.namespaces.*.lru_size=\$LRU_SIZE
            lctl set_param llite.*.max_cached_mb=$LUSTRE_MAX_CACHED_MB
            lctl set_param osc.*OST*.max_rpcs_in_flight=$LUSTRE_OST_MAX_RPC
            lctl set_param mdc.*.max_rpcs_in_flight=$LUSTRE_MDC_MAX_RPC
            lctl set_param mdc.*.max_mod_rpcs_in_flight=$LUSTRE_MDC_MOD_RPC
        }
        EOF

            # Create tuning script
            cat << EOF > $TUNINGS_SCRIPT
        #!/bin/bash
        exec 1> >(logger -s -t \$(basename \$0)) 2>&1

        source $FUNCTIONS_SCRIPT

        # Function to check if Lustre is mounted
        is_lustre_mounted() {
            mount | grep -q "type lustre"
        }

        # Function to mount Lustre
```

```
        mount_lustre() {
            echo "Mounting Lustre filesystem..."
            mkdir -p $MOUNT_POINT
            mount -t lustre ${FSX_DNS}@tcp:/${MOUNT_NAME} $MOUNT_POINT
            return \$?
        }

        # Main execution
        # Try to mount if not already mounted
        if ! is_lustre_mounted; then
            echo "Lustre filesystem not mounted, attempting to mount..."
            mount_lustre
        fi

        # Wait for successful mount (up to 5 minutes)
        for i in {1..30}; do
            if is_lustre_mounted; then
                echo "Lustre filesystem mounted, applying tunings..."
                apply_lustre_tunings
                exit 0
            fi
            echo "Waiting for Lustre filesystem to be mounted... (attempt $i/30)"
            sleep 10
        done

        echo "Timeout waiting for Lustre filesystem mount"
        exit 1
        EOF

        # Create systemd service

        # Create systemd directory if it doesn't exist
        sudo mkdir -p /etc/systemd/system/

            # Create service file directly for Ubuntu
            cat << EOF > $SERVICE_FILE
        [Unit]
        Description=Apply Lustre Performance Tunings
        After=network.target remote-fs.target

        [Service]
        Type=oneshot
        ExecStart=/bin/bash -c 'source $FUNCTIONS_SCRIPT && $TUNINGS_SCRIPT'
        RemainAfterExit=yes
```

```
        [Install]
        WantedBy=multi-user.target
        EOF


            # Make scripts executable and enable service
            sudo chmod +x $FUNCTIONS_SCRIPT
            sudo chmod +x $TUNINGS_SCRIPT
            systemctl enable lustre-tunings.service
            systemctl start lustre-tunings.service
        }

        echo "********Setting up persistent tuning********"
        setup_persistence

        echo "FSx for Lustre configuration completed."
```

(Facultatif) Étape 3. Vérifier EFA

Accéder au nœud via SSH :

```
# Get instance ID from EKS console or {aws} CLI
ssh -i /path/to/your-key.pem ec2-user@<node-internal-ip>
```

Vérifier EFA :

```
sudo lnetctl net show
```

Consultez les journaux de configuration :

```
sudo cat /var/log/cloud-init-output.log
```

Voici un exemple de sortie attendue pour `lnetctl net show` :

```
net:
    - net type: tcp
      ...
    - net type: efa
      local NI(s):
        - nid: xxx.xxx.xxx.xxx@efa
```

```
          status: up
```

Exemple de déploiement

a. Créer claim.yaml

```
#claim.yaml

apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: fsx-claim-efa
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: ""
  resources:
    requests:
      storage: 4800Gi
  volumeName: fsx-pv
```

Appliquez la réclamation :

```
kubectl apply -f claim.yaml
```

b. Créer pv.yaml

Mettre à jour le :`<replaceable-placeholders>`

```
#pv.yaml

apiVersion: v1
kind: PersistentVolume
metadata:
  name: fsx-pv
spec:
  capacity:
    storage: 4800Gi
  volumeMode: Filesystem
  accessModes:
    - ReadWriteMany
  mountOptions:
```

```
    - flock
  persistentVolumeReclaimPolicy: Recycle
  csi:
    driver: fsx.csi.aws.com
    volumeHandle: fs-<1234567890abcdef0>
    volumeAttributes:
      dnsname: fs-<1234567890abcdef0>.fsx.us-east-1.amazonaws.com
      mountname: <abcdef01>
```

Appliquez le volume persistant :

```
kubectl apply -f pv.yaml
```

c. Créer pod.yaml

```
#pod.yaml

apiVersion: v1
kind: Pod
metadata:
  name: fsx-efa-app
spec:
  containers:
  - name: app
    image: amazonlinux:2
    command: ["/bin/sh"]
    args: ["-c", "while true; do dd if=/dev/urandom bs=100M count=20 > data/test_file;
 sleep 10; done"]
    resources:
      requests:
        vpc.amazonaws.com/efa: 1
      limits:
        vpc.amazonaws.com/efa: 1
    volumeMounts:
    - name: persistent-storage
      mountPath: /data
  volumes:
  - name: persistent-storage
    persistentVolumeClaim:
      claimName: fsx-claim-efa
```

Appliquez le pod :

```
kubectl apply -f pod.yaml
```

Commandes de vérification supplémentaires

Vérifiez que Pod monte et écrit sur le système de fichiers :

```
kubectl exec -ti fsx-efa-app -- df -h | grep data
# Expected output:
# <192.0.2.0>@tcp:/<abcdef01>  4.5T  1.2G  4.5T   1% /data

kubectl exec -ti fsx-efa-app -- ls /data
# Expected output:
# test_file
```

Connectez-vous au nœud en SSH pour vérifier que le trafic passe par EFA :

```
sudo lnetctl net show -v
```

Le résultat attendu affichera les interfaces EFA avec les statistiques de trafic.

Informations connexes

- the section called "Déploiement du pilote"
- the section called "Optimize (non-EFA)"
- Amazon FSx pour Lustre Performance
- Elastic Fabric Adapter (EFA)

Optimisez les performances FSx d'Amazon for Lustre sur les nœuds (non EFA)

Vous pouvez optimiser les performances d'Amazon FSx for Lustre en appliquant des paramètres de réglage lors de l'initialisation du nœud à l'aide des données utilisateur du modèle de lancement.

> **Note**
>
> - Pour plus d'informations sur la création et le déploiement du pilote CSI FSx for Lustre, consultezthe section called "Déploiement du pilote". Pour optimiser les performances avec les nœuds compatibles EFA, consultez the section called "Optimiser (EFA)".

Pourquoi utiliser les données utilisateur du modèle de lancement ?

- Applique automatiquement les réglages lors de l'initialisation du nœud.
- Assure une configuration cohérente sur tous les nœuds.
- Élimine le besoin de configuration manuelle des nœuds.

Aperçu de l'exemple de script

Le script d'exemple défini dans cette rubrique effectue les opérations suivantes :

`# 1. Install Lustre client`

- Détecte automatiquement la version de votre système d'exploitation Amazon Linux (AL).
- Installe le package client Lustre approprié.

`# 2. Apply network and RPC tunings`

- Ensembles `ptlrpcd_per_cpt_max=64` pour le traitement RPC parallèle.
- Configure `ksocklnd credits=2560` pour optimiser les tampons réseau.

`# 3. Load Lustre modules`

- Supprime en toute sécurité les modules Lustre existants, le cas échéant.
- Gère le démontage des systèmes de fichiers existants.
- Charge les modules Lustre récents.

`# 4. Lustre Network Initialization`

- Initialise la configuration réseau Lustre.

- Configure les paramètres réseau requis.

 # 5. Mount FSx filesystem

- Vous devez ajuster les valeurs pour votre environnement dans cette section.

 # 6. Apply tunings

- Réglages LRU (Lock Resource Unit) :
  - `lru_max_age=600000`
  - `lru_size` calculé en fonction du nombre de processeurs
- Contrôle du cache client : `max_cached_mb=64`
- Contrôles RPC :
  - OST `max_rpcs_in_flight=32`
  - MDC `max_rpcs_in_flight=64`
  - MDC `max_mod_rpcs_in_flight=50`

 # 7. Verify tunings

- Vérifie tous les réglages appliqués.
- Signale le succès ou l'avertissement pour chaque paramètre.

 # 8. Setup persistence

- Vous devez également ajuster les valeurs pour votre environnement dans cette section.
- Détecte automatiquement la version de votre système d'exploitation (AL2023) pour déterminer le `Systemd` service à appliquer.
- Le système démarre.
- `Systemd` démarre le service `lustre-tunings` (en raison de `WantedBy=multi-user.target`).
- Le service exécute `apply_lustre_tunings.sh` qui :
  - Vérifie si le système de fichiers est monté.
  - Monte le système de fichiers s'il n'est pas déjà monté.
  - Attend que le montage soit réussi (jusqu'à cinq minutes).

  - Applique les paramètres de réglage après un montage réussi.
- Les paramètres restent actifs jusqu'au redémarrage.
- Le service se termine après l'exécution du script.
  - Systemd marque le service comme « actif (fermé) ».
- Le processus se répète au prochain redémarrage.

Création d'un modèle de lancement

1. Ouvrez la EC2 console Amazon à l'adresse https://console.aws.amazon.com/ec2/.
2. Choisissez Lancer modèles.
3. Choisissez Créer un modèle de lancement.
4. Dans les détails avancés, repérez la section Données utilisateur.
5. Collez le script ci-dessous, en mettant à jour tout ce qui est nécessaire.

> **Important**
>
> Ajustez ces valeurs en fonction de votre environnement dans les sections `# 5. Mount FSx filesystem` et dans la fonction `setup_persistence()` de `apply_lustre_tunings.sh` dans la section `# 8. Setup persistence` :
>
> ```
> FSX_DNS="<your-fsx-filesystem-dns>" # Needs to be adjusted.
> MOUNT_NAME="<your-mount-name>" # Needs to be adjusted.
> MOUNT_POINT="</your/mount/point>" # Needs to be adjusted.
> ```

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="==MYBOUNDARY=="
--==MYBOUNDARY==
Content-Type: text/x-shellscript; charset="us-ascii"
#!/bin/bash
exec 1> >(logger -s -t $(basename $0)) 2>&1
# Function definitions
check_success() {
    if [ $? -eq 0 ]; then
        echo "SUCCESS: $1"
    else
```

```
        echo "FAILED: $1"
        return 1
    fi
}
apply_tunings() {
    local NUM_CPUS=$(nproc)
    local LRU_SIZE=$((100 * NUM_CPUS))
    local params=(
        "ldlm.namespaces.*.lru_max_age=600000"
        "ldlm.namespaces.*.lru_size=$LRU_SIZE"
        "llite.*.max_cached_mb=64"
        "osc.*OST*.max_rpcs_in_flight=32"
        "mdc.*.max_rpcs_in_flight=64"
        "mdc.*.max_mod_rpcs_in_flight=50"
    )
    for param in "${params[@]}"; do
        lctl set_param $param
        check_success "Set ${param%%=*}"
    done
}
verify_param() {
    local param=$1
    local expected=$2
    local actual=$3

    if [ "$actual" == "$expected" ]; then
        echo "SUCCESS: $param is correctly set to $expected"
    else
        echo "WARNING: $param is set to $actual (expected $expected)"
    fi
}
verify_tunings() {
    local NUM_CPUS=$(nproc)
    local LRU_SIZE=$((100 * NUM_CPUS))
    local params=(
        "ldlm.namespaces.*.lru_max_age:600000"
        "ldlm.namespaces.*.lru_size:$LRU_SIZE"
        "llite.*.max_cached_mb:64"
        "osc.*OST*.max_rpcs_in_flight:32"
        "mdc.*.max_rpcs_in_flight:64"
        "mdc.*.max_mod_rpcs_in_flight:50"
    )
    echo "Verifying all parameters:"
    for param in "${params[@]}"; do
```

```
        name="${param%%:*}"
        expected="${param#*:}"
        actual=$(lctl get_param -n $name | head -1)
        verify_param "${name##*.}" "$expected" "$actual"
    done
}
setup_persistence() {
    # Create functions file
    cat << 'EOF' > /usr/local/bin/lustre_functions.sh
#!/bin/bash
apply_lustre_tunings() {
    local NUM_CPUS=$(nproc)
    local LRU_SIZE=$((100 * NUM_CPUS))

    echo "Applying Lustre performance tunings..."
    lctl set_param ldlm.namespaces.*.lru_max_age=600000
    lctl set_param ldlm.namespaces.*.lru_size=$LRU_SIZE
    lctl set_param llite.*.max_cached_mb=64
    lctl set_param osc.*OST*.max_rpcs_in_flight=32
    lctl set_param mdc.*.max_rpcs_in_flight=64
    lctl set_param mdc.*.max_mod_rpcs_in_flight=50
}
EOF
    # Create tuning script
    cat << 'EOF' > /usr/local/bin/apply_lustre_tunings.sh
#!/bin/bash
exec 1> >(logger -s -t $(basename $0)) 2>&1
# Source the functions
source /usr/local/bin/lustre_functions.sh
# FSx details
FSX_DNS="<your-fsx-filesystem-dns>" # Needs to be adjusted.
MOUNT_NAME="<your-mount-name>" # Needs to be adjusted.
MOUNT_POINT="</your/mount/point>" # Needs to be adjusted.
# Function to check if Lustre is mounted
is_lustre_mounted() {
    mount | grep -q "type lustre"
}
# Function to mount Lustre
mount_lustre() {
    echo "Mounting Lustre filesystem..."
    mkdir -p ${MOUNT_POINT}
    mount -t lustre ${FSX_DNS}@tcp:/${MOUNT_NAME} ${MOUNT_POINT}
    return $?
}
```

```
# Main execution
# Try to mount if not already mounted
if ! is_lustre_mounted; then
    echo "Lustre filesystem not mounted, attempting to mount..."
    mount_lustre
fi
# Wait for successful mount (up to 5 minutes)
for i in {1..30}; do
    if is_lustre_mounted; then
        echo "Lustre filesystem mounted, applying tunings..."
        apply_lustre_tunings
        exit 0
    fi
    echo "Waiting for Lustre filesystem to be mounted... (attempt $i/30)"
    sleep 10
done
echo "Timeout waiting for Lustre filesystem mount"
exit 1
EOF
    # Create systemd service
    cat << 'EOF' > /etc/systemd/system/lustre-tunings.service
[Unit]
Description=Apply Lustre Performance Tunings
After=network.target remote-fs.target
StartLimitIntervalSec=0
[Service]
Type=oneshot
ExecStart=/usr/local/bin/apply_lustre_tunings.sh
RemainAfterExit=yes
Restart=on-failure
RestartSec=30
[Install]
WantedBy=multi-user.target
EOF
    chmod +x /usr/local/bin/lustre_functions.sh
    chmod +x /usr/local/bin/apply_lustre_tunings.sh
    systemctl enable lustre-tunings.service
    systemctl start lustre-tunings.service
}
echo "Starting FSx for Lustre configuration..."
# 1. Install Lustre client
if grep -q 'VERSION="2"' /etc/os-release; then
    amazon-linux-extras install -y lustre
elif grep -q 'VERSION="2023"' /etc/os-release; then
```

```
    dnf install -y lustre-client
fi
check_success "Install Lustre client"
# 2. Apply network and RPC tunings
export PATH=$PATH:/usr/sbin
echo "Applying network and RPC tunings..."
if ! grep -q "options ptlrpc ptlrpcd_per_cpt_max" /etc/modprobe.d/modprobe.conf; then
    echo "options ptlrpc ptlrpcd_per_cpt_max=64" | tee -a /etc/modprobe.d/
modprobe.conf
    echo "options ksocklnd credits=2560" | tee -a /etc/modprobe.d/modprobe.conf
fi
# 3. Load Lustre modules
modprobe lustre
check_success "Load Lustre modules" || exit 1
# 4. Lustre Network Initialization
lctl network up
check_success "Initialize Lustre networking" || exit 1
# 5. Mount FSx filesystem
FSX_DNS="<your-fsx-filesystem-dns>" # Needs to be adjusted.
MOUNT_NAME="<your-mount-name>" # Needs to be adjusted.
MOUNT_POINT="</your/mount/point>" # Needs to be adjusted.
if [ ! -z "$FSX_DNS" ] && [ ! -z "$MOUNT_NAME" ]; then
    mkdir -p $MOUNT_POINT
    mount -t lustre ${FSX_DNS}@tcp:/${MOUNT_NAME} ${MOUNT_POINT}
    check_success "Mount FSx filesystem"
fi
# 6. Apply tunings
apply_tunings
# 7. Verify tunings
verify_tunings
# 8. Setup persistence
setup_persistence
echo "FSx for Lustre configuration completed."
--==MYBOUNDARY==--
```

6. Lorsque vous créez des groupes de nœuds Amazon EKS, sélectionnez ce modèle de lancement. Pour de plus amples informations, veuillez consulter the section called “Créer”.

Informations connexes

- the section called “Déploiement du pilote”
- the section called “Optimiser (EFA)”

- Amazon FSx pour Lustre Performance

Utilisez un stockage d'applications haute performance avec FSx pour NetApp ONTAP

NetApp Trident offre une orchestration dynamique du stockage à l'aide d'un pilote compatible avec l'interface CSI (Container Storage Interface). Cela permet aux clusters Amazon EKS de gérer le cycle de vie des volumes persistants (PV) basés sur les systèmes de fichiers Amazon FSx pour NetApp ONTAP. Veuillez noter que le pilote Amazon FSx pour NetApp ONTAP CSI n'est pas compatible avec les nœuds hybrides Amazon EKS. Pour commencer, consultez la section Utilisation de Trident avec Amazon FSx pour NetApp ONTAP dans la documentation NetApp Trident.

Amazon FSx for NetApp ONTAP est un service de stockage qui vous permet de lancer et d'exécuter des systèmes de fichiers ONTAP entièrement gérés dans le cloud. ONTAP est la technologie de système de fichiers de NetApp qui offre un ensemble largement adopté de fonctionnalités d'accès aux données et de gestion des données. FSx for ONTAP offre les fonctionnalités, les performances et les API des systèmes de fichiers NetApp sur site, avec l'agilité, la capacité de mise à l'échelle et la simplicité d'un service AWS entièrement géré. Pour plus d'informations, consultez le Guide de l'utilisateur FSx for ONTAP.

Important

Si vous utilisez Amazon FSx pour NetApp ONTAP avec le pilote Amazon EBS CSI pour provisionner des volumes EBS, vous devez spécifier dans le fichier `multipath.conf` de ne pas utiliser de périphériques EBS. Pour connaître les méthodes prises en charge, consultez la Liste noire des fichiers de configuration. Voici un exemple.

```
defaults {
       user_friendly_names yes
       find_multipaths no
     }
     blacklist {
       device {
         vendor "NVME"
         product "Amazon Elastic Block Store"
       }
     }
```

Utilisez le stockage de données avec Amazon FSx pour OpenZFS

Amazon FSx pour OpenZFS est un service de stockage de fichiers entièrement géré qui facilite le transfert de données vers AWS à partir de serveurs de fichiers ZFS sur site ou d'autres serveurs de fichiers basés sur Linux. Vous pouvez le faire sans modifier le code de votre application ni la façon dont vous gérez les données. Il offre un stockage de fichiers hautement fiable, évolutif, efficace et riche en fonctionnalités basé sur le système de fichiers open source OpenZFS. Il associe ces fonctionnalités à l'agilité, à l'évolutivité et à la simplicité d'un service AWS entièrement géré. Pour plus d'informations, consultez le Guide de l'utilisateur Amazon FSx pour OpenZFS.

Le pilote FSx pour OpenZFS Container Storage Interface (CSI) fournit une interface CSI qui permet aux clusters Amazon EKS de gérer le cycle de vie des volumes FSx pour OpenZFS. Veuillez noter que le pilote CSI Amazon FSx pour OpenZFS n'est pas compatible avec les nœuds hybrides Amazon EKS. Pour déployer le pilote FSx pour OpenZFS CSI sur votre cluster Amazon EKS, consultez aws-fsx-openzfs-csi-driver sur GitHub.

Minimisez le temps de latence avec Amazon File Cache

Amazon File Cache est un cache haut débit entièrement géré sur AWS qui est utilisé pour traiter les données des fichiers, quel que soit leur emplacement de stockage. Amazon File Cache charge automatiquement les données dans le cache lorsque vous y accédez pour la première fois et les publie lorsqu'elles ne sont pas utilisées. Pour de plus amples informations, veuillez consulter le Guide de l'utilisateur Amazon File Cache.

Le pilote Amazon File Cache Container Storage Interface (CSI) fournit une interface CSI qui permet aux clusters Amazon EKS de gérer le cycle de vie des systèmes de fichiers Amazon. Veuillez noter que le pilote CSI Amazon File Cache n'est pas compatible avec les nœuds hybrides Amazon EKS. Pour déployer le pilote CSI Amazon File Cache sur votre cluster Amazon EKS, consultez aws-file-cache-csi-driver sur GitHub.

Accéder aux objets Amazon S3 avec le pilote CSI Mountpoint pour Amazon S3

Avec le Pilote Container Storage Interface (CSI) Mountpoint pour Amazon S3, vos applications Kubernetes peuvent accéder aux objets Amazon S3 via une interface de système de fichiers, ce qui permet d'obtenir un débit global élevé sans modifier le code de l'application. Créé sur Mountpoint

pour Amazon S3, le pilote CSI présente un compartiment Amazon S3 comme un volume accessible par les conteneurs dans Amazon EKS et les clusters Kubernetes autogérés.

Considérations

- Le pilote CSI Mountpoint pour Amazon S3 n'est actuellement pas compatible avec les images de conteneurs Windows.
- Le pilote CSI Mountpoint pour Amazon S3 n'est actuellement pas compatible avec les nœuds hybrides Amazon EKS.
- Le pilote CSI Mountpoint pour Amazon S3 ne prend pas en charge AWS Fargate. Cependant, les conteneurs qui s'exécutent dans Amazon EC2 (avec Amazon EKS ou une installation Kubernetes personnalisée) sont pris en charge.
- Le pilote CSI Mountpoint pour Amazon S3 ne prend en charge que l'approvisionnement statique. L'approvisionnement dynamique, ou la création de nouveaux compartiments, n'est pas pris en charge.

  > **Note**
  >
  > L'approvisionnement statique fait référence à l'utilisation d'un compartiment Amazon S3 existant qui est spécifié comme `bucketName` dans les `volumeAttributes` de l'objet `PersistentVolume`. Pour plus d'informations, consultez Approvisionnement statique sur GitHub.

- Les volumes montés avec le pilote Mountpoint pour Amazon S3 CSI ne prennent pas en charge toutes les fonctionnalités du système de fichiers POSIX. Pour plus d'informations sur le comportement du système de fichiers, consultez Comportement du système de fichiers Mountpoint pour Amazon S3 sur GitHub.

Pour plus d'informations sur le déploiement du pilote, consultez the section called "Déploiement du pilote". Pour plus d'informations sur la suppression du pilote, consultez the section called "Supprimer le pilote".

Déployer le pilote Mountpoint pour Amazon S3

Avec le Pilote Container Storage Interface (CSI) Mountpoint pour Amazon S3, vos applications Kubernetes peuvent accéder aux objets Amazon S3 via une interface de système de fichiers, ce qui permet d'obtenir un débit global élevé sans modifier le code de l'application.

Cette procédure vous montrera comment déployer le pilote CSI Mountpoint pour Amazon S3 pour Amazon EKS. Avant de continuer, consultez les Considérations.

Conditions préalables

- Un fournisseur IAM (AWS Identity and Access Management) (IAM) OpenID Connect (OIDC) existant pour votre cluster. Pour déterminer si vous en avez déjà un, ou pour en créer un, consultez the section called "Fournisseur IAM OIDC".
- Version 2.12.3 ou ultérieure de la AWS CLI installée et configurée sur votre appareil ou. AWS CloudShell
- L'outil de ligne de commande `kubectl` est installé sur votre appareil ou AWS CloudShell. La version peut correspondre à celle utilisée par votre cluster Kubernetes, ou différer d'au plus une version mineure, qu'elle soit antérieure ou plus récente. Par exemple, si la version de votre cluster est `1.29`, vous pouvez utiliser la version `kubectl` `1.28`, `1.29` ou `1.30`. Pour installer ou mettre à niveau `kubectl`, veuillez consulter the section called "Configurer kubectl et eksctl".

Étape 1 : créer une politique IAM

Le pilote CSI Mountpoint pour Amazon S3 nécessite des autorisations Amazon S3 pour interagir avec votre système de fichiers. Cette section montre comment créer une politique IAM qui accorde les autorisations nécessaires.

L'exemple de politique suivant respecte les recommandations d'autorisation IAM pour Mountpoint. Vous pouvez également utiliser la politique AWS gérée AmazonS3 FullAccess, mais cette politique gérée accorde plus d'autorisations que ce qui est nécessaire pour Mountpoint.

Pour plus d'informations sur les autorisations recommandées pour Mountpoint, consultez la section Autorisations Mountpoint IAM sur. GitHub

1. Ouvrez la console IAM à https://console.aws.amazon.com/iam/ l'adresse.
2. Dans le volet de navigation de gauche, choisissez Politiques.
3. Sur la page Politiques, choisissez Créer une politique.
4. Dans Éditeur de politique, choisissez JSON.
5. Sous Éditeur de politique, copiez et collez ce qui suit :

> **Important**
>
> Remplacez `amzn-s3-demo-bucket1` par votre propre nom de compartiment Amazon S3.

```
{
   "Version":"2012-10-17",
   "Statement": [
        {
            "Sid": "MountpointFullBucketAccess",
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket1"
            ]
        },
        {
            "Sid": "MountpointFullObjectAccess",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:PutObject",
                "s3:AbortMultipartUpload",
                "s3:DeleteObject"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket1/*"
            ]
        }
   ]
}
```

Les compartiments de répertoires, introduits avec la classe de stockage Amazon S3 Express One Zone, utilisent un mécanisme d'authentification différent de celui des compartiments à usage général. Au lieu d'utiliser les actions `s3:*`, vous devez utiliser l'action `s3express:CreateSession`. Pour plus d'informations sur les compartiments de répertoires, consultez Compartiments de répertoires dans le Guide de l'utilisateur Amazon S3.

Vous trouverez ci-dessous un exemple de politique de privilège minimal que vous pouvez utiliser pour un compartiment répertoire.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3express:CreateSession",
            "Resource": "arn:aws:s3express:us-west-2:111122223333:bucket/amzn-s3-
demo-bucket1--usw2-az1--x-s3"
        }
    ]
}
```

6. Choisissez Suivant.
7. Sur la page Vérifier et créer, donnez un nom à votre politique. Cet exemple de démonstration utilise le nom `AmazonS3CSIDriverPolicy`.
8. Choisissez Create Policy (Créer une politique).

Étape 2 : Créer un rôle IAM

Le pilote CSI Mountpoint pour Amazon S3 nécessite des autorisations Amazon S3 pour interagir avec votre système de fichiers. Cette section montre comment créer un rôle IAM pour déléguer ces autorisations. Pour créer ce rôle, vous pouvez utiliser l'un des outils suivants :

- the section called "eksctl"
- the section called "AWS Management Console"
- the section called "AWS CLI"

> **Note**
>
> La politique IAM `AmazonS3CSIDriverPolicy` a été créée dans la section précédente.

eksctl

Pour créer votre rôle IAM du pilote CSI Mountpoint pour Amazon S3 avec **`eksctl`**

Pour créer le rôle IAM et le compte de service Kubernetes, exécutez les commandes suivantes. Ces commandes associent également la politique IAM `AmazonS3CSIDriverPolicy` au rôle, annotent le compte de service Kubernetes (`s3-csi-controller-sa`) avec le nom d'Amazon Resource Name (ARN) du rôle IAM et ajoutent le nom du compte de service Kubernetes à la politique d'approbation pour le rôle IAM.

```
CLUSTER_NAME=my-cluster
REGION=region-code
ROLE_NAME=AmazonEKS_S3_CSI_DriverRole
POLICY_ARN=AmazonEKS_S3_CSI_DriverRole_ARN
eksctl create iamserviceaccount \
    --name s3-csi-driver-sa \
    --namespace kube-system \
    --cluster $CLUSTER_NAME \
    --attach-policy-arn $POLICY_ARN \
    --approve \
    --role-name $ROLE_NAME \
    --region $REGION \
    --role-only
```

AWS Management Console

1. Ouvrez la console IAM à https://console.aws.amazon.com/iam/ l'adresse.
2. Dans le volet de navigation de gauche, choisissez Rôles.
3. Sur la page Rôles, choisissez Créer un rôle.
4. Sur la page Select trusted entity (Sélectionnez une entité de confiance), procédez comme suit :
   a. Dans la section Trusted entity type (Type d'entité de confiance), sélectionnez Web identity (Identité web).
   b. Pour Fournisseur d'identité, choisissez l'URL du fournisseur OpenID Connect pour votre cluster (comme indiqué sous Présentation dans Amazon EKS).

      Si ce n' URLs est pas le cas, passez en revue les conditions préalables.
   c. Pour Audience, choisissez `sts.amazonaws.com`.
   d. Choisissez Suivant.
5. Sur la page Add permissions (Ajouter des autorisations), procédez comme suit :
   a. Dans le champ Politiques de filtrage, saisissez la politique d'AmazonS3CSIDriver.

> **Note**
>
> Cette politique a été créée dans la section précédente.

b. Cochez la case à gauche du résultat `AmazonS3CSIDriverPolicy` renvoyé par la recherche.

c. Choisissez Suivant.

6. Sur la page Name, review, and create (Nommer, vérifier et créer), procédez comme suit :

a. Dans Nom du rôle, entrez un nom unique pour votre rôle, tel que DriverRole AmazoneKS_S3_CSI_.

b. Sous Ajouter des balises (Facultatif), ajoutez des métadonnées au rôle en attachant les identifications sous forme de paires clé-valeur. Pour plus d'informations sur l'utilisation des balises dans IAM, consultez la rubrique Balisage des ressources IAM dans le Guide de l'utilisateur IAM.

c. Choisissez Créer un rôle.

7. Une fois le rôle créé, choisissez le rôle dans la console pour l'ouvrir et le modifier.

8. Sélectionnez l'onglet Trust relationships (Relations d'approbation), puis Edit trust policy (Modifier la politique d'approbation).

9. Trouvez la ligne qui ressemble à ce qui suit :

```
"oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE:aud":
 "sts.amazonaws.com"
```

Ajoutez une virgule à la fin de la ligne précédente, puis ajoutez la ligne suivante. Remplacez *`region-code`* par la AWS région dans laquelle se trouve votre cluster. Remplacez *`EXAMPLED539D4633E53DE1B71EXAMPLE`* par l'ID du fournisseur OIDC de votre cluster.

```
"oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE:sub":
 "system:serviceaccount:kube-system:s3-csi-driver-sa"
```

10. Assurez-vous que l'opérateur `Condition` est défini sur `"StringEquals"`.

11. Sélectionnez Update Trust Policy (Mettre à jour la politique d'approbation) pour terminer.

AWS CLI

1. Affichez l'URL du fournisseur OIDC de votre cluster. Remplacez *my-cluster* par le nom de votre cluster. Si la sortie de la commande est None, consultez Prérequis.

```
aws eks describe-cluster --name my-cluster --query "cluster.identity.oidc.issuer" --
output text
```

   L'exemple qui suit illustre un résultat.

```
https://oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE
```

2. Créez le rôle IAM, en accordant au compte de service Kubernetes l'action AssumeRoleWithWebIdentity.

   a. Copiez le contenu suivant dans un fichier nommé aws-s3-csi-driver-trust-policy.json. Remplacez *111122223333* par votre ID de compte. Remplacez *EXAMPLED539D4633E53DE1B71EXAMPLE* et *region-code* par les valeurs renvoyées par l'étape précédente.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::111122223333:oidc-provider/oidc.eks.us-
east-1.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "oidc.eks.us-east-1.amazonaws.com/id/
EXAMPLED539D4633E53DE1B71EXAMPLE:sub": "system:serviceaccount:kube-system:s3-csi-
driver-sa",
          "oidc.eks.us-east-1.amazonaws.com/id/
EXAMPLED539D4633E53DE1B71EXAMPLE:aud": "sts.amazonaws.com"
        }
      }
    }
  ]
```

```
}
```

b. Créez le rôle. Vous pouvez changer le nom de *AmazonEKS_S3_CSI_DriverRole*. Dans ce cas, veillez à le changer également dans les étapes suivantes.

```
aws iam create-role \
  --role-name AmazonEKS_S3_CSI_DriverRole \
  --assume-role-policy-document file://"aws-s3-csi-driver-trust-policy.json"
```

3. Attachez la politique IAM créée précédemment au rôle à l'aide de la commande suivante.

```
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::aws:policy/AmazonS3CSIDriverPolicy \
  --role-name AmazonEKS_S3_CSI_DriverRole
```

> **Note**
>
> La politique IAM `AmazonS3CSIDriverPolicy` a été créée dans la section précédente.

4. Ignorez cette étape si vous installez le pilote en tant que module complémentaire Amazon EKS. Pour les installations autogérées du pilote, créez des comptes de service Kubernetes annotés avec l'ARN du rôle IAM que vous avez créé.

a. Enregistrez le contenu suivant dans un fichier nommé `mountpoint-s3-service-account.yaml`. Remplacez *111122223333* par votre ID de compte.

```
---
apiVersion: v1
kind: ServiceAccount
metadata:
  labels:
    app.kubernetes.io/name: aws-mountpoint-s3-csi-driver
  name: mountpoint-s3-csi-controller-sa
  namespace: kube-system
  annotations:
    eks.amazonaws.com/role-arn: arn:aws: iam::111122223333:role/
AmazonEKS_S3_CSI_DriverRole
```

b. Créez le compte de service Kubernetes sur votre cluster. Le compte de service Kubernetes (`mountpoint-s3-csi-controller-sa`) est annoté avec le nom du rôle IAM que vous avez créé. *AmazonEKS_S3_CSI_DriverRole*

```
kubectl apply -f mountpoint-s3-service-account.yaml
```

> **Note**
>
> Lorsque vous déployez le plugin dans cette procédure, il crée et est configuré pour utiliser un compte de service nommé `s3-csi-driver-sa`.

Étape 3 : installer le pilote CSI Mountpoint pour Amazon S3

Vous pouvez installer le pilote CSI Mountpoint pour Amazon S3 via le module complémentaire Amazon EKS. Vous pouvez utiliser les outils suivants pour ajouter le module complémentaire à votre cluster :

- the section called "eksctl"
- the section called "AWS Management Console"
- the section called "AWS CLI"

Vous pouvez également installer le pilote CSI Mountpoint pour Amazon S3 en tant qu'installation autogérée. Pour savoir comment procéder à une installation autogérée, consultez Installation sur GitHub.

À partir de la version `v1.8.0`, vous pouvez configurer les rejets à tolérer pour les pods du pilote CSI. Pour ce faire, spécifiez un ensemble personnalisé de rejets à tolérer avec `node.tolerations` ou tolérez tous les rejets avec `node.tolerateAllTaints`. Pour plus d'informations, consultez Rejets et tolérances dans la documentation de Kubernetes.

eksctl

Pour ajouter le module complémentaire CSI Amazon S3 à l'aide de **`eksctl`**

Exécutez la commande suivante. Remplacez *`my-cluster`* par le nom de votre cluster, *`111122223333`* par l'ID de votre compte et *`AmazonEKS_S3_CSI_DriverRole`* par le nom du rôle IAM créé précédemment.

```
eksctl create addon --name aws-mountpoint-s3-csi-driver --cluster my-cluster \
```

```
  --service-account-role-arn arn:aws: iam::111122223333:role/
AmazonEKS_S3_CSI_DriverRole --force
```

Si vous supprimez l'option *--force* et si des paramètres du module complémentaire Amazon EKS entrent en conflit avec vos paramètres existants, alors la mise à jour du module complémentaire Amazon EKS échoue et vous recevez un message d'erreur pour vous aider à résoudre le conflit. Avant de spécifier cette option, assurez-vous que le module complémentaire Amazon EKS ne gère pas les paramètres que vous devez gérer, car ces paramètres sont remplacés par cette option. Pour plus d'informations sur les autres options de ce paramètre, consultez Modules complémentaires dans la documentation `eksctl`. Pour plus d'informations sur la gestion des champs Amazon EKS Kubernetes, consultez the section called "Champs que vous pouvez personnaliser".

Vous pouvez personnaliser `eksctl` à l'aide de fichiers de configuration. Pour plus d'informations, consultez Utilisation des valeurs de configuration dans la documentation `eksctl`. L'exemple suivant montre comment tolérer tous les rejets.

```
# config.yaml
...

addons:
- name: aws-mountpoint-s3-csi-driver
  serviceAccountRoleARN: arn:aws: iam::111122223333:role/AmazonEKS_S3_CSI_DriverRole
  configurationValues: |-
    node:
      tolerateAllTaints: true
```

AWS Management Console

1. Ouvrez la console Amazon EKS.
2. Dans le volet de navigation de gauche, choisissez Clusters.
3. Choisissez le nom du cluster pour lequel vous voulez configurer le module complémentaire Mountpoint pour Amazon S3 CSI.
4. Choisissez l'onglet Modules complémentaires.
5. Choisissez Obtenez plus de modules complémentaires.
6. Sur la page Sélectionner des modules complémentaires, procédez comme suit :
   a. Dans la section Modules complémentaires Amazon EKS, sélectionnez la case à cocher Pilote CSI Mountpoint pour Amazon S3.
   b. Choisissez Suivant.

7. Sur la page Configurer les paramètres des modules complémentaires sélectionnés, procédez comme suit :
   a. Sélectionnez la Version que vous souhaitez utiliser.
   b. Pour Sélectionner un rôle IAM, sélectionnez le nom d'un rôle IAM auquel vous avez associé la politique IAM du pilote CSI Mountpoint pour Amazon S3.
   c. (Facultatif) Mettez à jour la Méthode de résolution des conflits après avoir développé les Paramètres de configuration facultatifs. Si vous sélectionnez Remplacer, un ou plusieurs paramètres du module complémentaire existant peuvent être remplacés par les paramètres du module complémentaire Amazon EKS. Si vous n'activez pas cette option et qu'il y a un conflit avec vos paramètres existants, l'opération échoue. Vous pouvez utiliser le message d'erreur qui en résulte pour résoudre le conflit. Avant de sélectionner cette option, veuillez vous assurer que le module complémentaire Amazon EKS ne gère pas les paramètres que vous devez gérer vous-même.
   d. (Facultatif) Veuillez configurer les tolérances dans le champ Valeurs de configuration après avoir développé les Paramètres de configuration facultatifs.
   e. Choisissez Suivant.
8. Sur la page Vérifier et ajouter, choisissez Créer. Une fois l'installation du module complémentaire terminée, vous pouvez voir le module complémentaire installé.

AWS CLI

Pour ajouter le module complémentaire CSI Mountpoint pour Amazon S3 à l'aide de la CLI AWS

Exécutez la commande suivante. Remplacez *my-cluster* par le nom de votre cluster, *111122223333* par l'ID de votre compte, et *AmazonEKS_S3_CSI_DriverRole* par le nom du rôle créé précédemment.

```
aws eks create-addon --cluster-name my-cluster --addon-name aws-mountpoint-s3-csi-
driver \
  --service-account-role-arn arn:aws: iam::111122223333:role/
AmazonEKS_S3_CSI_DriverRole
```

Vous pouvez personnaliser la commande à l'aide de l'indicateur `--configuration-values`. L'exemple alternatif suivant montre comment tolérer tous les rejets.

```
aws eks create-addon --cluster-name my-cluster --addon-name aws-mountpoint-s3-csi-
driver \
```

```
  --service-account-role-arn arn:aws: iam::111122223333:role/
AmazonEKS_S3_CSI_DriverRole \
  --configuration-values '{"node":{"tolerateAllTaints":true}}'
```

Étape 4 : configurer Mountpoint pour Amazon S3

Dans la plupart des cas, vous pouvez configurer Mountpoint pour Amazon S3 avec uniquement un nom de compartiment. Pour obtenir des instructions sur la configuration de Mountpoint pour Amazon S3, consultez Configuration de Mountpoint pour Amazon S3 sur. GitHub

Étape 5 : déployer un exemple d'application

Vous pouvez déployer l'allocation statique au pilote sur un compartiment Amazon S3 existant. Pour plus d'informations, consultez la section Provisionnement statique activé GitHub.

Supprimer le module complémentaire Mountpoint pour Amazon S3 pour Amazon EKS

Vous disposez de deux options pour supprimer le pilote CSI Mountpoint pour Amazon S3.

- Conserver le logiciel complémentaire sur votre cluster : cette option supprime la gestion de tous les paramètres par Amazon EKS. Elle supprime également la possibilité pour Amazon EKS de vous informer des mises à jour et de mettre automatiquement à jour le module complémentaire Amazon EKS après le lancement d'une mise à jour. Toutefois, elle préserve le logiciel complémentaire sur votre cluster. Cette option fait du module complémentaire une installation autogérée plutôt qu'un module complémentaire Amazon EKS. Avec cette option, il n'y a pas de durée d'indisponibilité pour le module complémentaire. Les commandes dans cette procédure utilisent cette option.
- Supprimer entièrement le logiciel complémentaire de votre cluster : nous vous recommandons de supprimer le module complémentaire Amazon EKS de votre cluster uniquement si aucune ressource de votre cluster n'en dépend. Pour effectuer cette option, supprimez `--preserve` à partir de la commande que vous utilisez dans cette procédure.

Si le module complémentaire est associé à un compte IAM, celui-ci n'est pas supprimé.

Vous pouvez utiliser les outils suivants pour supprimer le module complémentaire Amazon S3 CSI :

- the section called "eksctl"
- the section called "AWS Management Console"

- the section called "AWS CLI"

eksctl

Pour supprimer le module complémentaire CSI Amazon S3 à l'aide de **`eksctl`**

Remplacez *`my-cluster`* par le nom de votre cluster, puis exécutez la commande suivante.

```
eksctl delete addon --cluster my-cluster --name aws-mountpoint-s3-csi-driver --preserve
```

AWS Management Console

1. Ouvrez la console Amazon EKS.
2. Dans le volet de navigation de gauche, choisissez Clusters.
3. Choisissez le nom du cluster pour lequel vous souhaitez supprimer le module complémentaire CSI Amazon EBS.
4. Choisissez l'onglet Modules complémentaires.
5. Sélectionnez Pilote CSI Mountpoint pour Amazon S3.
6. Sélectionnez Remove (Supprimer).
7. Dans la boîte de dialogue de confirmation Supprimer : aws-mountpoint-s3-csi-driver, procédez comme suit :
   a. Si vous voulez qu'Amazon EKS cesse de gérer les paramètres du module complémentaire, sélectionnez Conserver sur le cluster. Faites ceci si vous voulez retenir le logiciel du module complémentaire sur votre cluster. Ceci afin que vous puissiez gérer vous-même tous les paramètres du module complémentaire.
   b. Saisissez `aws-mountpoint-s3-csi-driver`.
   c. Sélectionnez Remove (Retirer).

AWS CLI

Pour supprimer le module complémentaire CSI Amazon S3 à l'aide de l'AWS CLI

Remplacez *`my-cluster`* par le nom de votre cluster, puis exécutez la commande suivante.

```
aws eks delete-addon --cluster-name my-cluster --addon-name aws-mountpoint-s3-csi-
driver --preserve
```

Activer la fonctionnalité d'instantané pour les volumes CSI

La fonctionnalité Instantané permet de réaliser des copies ponctuelles de vos données. Pour que cette fonctionnalité fonctionne dans Kubernetes, vous avez besoin à la fois d'un pilote CSI prenant en charge les instantanés (tel que le pilote Amazon EBS CSI) et d'un contrôleur d'instantanés CSI. Le contrôleur de snapshot est disponible soit sous forme de module complémentaire géré par Amazon EKS, soit sous forme d'installation autogérée.

Voici quelques éléments à prendre en compte lors de l'utilisation du contrôleur d'instantané CSI.

- Le contrôleur d'instantané doit être installé aux côtés d'un pilote CSI doté d'une fonctionnalité d'instantané. Pour obtenir les instructions d'installation du pilote CSI Amazon EBS, consultez the section called "Amazon EBS".
- Kubernetes ne prend pas en charge les instantanés de volumes fournis via la migration CSI, tels que les volumes Amazon EBS utilisant un `StorageClass` avec un provisionneur `kubernetes.io/aws-ebs`. Les volumes doivent être créés avec un `StorageClass` qui fait référence au provisionneur de pilote CSI, `ebs.csi.aws.com`.
- Le mode automatique Amazon EKS n'inclut pas le contrôleur de snapshots. La capacité de stockage du mode automatique EKS est compatible avec le contrôleur de snapshots.

Nous vous recommandons d'installer le contrôleur d'instantané CSI via le module complémentaire géré Amazon EKS. Ce module complémentaire comprend les définitions de ressources personnalisées (CRD) nécessaires pour créer et gérer des instantanés sur Amazon EKS. Pour ajouter un module complémentaire Amazon EKS à votre cluster, voir the section called "Créer un module complémentaire". Pour plus d'informations sur les modules complémentaires, voir the section called "Modules complémentaires Amazon EKS".

Sinon, si vous souhaitez installer vous-même le contrôleur de snapshots CSI, consultez la section Utilisation dans le Kubernetes en amont `external-snapshotter` sur GitHub.

Configurer la mise en réseau pour les clusters Amazon EKS

Votre cluster Amazon EKS est créé dans un VPC. La mise en réseau des pods est assurée par le plug-in Amazon VPC Container Network Interface (CNI) pour les nœuds qui s'exécutent sur l'infrastructure AWS. Si vous utilisez des nœuds sur votre propre infrastructure, consultez the section called "Configurer CNI". Ce chapitre inclut les rubriques suivantes pour en savoir plus sur la mise en réseau de votre cluster.

Rubriques

- Ajouter un sous-réseau VPC existant à un cluster Amazon EKS à partir de la console de gestion
- Afficher les exigences réseau Amazon EKS pour les VPC et les sous-réseaux
- Création d'un VPC Amazon pour votre cluster Amazon EKS
- Voir les exigences relatives aux groupes de sécurité Amazon EKS pour les clusters
- Gérer les modules complémentaires réseau pour les clusters Amazon EKS

Ajouter un sous-réseau VPC existant à un cluster Amazon EKS à partir de la console de gestion

1. Accès à votre cluster dans la console de gestion
2. Dans l'onglet Mise en réseau, sélectionnez Gérer les ressources VPC
3. Dans le menu déroulant Sous-réseaux, sélectionnez des sous-réseaux supplémentaires dans le VPC de votre cluster.

Pour créer un sous-réseau VPC :

- Passez en revue les exigences relatives au sous-réseau EKS
- Consultez la section Créer un sous-réseau dans le Guide de l'utilisateur Amazon Virtual Private Cloud.

Afficher les exigences réseau Amazon EKS pour les VPC et les sous-réseaux

Lorsque vous créez un cluster, vous spécifiez un VPC et au moins deux sous-réseaux situés dans des zones de disponibilité différentes. Cette rubrique fournit une vue d'ensemble des exigences et considérations spécifiques à Amazon EKS qui sont requises pour le VPC et les sous-réseaux que vous utilisez avec votre cluster. Si vous ne disposez pas d'un VPC à utiliser avec Amazon EKS, consultez the section called "Création d'un VPC". Si vous créez un cluster local ou étendu sur AWS Outposts, consultez the section called "Créer un VPC et des sous-réseaux" plutôt cette rubrique. Le contenu de cette rubrique s'applique aux clusters Amazon EKS avec des nœuds hybrides. Pour connaître les exigences réseau supplémentaires pour les nœuds hybrides, consultez the section called "Préparez votre réseau".

Exigences et considérations requises pour le VPC

Lorsque vous créez un cluster, le VPC que vous spécifiez doit répondre aux exigences et aux considérations suivantes :

- Le VPC doit disposer d'un nombre suffisant d'adresses IP disponibles pour le cluster, pour tous les nœuds et pour les autres ressources Kubernetes que vous souhaitez créer. Si le VPC que vous voulez utiliser ne dispose pas d'un nombre suffisant d'adresses IP, essayez d'augmenter le nombre d'adresses IP disponibles.

  Pour ce faire, vous pouvez mettre à jour la configuration du cluster afin de modifier les sous-réseaux et les groupes de sécurité utilisés par le cluster. Vous pouvez effectuer la AWS Management Console mise à jour à partir de la dernière version de la AWS CLI et de `eksctl` la version `v0.164.0-rc.0` ou ultérieure. AWS CloudFormation Vous pouvez avoir besoin de procéder ainsi pour fournir aux sous-réseaux davantage d'adresses IP disponibles afin de réussir la mise à niveau d'une version de cluster.

  > **Important**
  >
  > Tous les sous-réseaux que vous ajoutez doivent appartenir au même ensemble AZs que celui fourni à l'origine lors de la création du cluster. Les nouveaux sous-réseaux doivent répondre à toutes les autres exigences, par exemple disposer d'un nombre suffisant d'adresses IP.
  > Par exemple, supposons que vous avez créé un cluster et spécifié quatre sous-réseaux. Dans l'ordre où vous les avez spécifiés, le premier sous-réseau se trouve dans la zone

de disponibilité `us-west-2a`, les deuxième et troisième sous-réseaux se trouvent dans la zone de disponibilité `us-west-2b` et le quatrième sous-réseau se trouve dans la zone de disponibilité `us-west-2c`. Si vous souhaitez modifier les sous-réseaux, vous devez fournir au moins un sous-réseau dans chacune des trois zones de disponibilité, et les sous-réseaux doivent se trouver dans le même VPC que les sous-réseaux d'origine.

Si vous avez besoin de plus d'adresses IP que les blocs CIDR du VPC, vous pouvez ajouter des blocs CIDR supplémentaires en associant des blocs CIDR (Routage inter-domaines sans classe) supplémentaires à votre VPC. Vous pouvez associer des blocs d'adresses CIDR privés (RFC 1918) et publics (non-RFC 1918) à votre VPC avant ou après la création de votre cluster.

Vous pouvez ajouter des nœuds qui utilisent le nouveau bloc CIDR immédiatement après l'avoir ajouté. Cependant, comme le plan de contrôle ne reconnaît le nouveau bloc CIDR qu'une fois la réconciliation terminée, il peut s'écouler jusqu'à une heure avant qu'un bloc CIDR que vous avez associé à un VPC soit reconnu par un cluster. Vous pouvez ensuite exécuter les `kubectl port-forward` commandes `kubectl attachkubectl cp`,`kubectl exec`,`kubectl logs`, et (ces commandes utilisent le`kubelet API`) pour les nœuds et les pods du nouveau bloc CIDR. De plus, si vous avez des pods qui fonctionnent comme un dorsal webhook, vous devez attendre que la réconciliation du plan de contrôle soit terminée.

- Évitez les chevauchements de plages d'adresses IP lorsque vous connectez votre cluster EKS à d'autres VPCs via Transit Gateway, le peering VPC ou d'autres configurations réseau. Des conflits CIDR se produisent lorsque le CIDR de service de votre cluster EKS chevauche le CIDR d'un VPC connecté. Dans ces scénarios, les adresses ServiceIP ont la priorité sur les ressources lorsqu'elles sont connectées VPCs à la même adresse IP, bien que le routage du trafic puisse devenir imprévisible et que les applications puissent ne pas se connecter aux ressources prévues.

  Pour éviter les conflits d'adresses CIDR, assurez-vous que le CIDR de votre service EKS ne se chevauche pas avec un VPC connecté CIDRs et conservez un enregistrement centralisé de toutes les assignations d'adresse CIDR. Si vous rencontrez des chevauchements CIDR, vous pouvez utiliser une passerelle de transit avec un VPC de services partagés. Pour plus d'informations, consultez Isolated VPCs with shared services et Modèles de conservation des adresses IP routables Amazon EKS VPC dans un réseau hybride. Reportez-vous également à la VPCs section Communication entre les deux parties de la page Considérations relatives aux VPC et aux sous-réseaux dans le guide des meilleures pratiques d'EKS.
- Si vous voulez que Kubernetes attribue des adresses `IPv6` aux pods et aux services, associez un bloc CIDR `IPv6` à votre VPC. Pour plus d'informations, consultez Associer un bloc IPv6 CIDR à

votre VPC dans le guide de l'utilisateur Amazon VPC. Vous ne pouvez pas utiliser d'adresses IPv6 avec des pods et des services s'exécutant sur des nœuds hybrides, et vous ne pouvez pas utiliser de nœuds hybrides avec des clusters configurés avec la famille d'adresses IP IPv6.

- Le VPC doit prend en charge le nom d'hôte DNS et la résolution DNS. Sinon, les nœuds ne pourront pas s'enregistrer sur votre cluster. Pour plus d'informations, consultez Attributs DNS pour votre VPC dans le guide de l'utilisateur d'Amazon VPC.
- Le VPC peut nécessiter que les points de terminaison du VPC utilisent. AWS PrivateLink Pour de plus amples informations, veuillez consulter the section called "Exigences et considérations requises pour les sous-réseaux".

Si vous avez créé un cluster avec Kubernetes 1.14 ou une version antérieure, Amazon EKS a ajouté l'identification suivante à votre VPC :

Clé	Valeur
kubernetes.io/cluster/ *my-cluster*	owned

Cette identification n'a été utilisée que par Amazon EKS. Vous pouvez supprimer l'identification sans affecter vos services. Il n'est pas utilisé avec les clusters de version 1.15 ou ultérieure.

Exigences et considérations requises pour les sous-réseaux

Lorsque vous créez un cluster, Amazon EKS doit créer entre 2 et 4 interfaces réseau Elastic dans les sous-réseaux que vous spécifiez. Ces interfaces réseau permettent la communication entre votre cluster et votre VPC. Ces interfaces réseau activent également les fonctionnalités de Kubernetes qui utilisent le. kubelet API Les connexions à l'kubeletAPI sont utilisées dans les kubectl port-forward commandes kubectl attach kubectl cpkubectl exec,kubectl logs,, et. Chaque interface réseau créée par Amazon EKS possède le texte Amazon EKS *cluster-name* dans sa description.

Amazon EKS peut créer ses interfaces réseau dans n'importe quel sous-réseau que vous spécifiez lorsque vous créez un cluster. Vous pouvez modifier les sous-réseaux dans lesquels Amazon EKS crée ses interfaces réseau après la création de votre cluster. Lorsque vous mettez à jour la version Kubernetes d'un cluster, Amazon EKS supprime les interfaces réseau d'origine qu'il a créées et crée de nouvelles interfaces réseau. Ces interfaces réseau peuvent être créées dans les mêmes sous-réseaux que les interfaces réseau d'origine ou dans des sous-réseaux différents des interfaces

réseau d'origine. Pour contrôler les sous-réseaux dans lesquels les interfaces réseau sont créées, vous pouvez limiter le nombre de sous-réseaux que vous spécifiez à seulement deux lorsque vous créez un cluster ou mettre à jour les sous-réseaux après la création du cluster.

Exigences requises pour les sous-réseaux des clusters

Les sous-réseaux que vous spécifiez lors de la création ou de la mise à jour d'un cluster doivent répondre aux exigences suivantes :

- Les sous-réseaux doivent comporter chacun au moins six adresses IP à utiliser par Amazon EKS. Toutefois, nous vous recommandons au moins 16 adresses IP.
- Les sous-réseaux doivent se trouver dans au moins deux zones de disponibilité différentes.
- Les sous-réseaux ne peuvent pas résider dans AWS Outposts ou AWS Wavelength. Toutefois, si vous les avez dans votre VPC, vous pouvez déployer des nœuds autogérés et des ressources Kubernetes sur ces types de sous-réseaux. Pour plus d'informations sur les nœuds autogérés, consultez the section called "Nœuds autogérés".
- Les sous-réseaux peuvent être publics ou privés. Toutefois, nous vous recommandons de spécifier des sous-réseaux privés, si possible. Un sous-réseau public est un sous-réseau dont la table de routage comprend une route vers une passerelle Internet, tandis qu'un sous-réseau privé est un sous-réseau dont la table de routage ne comprend pas de route vers une passerelle Internet.
- Les sous-réseaux ne peuvent pas résider dans les zones de disponibilité suivantes :

AWS Région	Nom de la région	Zone de disponibilité non autorisée IDs
`us-east-1`	US East (N. Virginia)	`use1-az3`
`us-west-1`	USA Ouest (Californie du Nord)	`usw1-az2`
`ca-central-1`	Canada (Centre)	`cac1-az3`

Utilisation de la famille d'adresses IP par composant

Le tableau suivant contient la famille d'adresses IP utilisée par chaque composant d'Amazon EKS. Vous pouvez utiliser une traduction d'adresses réseau (NAT) ou un autre système de compatibilité

pour vous connecter à ces composants à partir d'adresses IP sources dans des familles dont la valeur « Non » est indiquée dans une entrée du tableau.

Les fonctionnalités peuvent varier en fonction du paramètre de famille IP (`ipFamily`) du cluster. Ce paramètre modifie le type d'adresses IP utilisées pour le bloc CIDR que Kubernetes attribue aux services. Un cluster avec la valeur de réglage de IPv4 est appelé IPv4 cluster, et un cluster avec la valeur de réglage de IPv6 est appelé IPv6 cluster.

Composant	IPv4 adresses	IPv6 adresses	Adresses double pile
Point de terminaison public de l'API EKS	Oui[1,3]	Oui[1,3]	Oui[1,3]
Point de terminaison d'un VPC de l'API EKS	Oui	Non	Non
Point de terminaison public de l'API d'authentification EKS (identité du pod EKS)	Oui[1]	Oui[1]	Oui[1]
Point de terminaison d'un VPC de l'API d'authentification EKS (identité du pod EKS)	Oui[1]	Oui[1]	Oui[1]
Point de terminaison public du cluster Kubernetes `IPv4`[2]	Oui	Non	Non
Point de terminaison privé du cluster Kubernetes `IPv4`[2]	Oui	Non	Non
Point de terminaison public du cluster Kubernetes `IPv6`[2]	Oui[1,4]	Oui[1,4]	Oui[4]

Composant	IPv4 adresses	IPv6 adresses	Adresses double pile
Point de terminaison privé du cluster Kubernetes `IPv6`[2]	Oui[1,4]	Oui[1,4]	Oui[4]
Sous-réseaux du cluster Kubernetes	Oui[2]	Non	Oui[2]
Adresses IP principales des nœuds	Oui[2]	Non	Oui[2]
Plage CIDR du cluster pour les adresses IP des services	Oui[2]	Oui[2]	Non
Adresses IP des pods provenant du CNI d'un VPC	Oui[2]	Oui[2]	Non
Émetteur IRSA OIDC URLs	Oui[1,3]	Oui[1,3]	Oui[1,3]

Note

[1] Le point de terminaison est double pile avec des adresses `IPv4` et `IPv6`. Vos applications extérieures AWS, vos nœuds pour le cluster et vos pods à l'intérieur du cluster peuvent atteindre ce point de terminaison par l'un `IPv4` ou l'autre des moyens`IPv6`.

[2] Vous choisissez entre un cluster `IPv4` et un cluster `IPv6` dans le paramètre de famille IP (`ipFamily`) du cluster lorsque vous créez un cluster, et cela ne peut pas être modifié. Vous devez choisir un paramètre différent lorsque vous créez un autre cluster et migrez vos charges de travail.

[3] Le point de terminaison double pile a été introduit en août 2024. Pour utiliser les points de terminaison à double pile avec la AWS CLI, consultez la configuration des points de terminaison à double pile et FIPS dans le AWS SDKs guide de référence des outils et. Voici la liste des nouveaux points de terminaison :

Point de terminaison public de l'API EKS

```
eks.region.api.aws
```

Émetteur IRSA OIDC URLs

```
oidc-eks.region.api.aws
```

[4] Le point de terminaison du cluster double pile a été introduit en octobre 2024. EKS crée le point de terminaison suivant pour les nouveaux clusters créés après cette date et qui sélectionnent `IPv6` dans le paramètre de famille IP (ipFamily) du cluster :

public/private Point final du cluster EKS

```
eks-cluster.region.api.aws
```

Exigences requises pour les sous-réseaux des nœuds

Vous pouvez déployer des nœuds et des ressources Kubernetes sur les mêmes sous-réseaux que ceux que vous avez spécifiés lors de la création de votre cluster. Cependant, cela n'est pas nécessaire. En effet, vous pouvez également déployer des nœuds et des ressources Kubernetes sur des sous-réseaux que vous n'avez pas spécifiés lors de la création du cluster. Si vous déployez des nœuds sur différents sous-réseaux, Amazon EKS ne crée pas d'interfaces réseau de cluster dans ces sous-réseaux. Tout sous-réseau sur lequel vous déployez des nœuds et des ressources Kubernetes doit répondre aux exigences suivantes :

- Les sous-réseaux doivent disposer de suffisamment d'adresses IP disponibles pour déployer tous vos nœuds et ressources Kubernetes.
- Si vous voulez que Kubernetes attribue des adresses `IPv6` aux pod et aux services, vous devez disposer d'un bloc CIDR `IPv6` et d'un bloc CIDR `IPv4` associés à votre sous-réseau. Pour plus d'informations, consultez Associer un bloc IPv6 CIDR à votre sous-réseau dans le guide de l'utilisateur Amazon VPC. Les tables de routage associées aux sous-réseaux doivent inclure des routes vers les adresses `IPv4` et `IPv6`. Pour plus d'informations, veuillez consulter Routes dans le guide de l'utilisateur d'Amazon VPC. Les pods se voient attribuer uniquement une adresse `IPv6`. Toutefois, les interfaces réseau créées par Amazon EKS pour votre cluster et vos nœuds se voient attribuer une adresse `IPv4` et une adresse `IPv6`.

- Si vous avez besoin d'un accès entrant depuis Internet vers vos pods, assurez-vous de disposer d'au moins un sous-réseau public avec suffisamment d'adresses IP disponibles pour déployer des équilibreurs de charge et des entrées. Vous pouvez déployer des équilibreurs de charge sur des sous-réseaux publics. Les équilibreurs de charge peuvent équilibrer la charge vers des pods dans des sous-réseaux privés ou publics. Nous vous recommandons de déployer vos nœuds sur des sous-réseaux privés, si possible.
- Si vous envisagez de déployer des nœuds sur un sous-réseau public, ce sous-réseau doit attribuer automatiquement des adresses publiques `IPv4` ou des adresses `IPv6`. Si vous déployez des nœuds sur un sous-réseau privé doté d'un bloc d'adresse CIDR `IPv6` associé, le sous-réseau privé doit également attribuer automatiquement des adresses `IPv6`. Si vous avez utilisé le AWS CloudFormation modèle fourni par Amazon EKS pour déployer votre VPC après le 26 mars 2020, ce paramètre est activé. Si vous avez utilisé les modèles pour déployer votre VPC avant cette date ou si vous utilisez votre propre VPC, vous devez activer ce paramètre manuellement. Pour le modèle, consultez the section called "Création d'un VPC". Pour plus d'informations, consultez Modifier l'attribut d' IPv4 adressage public de votre sous-réseau et Modifier l'attribut d' IPv6 adressage de votre sous-réseau dans le guide de l'utilisateur Amazon VPC.
- Si le sous-réseau sur lequel vous déployez un nœud est un sous-réseau privé et que sa table de routage n'inclut pas de route vers un périphérique de traduction d'adresses réseau (NAT) () ou une passerelle de sortie uniquement (`IPv4`)`IPv6`, ajoutez des points de terminaison VPC à votre VPC. AWS PrivateLink Les points de terminaison VPC sont nécessaires pour tous les AWS services avec lesquels vos nœuds et vos pods ont besoin pour communiquer. Les exemples incluent Amazon ECR, Elastic Load Balancing CloudWatch, Amazon, AWS Security Token Service et Amazon Simple Storage Service (Amazon S3). Le point de terminaison doit inclure le sous-réseau dans lequel se trouvent les nœuds. Tous les AWS services ne prennent pas en charge les points de terminaison VPC. Pour plus d'informations, voir Qu'est-ce que c'est AWS PrivateLink ? et AWS des services qui s'intègrent à AWS PrivateLink. Pour obtenir une liste complète des exigences Amazon EKS, consultez the section called "Clusters privés".
- Si vous souhaitez déployer des équilibreurs de charge sur un sous-réseau, le sous-réseau doit comporter l'identification suivante :
  - Sous-réseaux privés

Clé	Valeur
`kubernetes.io/role/internal-elb`	`1`

- Sous-réseaux publics

Clé	Valeur
`kubernetes.io/role/elb`	`1`

Lorsqu'un cluster Kubernetes de version `1.18` ou antérieure a été créé, Amazon EKS a balisé la balise suivante à tous les sous-réseaux spécifiés.

Clé	Valeur
`kubernetes.io/cluster/` *`my-cluster`*	`shared`

Lorsque vous créez un nouveau cluster Kubernetes, Amazon EKS n'ajoute pas la balise à vos sous-réseaux. Si la balise se trouvait sur des sous-réseaux utilisés par un cluster dont la version était antérieure à `1.19`, elle n'a pas été automatiquement supprimée des sous-réseaux lors de la mise à jour du cluster vers une version plus récente. La version `2.1.1` ou antérieure du AWS Load Balancer Controller nécessite cette balise. Si vous utilisez une version plus récente du Load Balancer Controller, vous pouvez supprimer l'identification sans interrompre vos services. Pour plus d'informations sur le contrôleur, consultez the section called " AWS Contrôleur Load Balancer".

Si vous avez déployé un VPC à l'aide de l'un `eksctl` des modèles de AWS CloudFormation VPC Amazon EKS, les règles suivantes s'appliquent :

- À partir du 26 mars 2020 : les adresses `IPv4` publiques sont automatiquement attribuées par des sous-réseaux publics aux nouveaux nœuds déployés sur des sous-réseaux publics.
- Avant le 26 mars 2020 : les adresses `IPv4` publiques ne sont pas automatiquement attribuées par les sous-réseaux publics aux nouveaux nœuds déployés sur les sous-réseaux publics.

Cette modification affecte les nouveaux groupes de nœuds déployés sur des sous-réseaux publics de la manière suivante :

- Groupes de nœuds gérés : si le groupe de nœuds est déployé sur un sous-réseau public à compter du 22 avril 2020, l'attribution automatique d'adresses IP publiques doit être activée pour le

sous-réseau public. Pour plus d'informations, consultez la section Modification de l'attribut d' IPv4 adressage public de votre sous-réseau.

- Groupes de nœuds autogérés Linux, Windows ou Arm : si le groupe de nœuds est déployé sur un sous-réseau public à compter du 26 mars 2020, l'attribution automatique d'adresses IP publiques doit être activée pour le sous-réseau public. Sinon, les nœuds doivent être lancés avec une adresse IP publique. Pour plus d'informations, consultez Modification de l'attribut d' IPv4 adressage public pour votre sous-réseau ou Affectation d'une IPv4 adresse publique lors du lancement de l'instance.

Exigences et considérations requises pour les sous-réseaux partagés

Vous pouvez utiliser le partage VPC pour partager des sous-réseaux avec d'autres AWS comptes au sein des mêmes Organisations. AWS Vous pouvez créer des clusters Amazon EKS dans des sous-réseaux partagés, en tenant compte des points suivants :

- Le propriétaire du sous-réseau VPC doit partager un sous-réseau avec un compte participant avant que ce compte puisse y créer un cluster Amazon EKS.
- Vous ne pouvez pas lancer de ressources à l'aide du groupe de sécurité par défaut pour le VPC, car il appartient au propriétaire. De plus, les participants ne peuvent pas lancer de ressources avec des groupes de sécurité détenus par d'autres participants ou par le propriétaire.
- Dans un sous-réseau partagé, le participant et le propriétaire contrôlent séparément les groupes de sécurité au sein de leur compte respectif. Le propriétaire du sous-réseau peut voir ces groupes de sécurité créés par les participants, mais ne peut pas exécuter d'actions sur ceux-ci. Si le propriétaire du sous-réseau souhaite supprimer ou modifier ces groupes de sécurité, le participant qui a créé le groupe de sécurité doit effectuer l'action.
- Si un cluster est créé par un participant, il faut tenir compte des éléments suivants :
  - Le rôle IAM de cluster et les rôles IAM de nœud doivent être créés dans ce compte. Pour plus d'informations, consultez the section called "Rôle IAM du cluster" et the section called "Rôle IAM de nœud".
  - Tous les nœuds doivent être créés par le même participant, y compris les groupes de nœuds gérés.
- Le propriétaire du VPC partagé ne peut pas afficher, mettre à jour ou supprimer un cluster créé par un participant dans le sous-réseau partagé. Cela s'ajoute aux ressources VPC auxquelles chaque compte a un accès différent. Pour plus d'informations, consultez Responsabilités et autorisations pour les propriétaires et les participants dans le Guide de l'utilisateur Amazon VPC.

- Si vous utilisez la fonctionnalité réseau personnalisé du module complément CNI Amazon VPC pour Kubernetes, vous devez utiliser les mappages d'ID de zone de disponibilité répertoriés dans le compte propriétaire pour créer chaque `ENIConfig`. Pour de plus amples informations, veuillez consulter the section called "Mise en réseau personnalisée".

Pour plus d'informations sur le partage de sous-réseaux VPC, consultez Partager votre VPC avec d'autres comptes dans le Guide de l'utilisateur Amazon VPC.

Création d'un VPC Amazon pour votre cluster Amazon EKS

Vous pouvez utiliser Amazon Virtual Private Cloud (Amazon VPC) pour lancer AWS des ressources dans un réseau virtuel que vous avez défini. Ce réseau virtuel ressemble beaucoup à un réseau traditionnel que vous pourriez exécuter dans votre propre centre de données. Toutefois, il présente l'avantage d'utiliser l'infrastructure évolutive d'Amazon Web Services. Il est recommandé de bien connaître le service Amazon VPC avant de déployer des clusters Amazon EKS de production. Pour de plus amples informations, consultez le Guide de l'utilisateur Amazon VPC.

Un cluster Amazon EKS, des nœuds et des ressources Kubernetes sont déployés sur un VPC. Si vous souhaitez utiliser un VPC existant avec Amazon EKS, ce VPC doit répondre aux exigences décrites dans the section called "Exigences requises pour le VPC et les sous-réseaux". Cette rubrique explique comment créer un VPC répondant aux exigences d'Amazon EKS à l'aide d'un modèle fourni par AWS CloudFormation Amazon EKS. Une fois le modèle déployé, vous pouvez consulter les ressources qu'il crée afin de connaître précisément les ressources provisionnées et leur configuration. Si vous utilisez des nœuds hybrides, votre VPC doit contenir des routes dans sa table de routage pour votre réseau sur site. Pour plus d'informations sur les exigences réseau pour les nœuds hybrides, consultez the section called "Préparez votre réseau".

Conditions préalables

Pour créer un VPC pour Amazon EKS, vous devez disposer des autorisations IAM nécessaires pour créer des ressources Amazon VPC. Ces ressources sont les sous-réseaux VPCs, les groupes de sécurité, les tables de routage et les routes, ainsi que les passerelles Internet et NAT. Pour de plus d'informations, consultez Création d'un VPC avec un exemple de politique de sous-réseau public dans le Guide de l'utilisateur Amazon VPC et la liste complète des Actions dans la Référence de l'autorisation de service.

Vous pouvez créer un VPC avec des sous-réseaux à la fois publics et privés, uniquement avec des sous-réseaux publics ou uniquement avec des sous-réseaux privés.

Sous-réseaux publics et privés

Ce VPC comporte deux sous-réseaux publics et deux privés. Un sous-réseau public possède dans sa table de routage une route vers une passerelle Internet. En revanche, la table de routage d'un sous-réseau privé ne contient pas de route vers une passerelle Internet. Un sous-réseau public et un sous-réseau privé sont déployés dans la même zone de disponibilité. Les autres sous-réseaux publics et privés sont déployés dans une deuxième zone de disponibilité de la même AWS région. Nous recommandons cette option pour la plupart des déploiements.

Avec cette option, vous pouvez déployer vos nœuds sur des sous-réseaux privés. Cette option permet à Kubernetes de déployer des équilibreurs de charge dans les sous-réseaux publics, qui peuvent répartir le trafic vers les pods exécutés sur des nœuds situés dans les sous-réseaux privés. Les adresses `IPv4` publiques sont automatiquement attribuées aux nœuds déployés dans des sous-réseaux publics, mais les nœuds dans des sous-réseaux privés ne reçoivent pas d'adresses `IPv4` publiques.

Vous pouvez également attribuer des adresses `IPv6` aux nœuds des sous-réseaux publics et privés. Les nœuds des sous-réseaux privés peuvent communiquer avec le cluster et d'autres AWS services. Les pods peuvent accéder à Internet par le biais d'une passerelle NAT à l'aide d'adresses `IPv4` ou d'une passerelle Internet sortante uniquement utilisant des adresses `IPv6` qui est déployée dans chaque zone de disponibilité. Un groupe de sécurité est déployé et comporte des règles qui refusent tout trafic entrant provenant de sources autres que le cluster ou les nœuds mais autorise tout le trafic sortant. Les sous-réseaux sont balisés de sorte que Kubernetes puisse y déployer des équilibreurs de charge.

a. Ouvrez la AWS CloudFormation console.

b. Dans la barre de navigation, sélectionnez une AWS région compatible avec Amazon EKS.

c. Sélectionnez Créer une pile, Avec de nouvelles ressources (standard).

d. Sous Prerequisite - Prepare template (Prérequis - Préparer le modèle), assurez-vous que Template is ready (Le modèle est prêt) est sélectionné, puis sous Spécifier un modèle, sélectionnez Amazon S3 URL (URL Amazon S3).

e. Vous pouvez créer un VPC qui prend en charge uniquement `IPv4`, ou un VPC qui prend en charge `IPv4` et `IPv6`. Collez l'un des éléments suivants URLs dans la zone de texte sous l'URL Amazon S3 et choisissez Next :

   - `IPv4`

```
https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/amazon-eks-vpc-private-subnets.yaml
```

- IPv4 et IPv6

```
https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/amazon-eks-ipv6-vpc-public-private-subnets.yaml
```

a. Dans la page Spécifier les détails de la pile, entrez les paramètres, puis choisissez Suivant.

- Nom de pile : Choisissez un nom de pile pour votre AWS CloudFormation pile. Par exemple, vous pouvez utiliser le nom du modèle que vous avez utilisé à l'étape précédente. Un nom ne peut contenir que des caractères alphanumériques (sensibles à la casse) et des traits d'union. Il doit commencer par un caractère alphanumérique et ne peut pas dépasser 100 caractères. Le nom doit être unique au sein de la AWS région et du AWS compte dans lesquels vous créez le cluster.
- VpcBlock: Choisissez une plage d'IPv4adresses CIDR pour votre VPC. Chaque nœud, pod et équilibreur de charge que vous déployez se voit attribuer une adresse IPv4 à partir de ce bloc. Les valeurs IPv4 par défaut fournissent suffisamment d'adresses IP pour la plupart des implémentations, mais si ce n'est pas le cas, vous pouvez les modifier. Pour plus d'informations, consultez Dimensionnement des VPC et des sous-réseaux dans le Guide de l'utilisateur Amazon VPC. Vous pouvez également ajouter d'autres blocs CIDR au VPC après sa création. Si vous créez un VPC IPv6, des plages CIDR IPv6 sont automatiquement attribuées à partir de l'espace d'adressage Global Unicast d'Amazon.
- PublicSubnet01Block : Spécifiez un bloc IPv4 CIDR pour le sous-réseau public 1. La valeur par défaut fournit suffisamment d'adresses IP pour la plupart des implémentations, mais si ce n'est pas le cas, vous pouvez la modifier. Si vous créez un VPC IPv6, ce bloc est spécifié pour vous dans le modèle.
- PublicSubnet02Block : Spécifiez un bloc IPv4 CIDR pour le sous-réseau public 2. La valeur par défaut fournit suffisamment d'adresses IP pour la plupart des implémentations, mais si ce n'est pas le cas, vous pouvez la modifier. Si vous créez un VPC IPv6, ce bloc est spécifié pour vous dans le modèle.
- PrivateSubnet01Block : Spécifiez un bloc IPv4 CIDR pour le sous-réseau privé 1. La valeur par défaut fournit suffisamment d'adresses IP pour la plupart des implémentations, mais si ce n'est pas le cas, vous pouvez la modifier. Si vous créez un VPC IPv6, ce bloc est spécifié pour vous dans le modèle.

- PrivateSubnet02Block : Spécifiez un bloc `IPv4` CIDR pour le sous-réseau privé 2. La valeur par défaut fournit suffisamment d'adresses IP pour la plupart des implémentations, mais si ce n'est pas le cas, vous pouvez la modifier. Si vous créez un VPC `IPv6`, ce bloc est spécifié pour vous dans le modèle.

b. (Facultatif) Sur la page Configure stack options (Configurer les options de pile), étiquetez vos ressources de pile, puis choisissez Next (Suivant).

c. Sur la page Review (Vérification), choisissez Create stack (Créer une pile).

d. Lorsque votre pile est créée, sélectionnez-la dans la console et choisissez Outputs (Sorties).

e. Enregistrez le VpcIdpour le VPC qui a été créé. Vous en avez besoin lors de la création de votre cluster et de vos nœuds.

f. Enregistrez SubnetIdsles sous-réseaux créés et indiquez si vous les avez créés en tant que sous-réseaux publics ou privés. Vous en avez besoin d'au moins deux lors de la création de votre cluster et de vos nœuds.

g. Si vous avez créé un VPC `IPv4`, ignorez cette étape. Si vous avez créé un VPC `IPv6`, vous devez activer l'option d'attribution automatique d'adresses `IPv6` pour les sous-réseaux publics qui ont été créés par le modèle. Ce paramètre est déjà activé pour les sous-réseaux privés. Pour activer le paramètre, effectuez les étapes suivantes :

   i. Ouvrez la console Amazon VPC à l'adresse. https://console.aws.amazon.com/vpc/

   ii. Dans le volet de navigation de gauche, sélectionnez Subnets (Sous-réseaux).

   iii. Sélectionnez l'un de vos sous-réseaux publics (***stack-name***/SubnetPublic01 ou ***stack-name***/SubnetPublic02 contient le mot public) et choisissez Actions, Modifier les paramètres du sous-réseau.

   iv. Cochez la case Activer l'attribution automatique d' IPv6 adresses, puis sélectionnez Enregistrer.

   v. Effectuez à nouveau les étapes précédentes pour votre autre sous-réseau public.

Sous-réseaux publics uniquement

Ce VPC possède trois sous-réseaux publics déployés dans différentes zones de disponibilité d'une région. AWS Tous les nœuds se voient attribuer automatiquement des adresses `IPv4` publiques et peuvent envoyer et recevoir du trafic Internet via une passerelle Internet. Un groupe de sécurité est déployé qui refuse tout le trafic entrant et autorise tout le trafic sortant. Les sous-réseaux sont balisés de sorte que Kubernetes puisse y déployer des équilibreurs de charge.

a. Ouvrez la AWS CloudFormation console.

b. Dans la barre de navigation, sélectionnez une AWS région compatible avec Amazon EKS.

c. Choisissez Create stack (Créer une pile), Avec de nouvelles ressources (standard).

d. Sous Prepare template (Préparer le modèle), assurez-vous que Prepare template (Modèle est prêt) est sélectionné, puis sous Template source (Source du modèle), sélectionnez Amazon S3 URL (URL Amazon S3).

e. Collez l'URL suivante dans la zone de texte sous URL Amazon S3 et sélectionnez Next (Suivant) :

```
https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/amazon-eks-vpc-sample.yaml
```

a. Dans la page Spécifier les détails, entrez les paramètres, puis choisissez Suivant.

- Nom de pile : Choisissez un nom de pile pour votre AWS CloudFormation pile. Par exemple, vous pouvez l'appeler *amazon-eks-vpc-sample*. Un nom ne peut contenir que des caractères alphanumériques (sensibles à la casse) et des traits d'union. Il doit commencer par un caractère alphanumérique et ne peut pas dépasser 100 caractères. Le nom doit être unique au sein de la AWS région et du AWS compte dans lesquels vous créez le cluster.
- VpcBlock: Choisissez un bloc CIDR pour votre VPC. Chaque nœud, pod et équilibreur de charge que vous déployez se voit attribuer une adresse `IPv4` à partir de ce bloc. Les valeurs `IPv4` par défaut fournissent suffisamment d'adresses IP pour la plupart des implémentations, mais si ce n'est pas le cas, vous pouvez les modifier. Pour plus d'informations, consultez Dimensionnement des VPC et des sous-réseaux dans le Guide de l'utilisateur Amazon VPC. Vous pouvez également ajouter d'autres blocs CIDR au VPC après sa création.
- Subnet01Block : spécifiez un bloc d'adresse CIDR pour le sous-réseau 1. La valeur par défaut fournit suffisamment d'adresses IP pour la plupart des implémentations, mais si ce n'est pas le cas, vous pouvez la modifier.
- Subnet02Block : spécifiez un bloc d'adresse CIDR pour le sous-réseau 2. La valeur par défaut fournit suffisamment d'adresses IP pour la plupart des implémentations, mais si ce n'est pas le cas, vous pouvez la modifier.
- Subnet03Block : spécifiez un bloc d'adresse CIDR pour le sous-réseau 3. La valeur par défaut fournit suffisamment d'adresses IP pour la plupart des implémentations, mais si ce n'est pas le cas, vous pouvez la modifier.

b. (Facultatif) Sur la page Options (Options), étiquetez les ressources de votre pile. Choisissez Suivant.

c. Sur la page Review (Vérification), choisissez Create (Créer).

d. Lorsque votre pile est créée, sélectionnez-la dans la console et choisissez Outputs (Sorties).

e. Enregistrez le VpcIdpour le VPC qui a été créé. Vous en avez besoin lors de la création de votre cluster et de vos nœuds.

f. Enregistrez le SubnetIdspour les sous-réseaux qui ont été créés. Vous en avez besoin d'au moins deux lors de la création de votre cluster et de vos nœuds.

g. (Facultatif) Tout cluster que vous déployez dans ce VPC peut attribuer des adresses `IPv4` privées à vos pods et services. Si vous souhaitez que les clusters déployés dans ce VPC attribuent des adresses `IPv6` privées à vos pods et services, vous devrez modifier le VPC, les sous-réseaux, les tables de routage et les groupes de sécurité. Pour plus d'informations, consultez la section Migrer les VPCs fichiers existants de IPv4 vers IPv6 dans le guide de l'utilisateur Amazon VPC. Amazon EKS requiert que l'option `Auto-assign` d'adresses `IPv6` soit activée pour vos sous-réseaux. Cette option est désactivée par défaut.

Sous-réseaux privés uniquement

Ce VPC possède trois sous-réseaux privés déployés dans différentes zones de disponibilité de la région. AWS Les ressources déployées dans ces sous-réseaux ne peuvent pas accéder à Internet et ne sont pas accessibles depuis Internet. Le modèle crée des points de terminaison VPC à l'aide AWS PrivateLink de plusieurs AWS services auxquels les nœuds ont généralement besoin d'accéder. Si vos nœuds ont besoin d'un accès Internet sortant, vous pouvez ajouter une passerelle NAT publique dans la zone de disponibilité de chaque sous-réseau après la création du VPC. Un groupe de sécurité est créé qui refuse tout trafic entrant, à l'exception des ressources déployées dans les sous-réseaux. Un groupe de sécurité autorise également tout le trafic sortant. Les sous-réseaux sont balisés de sorte que Kubernetes puisse y déployer des équilibreurs de charge. Si vous créez un VPC avec cette configuration, consultez the section called “Clusters privés” pour des exigences et des considérations supplémentaires.

a. Ouvrez la AWS CloudFormation console.

b. Dans la barre de navigation, sélectionnez une AWS région compatible avec Amazon EKS.

c. Choisissez Create stack (Créer une pile), Avec de nouvelles ressources (standard).

d. Sous Prepare template (Préparer le modèle), assurez-vous que Prepare template (Modèle est prêt) est sélectionné, puis sous Template source (Source du modèle), sélectionnez Amazon S3 URL (URL Amazon S3).

e. Collez l'URL suivante dans la zone de texte sous URL Amazon S3 et sélectionnez Next (Suivant) :

```
https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/amazon-eks-fully-private-vpc.yaml
```

a. Sur la page Spécifier les détails, renseignez les paramètres, puis choisissez Suivant.
   - Nom de pile : Choisissez un nom de pile pour votre AWS CloudFormation pile. Par exemple, vous pouvez l'appeler *`amazon-eks-fully-private-vpc`*. Un nom ne peut contenir que des caractères alphanumériques (sensibles à la casse) et des traits d'union. Il doit commencer par un caractère alphanumérique et ne peut pas dépasser 100 caractères. Le nom doit être unique au sein de la AWS région et du AWS compte dans lesquels vous créez le cluster.
   - VpcBlock: Choisissez un bloc CIDR pour votre VPC. Chaque nœud, pod et équilibreur de charge que vous déployez se voit attribuer une adresse `IPv4` à partir de ce bloc. Les valeurs `IPv4` par défaut fournissent suffisamment d'adresses IP pour la plupart des implémentations, mais si ce n'est pas le cas, vous pouvez les modifier. Pour plus d'informations, consultez Dimensionnement des VPC et des sous-réseaux dans le Guide de l'utilisateur Amazon VPC. Vous pouvez également ajouter d'autres blocs CIDR au VPC après sa création.
   - PrivateSubnet01Block : Spécifiez un bloc CIDR pour le sous-réseau 1. La valeur par défaut fournit suffisamment d'adresses IP pour la plupart des implémentations, mais si ce n'est pas le cas, vous pouvez la modifier.
   - PrivateSubnet02Block : Spécifiez un bloc CIDR pour le sous-réseau 2. La valeur par défaut fournit suffisamment d'adresses IP pour la plupart des implémentations, mais si ce n'est pas le cas, vous pouvez la modifier.
   - PrivateSubnet03Block : Spécifiez un bloc CIDR pour le sous-réseau 3. La valeur par défaut fournit suffisamment d'adresses IP pour la plupart des implémentations, mais si ce n'est pas le cas, vous pouvez la modifier.
b. (Facultatif) Sur la page Options (Options), étiquetez les ressources de votre pile. Choisissez Suivant.
c. Sur la page Review (Vérification), choisissez Create (Créer).
d. Lorsque votre pile est créée, sélectionnez-la dans la console et choisissez Outputs (Sorties).
e. Enregistrez le VpcIdpour le VPC qui a été créé. Vous en avez besoin lors de la création de votre cluster et de vos nœuds.
f. Enregistrez le SubnetIdspour les sous-réseaux qui ont été créés. Vous en avez besoin d'au moins deux lors de la création de votre cluster et de vos nœuds.
g. (Facultatif) Tout cluster que vous déployez dans ce VPC peut attribuer des adresses `IPv4` privées à vos pods et services. Si vous souhaitez déployer des clusters sur ce VPC pour attribuer des

adresses IPv6 privées à vos pods et services, vous devez mettre à jour votre VPC, votre sous-réseau, vos tables de routage et vos groupes de sécurité. Pour plus d'informations, consultez la section Migrer les VPCs fichiers existants de IPv4 vers IPv6 dans le guide de l'utilisateur Amazon VPC. Amazon EKS exige que l'option d'adresses Auto-assign IPv6 soit activée pour vos sous-réseaux (elle est désactivée par défaut).

Voir les exigences relatives aux groupes de sécurité Amazon EKS pour les clusters

Cette rubrique décrit les exigences de groupe de sécurité d'un cluster Amazon EKS.

Groupe de sécurité par défaut du cluster

Lorsque vous créez un cluster, Amazon EKS crée un groupe de sécurité nommé `eks-cluster-sg-`*`my-cluster`*`-`*`uniqueID`* . Ce groupe de sécurité dispose des règles par défaut suivantes :

Type de règle	Protocole	Ports	Source	Destination
Entrant	Tous	Tous	Auto-utilisateur	
Sortant	Tous	Tous		0.0.0.0/0(IPv4) ou ::/0 (IPv6)
Sortant	Tous	Tous		Self (pour le trafic EFA)

Le groupe de sécurité par défaut comprend une règle sortante qui autorise le trafic Elastic Fabric Adapter (EFA) vers la même destination que le groupe de sécurité. Cela permet le trafic EFA au sein du cluster, ce qui est avantageux pour les charges de travail IA/ML et le calcul haute performance (HPC). Pour plus d'informations, consultez Elastic Fabric Adapter pour les charges de travail IA/ML et HPC sur Amazon EC2 dans le Guide de l'utilisateur Amazon Elastic Compute Cloud.

> **Important**
>
> Si votre cluster n'a pas besoin de la règle sortante, vous pouvez la supprimer. Si vous la supprimez, vous devez toujours avoir les règles minimales énumérées dans la section

Restriction du trafic du cluster. Si vous supprimez la règle entrante, Amazon EKS la recrée à chaque fois que le cluster est mis à jour.

Amazon EKS ajoute les balises suivantes au groupe de sécurité. Si vous supprimez les balises, Amazon EKS les ajoute à nouveau au groupe de sécurité à chaque fois que le cluster est mis à jour.

Clé	Valeur
kubernetes.io/cluster/ *my-cluster*	owned
aws:eks:cluster-name	*mon-cluster*
Name	eks-cluster-sg- *my-cluste r -uniqueid*

Amazon EKS associe automatiquement ce groupe de sécurité aux ressources suivantes qu'il crée également :

- 2 à 4 interfaces réseau élastiques (désignées pour le reste de ce document sous le nom d'interface réseau) créés lors de la création de votre cluster.
- Interfaces réseau des nœuds dans n'importe quel groupe de nœuds géré que vous créez.

Les règles par défaut permettent à tout le trafic de circuler librement entre votre cluster et vos nœuds, et autorise tout le trafic sortant vers n'importe quelle destination. Lorsque vous créez un cluster, spécifiez (éventuellement) vos propres groupes de sécurité. Si c'est le cas, Amazon EKS associe également les groupes de sécurité que vous spécifiez aux interfaces réseau qu'il crée pour votre cluster. Cependant, cela ne les associe à aucun groupe de nœuds que vous créez.

Vous pouvez déterminer l'ID de votre groupe de sécurité du cluster dans la AWS Management Console, sous la section Réseaux du cluster. Vous pouvez également le faire en exécutant la commande AWS CLI suivante.

```
aws eks describe-cluster --name my-cluster --query
 cluster.resourcesVpcConfig.clusterSecurityGroupId
```

Restriction du trafic de cluster

Si vous devez limiter les ports ouverts entre le cluster et les nœuds, vous pouvez supprimer la règle sortante par défaut et ajouter les règles minimales suivantes qui sont requises pour le cluster. Si vous supprimez la règle entrante par défaut, Amazon EKS la recrée à chaque fois que le cluster est mis à jour.

Type de règle	Protocole	Port	Destination
Sortant	TCP	443	Groupe de sécurité du cluster
Sortant	TCP	10250	Groupe de sécurité du cluster
Sortant (DNS)	TCP et UDP	53	Groupe de sécurité du cluster

Vous devez également ajouter des règles pour le trafic suivant :

- Tout protocole et port que vos nœuds peuvent utiliser pour la communication entre les nœuds.
- Accès Internet sortant afin que les nœuds puissent accéder aux API Amazon EKS pour l'introspection des clusters et l'enregistrement des nœuds au moment du lancement. Si vos nœuds n'ont pas accès à Internet, consultez Déployer des clusters privés avec un accès Internet limité pour obtenir des informations supplémentaires.
- Accès au nœud pour extraire des images de conteneur d'Amazon ECR ou d'autres API de registres de conteneurs dont ils ont besoin pour extraire des images, telles que DockerHub. Pour plus d'informations, consultez la section Plages d'adresses IP AWS dans la référence générale AWS.
- Accès au nœud Amazon S3.
- Des règles distinctes sont requises pour les adresses `IPv4` et `IPv6`.
- Si vous utilisez des nœuds hybrides, vous devez ajouter un groupe de sécurité supplémentaire à votre cluster pour permettre la communication avec vos nœuds et pods sur site. Pour de plus amples informations, consultez the section called "Préparez votre réseau".

Si vous envisagez de limiter les règles, nous vous recommandons de tester minutieusement tous vos pods avant d'appliquer vos règles modifiées à un cluster de production.

Si vous avez initialement déployé un cluster avec Kubernetes `1.14` et une version plateforme `eks.3` ou antérieure, prenez en considération les éléments suivants :

- Vous pouvez également disposer de groupes de sécurité de nœud et de plan de contrôle. Lorsque ces groupes ont été créés, ils incluaient les règles restreintes répertoriées dans le tableau précédent. Ces groupes de sécurité ne sont plus nécessaires et peuvent être supprimés. Toutefois, vous devez vous assurer que le groupe de sécurité de votre cluster contient les règles que ces groupes contiennent.
- Si vous avez déployé le cluster directement à l'aide de l'API ou si vous avez utilisé un outil tel que l'interface AWS CLI ou AWS CloudFormation pour créer le cluster et que vous n'avez pas spécifié de groupe de sécurité lors de la création du cluster, le groupe de sécurité par défaut pour le VPC a été appliqué aux interfaces réseau du cluster créées par Amazon EKS.

Groupes de sécurité partagés

Amazon EKS prend en charge les groupes de sécurité partagés.

- Associations de groupes de sécurité VPC associent des groupes de sécurité à plusieurs VPC dans le même compte et la même région.
  - Découvrez comment associer des groupes de sécurité à plusieurs VPC dans le Guide de l'utilisateur Amazon VPC.
- Groupes de sécurité partagés vous permettent de partager des groupes de sécurité avec d'autres comptes AWS. Les comptes doivent appartenir à la même organisation AWS.
  - Découvrez comment partager des groupes de sécurité avec des organisations dans le Guide de l'utilisateur Amazon VPC.
- Les groupes de sécurité sont toujours limités à une seule région AWS.

Considérations relatives à Amazon EKS

- EKS a les mêmes exigences que les groupes de sécurité standard en matière de groupes de sécurité partagés ou multi-VPC.

Gérer les modules complémentaires réseau pour les clusters Amazon EKS

Plusieurs modules complémentaires de mise en réseau sont disponibles pour votre cluster Amazon EKS.

Modules complémentaires intégrés

> **Note**
>
> Lorsque vous créez un cluster EKS :
>
> - Utilisation de la AWS console : les modules complémentaires intégrés (tels que CoreDNS, kube-proxy, etc.) sont automatiquement installés en tant que modules complémentaires Amazon EKS. Ils peuvent être facilement configurés et mis à jour via la AWS console, la CLI ou SDKs.
> - Utilisation d'autres méthodes (CLI SDKs, etc.) : les mêmes modules complémentaires intégrés sont installés sous forme de versions autogérées qui s'exécutent comme des déploiements Kubernetes réguliers. Ils nécessitent une configuration et des mises à jour manuelles car ils ne peuvent pas être gérés par le biais AWS d'outils.
>
> Nous recommandons d'utiliser des modules complémentaires Amazon EKS plutôt que des versions autogérées afin de simplifier la gestion des modules complémentaires et de permettre une configuration et des mises à jour centralisées par le biais de AWS services.

Plug-in CNI Amazon VPC pour Kubernetes

Ce module complémentaire CNI crée des interfaces réseau élastiques et les attache à vos EC2 nœuds Amazon. Le module complémentaire attribue également une adresse privée `IPv4` ou `IPv6` de votre VPC à chaque pod et service. Par défaut, ce module complémentaire est installé sur votre cluster. Pour de plus amples informations, veuillez consulter the section called “CNI Amazon VPC”. Si vous utilisez des nœuds hybrides, le VPC CNI est toujours installé par défaut, mais son exécution sur vos nœuds hybrides est empêchée par une règle d'anti-affinité. Pour plus d'informations sur vos options CNI pour les nœuds hybrides, consultez the section called “Configurer CNI”.

CoreDNS

CoreDNS est un serveur DNS flexible et extensible qui peut servir de DNS de cluster Kubernetes. CoreDNS fournit la résolution de noms pour tous les pods du cluster. Par défaut, ce module complémentaire est installé sur votre cluster. Pour de plus amples informations, veuillez consulter the section called "CoreDNS".

`kube-proxy`

Ce module complémentaire gère les règles du réseau sur vos EC2 nœuds Amazon et permet la communication réseau avec vos Pods. Par défaut, ce module complémentaire est installé sur votre cluster. Pour de plus amples informations, veuillez consulter the section called "kube-proxy".

Modules complémentaires AWS réseau en option

Contrôleur d'équilibreur de charge AWS

Lorsque vous déployez des objets de service Kubernetes de type Kubernetes`loadbalancer`, le contrôleur crée des équilibreurs de charge AWS réseau. Lorsque vous créez des objets d'entrée Kubernetes, le contrôleur crée AWS des équilibreurs de charge d'application. Nous recommandons d'utiliser ce contrôleur pour provisionner les Network Load Balancers, plutôt que d'utiliser l'ancien contrôleur Cloud Provider intégré à Kubernetes. Pour plus d'informations, consultez la documentation du contrôleur AWS Load Balancer Controller.

AWS Gateway API Controller

Ce contrôleur vous permet de connecter des services sur plusieurs clusters Kubernetes à l'aide de l'API Kubernetes Gateway. Le contrôleur connecte les services Kubernetes exécutés sur des EC2 instances Amazon, des conteneurs et des fonctions sans serveur à l'aide du service Amazon VPC Lattice. Pour plus d'informations, veuillez consulter la documentation Contrôleur d'API Gateway AWS.

Pour plus d'informations sur les modules complémentaires, voir the section called "Modules complémentaires Amazon EKS".

Attribuer IPs à des pods avec l'Amazon VPC CNI

> **Tip**
>
> Inscrivez-vous aux prochains ateliers Amazon EKS.

> **Tip**
>
> Avec le mode automatique Amazon EKS, il n'est pas nécessaire d'installer ou de mettre à niveau de modules complémentaires de mise en réseau. Le mode automatique inclut des fonctionnalités de mise en réseau des pods et d'équilibrage de charge.
> Pour de plus amples informations, veuillez consulter Mode automatique EKS.

Le plug-in Amazon VPC CNI pour Kubernetes est déployé sur chaque nœud Amazon EC2 de votre cluster Amazon EKS. Le module complémentaire crée des interfaces réseau élastiques et les attache à vos nœuds Amazon EC2. Le module complémentaire attribue également une adresse `IPv4` ou `IPv6` privée ou une adresse de votre VPC à chaque pod.

Une version du module complémentaire est déployée avec chaque nœud Fargate de votre cluster, mais vous ne devez pas la mettre à jour sur les nœuds Fargate. D'autres plug-ins CNI compatibles peuvent être utilisés sur les clusters Amazon EKS, mais il s'agit du seul plug-in CNI pris en charge par Amazon EKS pour les nœuds exécutés sur AWS une infrastructure. Pour plus d'informations sur les autres plugins CNI compatibles, voir the section called "Plug-ins CNI alternatifs". Le CNI VPC n'est pas compatible avec les nœuds hybrides. Pour plus d'informations sur vos options CNI pour les nœuds hybrides, consultez the section called "Configurer CNI".

Le tableau suivant répertorie la dernière version disponible du type de module complémentaire Amazon EKS pour chaque version de Kubernetes.

Versions CNI d'Amazon VPC

Version de Kubernetes	Type de version VPC CNI d'Amazon EKS
1,35	v1.21.1-eksbuild.3
1,34	v1.21.1-eksbuild.3

Version de Kubernetes	Type de version VPC CNI d'Amazon EKS
1,33	v1.21.1-eksbuild.3
1,32	v1.21.1-eksbuild.3
1,31	v1.21.1-eksbuild.3
1,30	v1.21.1-eksbuild.3
1,29	v1.21.1-eksbuild.3

Important

Si vous gérez vous-même ce module complémentaire, les versions répertoriées dans le tableau peuvent ne pas être les mêmes que les versions autogérées disponibles. Pour plus d'informations sur la mise à jour du type autogéré de ce module complémentaire, consultez la rubrique the section called “Mise à jour (autogéré)”.

Important

Pour passer à la version VPC CNI v1.12.0 ou ultérieure, vous devez d'abord passer à la version VPC CNI v1.7.0. Nous vous recommandons de mettre à jour une version mineure à la fois.

Considérations

Tenez compte des points suivants pour utiliser cette fonctionnalité.

- Les versions sont spécifiées comme `major-version.minor-version.patch-version-eksbuild.build-number`.
- Vérifiez la compatibilité des versions pour chaque fonctionnalité. Certaines fonctionnalités de chaque version du plug-in CNI Amazon VPC pour Kubernetes nécessitent certaines versions de Kubernetes. Lorsque vous utilisez différentes fonctionnalités Amazon EKS, si une version spécifique du module complémentaire est requise, celle-ci est indiquée dans la documentation

des fonctionnalités. Nous vous recommandons d'utiliser la dernière version, sauf si vous avez une raison précise d'utiliser une version antérieure.

Créer le CNI Amazon VPC (module complémentaire Amazon EKS)

Suivez les étapes suivantes pour créer le module complémentaire Plug-in CNI Amazon VPC pour Kubernetes pour Amazon EKS.

Avant de commencer, consultez les points à prendre en compte. Pour de plus amples informations, veuillez consulter the section called “Considérations”.

Conditions préalables

Voici les conditions préalables pour le module complémentaire Plug-in CNI Amazon VPC pour Kubernetes pour Amazon EKS.

- Un cluster Amazon EKS existant. Pour en déployer un, consultez Démarrer.
- Un fournisseur IAM (AWS Identity and Access Management) (IAM) OpenID Connect (OIDC) existant pour votre cluster. Pour déterminer si vous en avez déjà un, ou pour en créer un, consultez the section called “Fournisseur IAM OIDC”.
- Un rôle IAM auquel est attachée la politique IAM Amazoneks_CNI_Policy (si votre cluster utilise la famille) ou une politique (si votre cluster utilise `IPv4` la famille). IPv6 `IPv6` Pour plus d'informations sur le rôle CNI VPC, consultez the section called “Configuration pour IRSA”. Pour plus d'informations sur IPv6 cette politique, consultezthe section called “Créer une politique IAM pour les clusters qui utilisent la famille `IPv6`”.

> **Important**
>
> Les versions `v1.16.0` à `v1.16.1` du plug-in CNI Amazon VPC pour Kubernetes implémentent la spécification CNI version `v1.0.0`. Pour plus d'informations sur `v1.0.0` la spécification CNI, voir Spécification de l'interface réseau de conteneurs (CNI) sur. GitHub

Procédure

Une fois les conditions préalbasales remplies, veuillez suivre les étapes suivantes pour créer le module complémentaire.

1. Déterminez la version du module complémentaire actuellement installée sur votre cluster.

```
kubectl describe daemonset aws-node --namespace kube-system | grep amazon-k8s-cni: |
 cut -d : -f 3
```

   L'exemple qui suit illustre un résultat.

```
v1.16.4-eksbuild.2
```

2. Déterminez le type de module complémentaire installé sur votre cluster. Selon l'outil avec lequel vous avez créé votre cluster, le type de module complémentaire Amazon EKS peut ne pas être actuellement installé sur votre cluster. Remplacez *my-cluster* par le nom de votre cluster.

```
aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni --query
 addon.addonVersion --output text
```

   Si un numéro de version est renvoyé, le type de module complémentaire Amazon EKS est installé sur votre cluster, et vous n'avez donc pas besoin de suivre les étapes restantes de cette procédure. Si une erreur est renvoyée, cela signifie que le type de module complémentaire Amazon EKS n'est pas installé sur votre cluster. Suivez les étapes restantes de cette procédure pour l'installer.

3. Enregistrez la configuration du module complémentaire actuellement installé.

```
kubectl get daemonset aws-node -n kube-system -o yaml > aws-k8s-cni-old.yaml
```

4. Créez le module complémentaire à l'aide de la AWS CLI. Si vous souhaitez utiliser le AWS Management Console ou `eksctl` pour créer le module complémentaire, consultez the section called "Créer un module complémentaire" et spécifiez `vpc-cni` le nom du module complémentaire. Copiez la commande qui suit sur votre appareil. Si nécessaire, apportez les modifications suivantes à la commande, puis exécutez la commande modifiée.
   - Remplacez *my-cluster* par le nom de votre cluster.
   - *v1.20.3-eksbuild.1*Remplacez-le par la dernière version répertoriée dans le tableau des dernières versions pour la version de votre cluster. Pour consulter le tableau des dernières versions, consultez the section called "Versions CNI d'Amazon VPC".
   - *111122223333*Remplacez-le par votre identifiant de compte et *AmazonEKSVPCCNIRole* par le nom d'un rôle IAM existant que vous avez créé. Pour spécifier un rôle, vous devez disposer d'un

fournisseur IAM OpenID Connect (OIDC) pour votre cluster. Pour déterminer si vous en avez un pour votre cluster ou pour en créer un, consultez the section called "Fournisseur IAM OIDC".

```
aws eks create-addon --cluster-name my-cluster --addon-name vpc-cni --addon-version
 v1.20.3-eksbuild.1 \
    --service-account-role-arn arn:aws: iam::111122223333:role/AmazonEKSVPCCNIRole
```

Si vous avez appliqué à votre module complémentaire actuel des paramètres personnalisés qui entrent en conflit avec les paramètres par défaut du module complémentaire Amazon EKS, la création peut échouer. Si la création échoue, vous recevez un message d'erreur qui peut vous aider à résoudre le problème. Vous pouvez également ajouter `--resolve-conflicts OVERWRITE` à la commande précédente. Cela permet au module complémentaire de remplacer les paramètres personnalisés existants. Une fois que vous avez créé le module complémentaire, vous pouvez le mettre à jour avec vos paramètres personnalisés.

5. Vérifiez que la dernière version du module complémentaire pour la version Kubernetes de votre cluster a bien été ajoutée à votre cluster. Remplacez *`my-cluster`* par le nom de votre cluster.

```
aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni --query
 addon.addonVersion --output text
```

La création du module complémentaire peut prendre plusieurs secondes.

L'exemple qui suit illustre un résultat.

```
v1.20.3-eksbuild.1
```

6. Si vous avez défini des paramètres personnalisés pour votre module complémentaire d'origine avant de créer le module complémentaire Amazon EKS, utilisez la configuration que vous avez enregistrée à l'étape précédente pour mettre à jour le module complémentaire EKS avec vos paramètres personnalisés. Suivez les étapes de the section called "Mise à jour (module complémentaire EKS)".
7. (Facultatif) Installez le `cni-metrics-helper` sur votre cluster. Il collecte les informations relatives à l'interface Elastic Network et aux adresses IP, les agrège au niveau du cluster et publie les statistiques sur Amazon. CloudWatch Pour plus d'informations, voir cni-metrics-helperci-dessous GitHub.

Mettre à jour le CNI Amazon VPC (module complémentaire Amazon EKS)

Mettez à jour le type Amazon EKS du module complémentaire CNI Amazon VPC pour Kubernetes. Si vous n'avez pas ajouté le type Amazon EKS du module complémentaire à votre cluster, vous pouvez l'installer en suivant les instructions fournies à l'adresse suivante : the section called "Créer". Vous pouvez également mettre à jour l'autre type d'installation CNI VPC en suivant les instructions fournies à l'adresse suivante : the section called "Mise à jour (autogéré)".

1. Déterminez la version du module complémentaire actuellement installée sur votre cluster. Remplacez *my-cluster* par le nom de votre cluster.

   ```
   aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni --query
    "addon.addonVersion" --output text
   ```

   L'exemple qui suit illustre un résultat.

   ```
   v1.20.0-eksbuild.1
   ```

   Veuillez comparer la version avec le tableau des dernières versions disponibles à l'adresse suivante : the section called "Versions CNI d'Amazon VPC". Si la version renvoyée correspond à la version Kubernetes de votre cluster dans le tableau des dernières versions, cela signifie que vous disposez déjà de la dernière version installée sur votre cluster et que vous n'avez pas besoin de suivre le reste de cette procédure. Si vous recevez une erreur au lieu d'un numéro de version dans votre sortie, cela signifie que le type de module complémentaire Amazon EKS n'est pas installé sur votre cluster. Vous devez créer le module complémentaire avant de pouvoir le mettre à jour à l'aide de cette procédure. Pour créer le type Amazon EKS du module complémentaire CNI VPC, veuillez suivre les instructions fournies à l'adresse suivante : the section called "Créer".
2. Enregistrez la configuration du module complémentaire actuellement installé.

   ```
   kubectl get daemonset aws-node -n kube-system -o yaml > aws-k8s-cni-old.yaml
   ```
3. Mettez à jour votre module complémentaire à l'aide de la AWS CLI. Si vous souhaitez utiliser AWS Management Console ou mettre `eksctl` à jour le module complémentaire, consultezthe section called "Mettre à jour un module complémentaire". Copiez la commande qui suit sur votre appareil. Si nécessaire, apportez les modifications suivantes à la commande, puis exécutez la commande modifiée.

   - Remplacez *my-cluster* par le nom de votre cluster.

- *v1.20.0-eksbuild.1*Remplacez-le par la dernière version répertoriée dans le tableau des dernières versions de votre cluster.
- *111122223333*Remplacez-le par votre identifiant de compte et *AmazonEKSVPCCNIRole* par le nom d'un rôle IAM existant que vous avez créé. Pour créer un rôle IAM pour le CNI VPC, consultez the section called "Étape 1 : créer le rôle IAM du plug-in Amazon VPC CNI pour Kubernetes". Pour spécifier un rôle, vous devez disposer d'un fournisseur IAM OpenID Connect (OIDC) pour votre cluster. Pour déterminer si vous en avez un pour votre cluster ou pour en créer un, consultez the section called "Fournisseur IAM OIDC".
- Cette option `--resolve-conflicts PRESERVE` conserve les valeurs de configuration existantes pour le module complémentaire. Si vous avez défini des valeurs personnalisées pour les paramètres des modules complémentaires et que vous n'utilisez pas cette option, Amazon EKS remplace vos valeurs par ses valeurs par défaut. Si vous utilisez cette option, nous vous recommandons de tester les modifications de champ et de valeur sur un cluster hors production avant de mettre à jour le module complémentaire sur votre cluster de production. Si vous remplacez cette valeur par `OVERWRITE`, tous les paramètres sont remplacés par les valeurs par défaut d'Amazon EKS. Si vous avez défini des valeurs personnalisées pour certains paramètres, il est possible qu'elles soient remplacées par les valeurs par défaut d'Amazon EKS. Si vous remplacez cette valeur par `none`, Amazon EKS ne modifie la valeur d'aucun paramètre, mais la mise à jour risque d'échouer. Si la mise à jour échoue, vous recevez un message d'erreur pour vous aider à résoudre le conflit.
- Si vous ne mettez à jour aucun paramètre de configuration, supprimez `--configuration-values '{`*`"env":{"AWS_VPC_K8S_CNI_EXTERNALSNAT":"true"}`*`}'` de la commande. Si vous mettez à jour un paramètre de configuration, *`"env":{"AWS_VPC_K8S_CNI_EXTERNALSNAT":"true"}`* remplacez-le par le paramètre que vous souhaitez définir. Dans cet exemple, la variable d'environnement `AWS_VPC_K8S_CNI_EXTERNALSNAT` est définie sur `true`. La valeur que vous spécifiez doit être valide pour le schéma de configuration. Si vous ne connaissez pas le schéma de configuration`aws eks describe-addon-configuration --addon-name vpc-cni --addon-version` *`v1.20.0-eksbuild.1`* , exécutez-le en le *`v1.20.0-eksbuild.1`* remplaçant par le numéro de version du module complémentaire dont vous souhaitez consulter la configuration. Le schéma est renvoyé dans la sortie. Si vous avez une configuration personnalisée existante, que vous souhaitez tout supprimer et rétablir les valeurs de tous les paramètres sur les valeurs par défaut d'Amazon EKS, supprimez-la *`"env":{"AWS_VPC_K8S_CNI_EXTERNALSNAT":"true"}`* de la commande afin d'avoir le champ vide`{}`. Pour une explication de chaque paramètre, voir Variables de configuration CNI activées GitHub.

```
aws eks update-addon --cluster-name my-cluster --addon-name vpc-cni --addon-version
 v1.20.3-eksbuild.1 \
    --service-account-role-arn arn:aws: iam::111122223333:role/AmazonEKSVPCCNIRole
 \
    --resolve-conflicts PRESERVE --configuration-values '{"env":
{"AWS_VPC_K8S_CNI_EXTERNALSNAT":"true"}}'
```

La mise à jour peut prendre plusieurs secondes.

4. Assurez-vous que la version du module complémentaire a été mise à jour. Remplacez *my-cluster* par le nom de votre cluster.

```
aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni
```

La mise à jour peut prendre plusieurs secondes.

L'exemple qui suit illustre un résultat.

```
{
    "addon": {
        "addonName": "vpc-cni",
        "clusterName": "my-cluster",
        "status": "ACTIVE",
        "addonVersion": "v1.20.3-eksbuild.1",
        "health": {
            "issues": []
        },
        "addonArn": "arn:aws: eks:region:111122223333:addon/my-cluster/vpc-
cni/74c33d2f-b4dc-8718-56e7-9fdfa65d14a9",
        "createdAt": "2023-04-12T18:25:19.319000+00:00",
        "modifiedAt": "2023-04-12T18:40:28.683000+00:00",
        "serviceAccountRoleArn": "arn:aws: iam::111122223333:role/
AmazonEKSVPCCNIRole",
        "tags": {},
        "configurationValues": "{\"env\":{\"AWS_VPC_K8S_CNI_EXTERNALSNAT\":\"true
\"}}"
    }
}
```

Résolution des problèmes

Lors de la mise à niveau du VPC CNI à partir d'une version antérieure à la version v1.13.2, vous devez remplacer tous les nœuds du cluster après la mise à jour. Les versions antérieures à la v1.13.2 utilisent le backend iptables-legacy pour insérer les règles iptables nécessaires au bon fonctionnement, telles que le NAT source (SNAT).

La version v1.13.2 était une version importante qui a introduit iptables-wrapper, qui détecte automatiquement le backend iptables approprié (iptables-legacy ou iptables-nft) pour insérer des chaînes et des règles. Ce changement s'inscrivait dans la décision prise en amont par Kubernetes de s'éloigner de l'ancien backend en raison de limites de performances.

Le remplacement des nœuds suite à une mise à niveau depuis une version antérieure à la version v1.13.2 du VPC CNI est nécessaire, car l'introduction de règles dans les backends iptables-legacy et iptables-nft peut entraîner un comportement inattendu pour le trafic provenant de sites non principaux. ENIs

Mettre à jour le CNI Amazon VPC (module complémentaire autogéré)

> **Important**
>
> Nous recommandons d'ajouter le type Amazon EKS du module complémentaire à votre cluster au lieu d'utiliser le type autogéré du module complémentaire. Si la différence entre les types ne vous est pas familière, consultez the section called “Modules complémentaires Amazon EKS”. Pour plus d'informations sur l'ajout d'un module complémentaire Amazon EKS à votre cluster, consultez the section called “Créer un module complémentaire”. Si vous ne parvenez pas à utiliser le module complémentaire Amazon EKS, nous vous encourageons à signaler les raisons pour lesquelles vous ne pouvez pas utiliser le module complémentaire Amazon EKS dans le GitHub référentiel de feuilles de route pour les conteneurs.

1. Vérifiez que le module complémentaire de type Amazon EKS n'est pas installé sur votre cluster. Remplacez *my-cluster* par le nom de votre cluster.

   ```
   aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni --query
    addon.addonVersion --output text
   ```

   Si un message d'erreur s'affiche, cela signifie que le module complémentaire de type Amazon EKS n'est pas installé sur votre cluster. Pour gérer vous-même le module complémentaire, suivez les

étapes restantes de cette procédure afin de le mettre à jour. Si un numéro de version est renvoyé, le type de module complémentaire Amazon EKS est installé sur votre cluster. Pour le mettre à jour, suivez la procédure décrite dans la rubrique the section called "Mettre à jour un module complémentaire" plutôt que cette procédure. Si vous ne connaissez pas les différences entre les types d'extensions, consultez the section called "Modules complémentaires Amazon EKS".

2. Découvrez quelle version de l'image de conteneur est actuellement installée sur votre cluster.

```
kubectl describe daemonset aws-node --namespace kube-system | grep amazon-k8s-cni: |
 cut -d : -f 3
```

L'exemple qui suit illustre un résultat.

```
v1.20.0-eksbuild.1
```

Il est possible que votre sortie n'inclue pas le numéro de build.

3. Sauvegardez vos paramètres actuels afin de pouvoir les reconfigurer une fois la mise à jour effectuée.

```
kubectl get daemonset aws-node -n kube-system -o yaml > aws-k8s-cni-old.yaml
```

Pour consulter les versions disponibles et vous familiariser avec les modifications apportées à la version vers laquelle vous souhaitez effectuer la mise à jour, consultez les versions publiées sur GitHub. Notez que nous vous recommandons de procéder à la même mise à jour`major`. `minor`. `patch`version répertoriée dans le tableau des dernières versions disponibles, même si des versions ultérieures sont disponibles sur GitHub. Pour consulter le tableau des dernières versions disponibles, consultez the section called "Versions CNI d'Amazon VPC". Les versions de build répertoriées dans le tableau ne sont pas spécifiées dans les versions autogérées répertoriées sur GitHub. Mettez à jour votre version en procédant comme suit, selon le cas :

- Si vous n'avez aucun paramètre personnalisé pour le module complémentaire, exécutez la commande sous l'`To apply this release:`en-tête correspondant à la version vers laquelle vous souhaitez effectuer la mise à jour. GitHub
- Si vous avez personnalisé des paramètres, téléchargez le fichier manifeste avec la commande suivante. Modifiez *`https://raw.githubusercontent.com/aws/amazon-vpc-cni-k8s/v1.20.0/config/master/aws-k8s-cni.yaml`* l'URL de la version vers GitHub laquelle vous souhaitez effectuer la mise à jour.

```
curl -O https://raw.githubusercontent.com/aws/amazon-vpc-cni-k8s/v1.20.3/config/
master/aws-k8s-cni.yaml
```

Si nécessaire, modifiez le manifeste avec les paramètres personnalisés de la sauvegarde que vous avez effectuée lors d'une étape précédente, puis appliquez le manifeste modifié à votre cluster. Si vos nœuds n'ont pas accès aux référentiels privés Amazon EKS Amazon ECR à partir desquels les images sont extraites (voir les lignes commençant par `image:` dans le manifeste), vous devrez télécharger les images, les copier dans votre propre référentiel et modifier le manifeste pour extraire les images de votre référentiel. Pour de plus amples informations, veuillez consulter the section called "Copier une image dans un référentiel".

```
kubectl apply -f aws-k8s-cni.yaml
```

4. Vérifiez que la nouvelle version est maintenant installée sur votre cluster.

```
kubectl describe daemonset aws-node --namespace kube-system | grep amazon-k8s-cni: |
 cut -d : -f 3
```

L'exemple qui suit illustre un résultat.

```
v1.20.3
```

5. (Facultatif) Installez le `cni-metrics-helper` sur votre cluster. Il collecte les informations relatives à l'interface Elastic Network et aux adresses IP, les agrège au niveau du cluster et publie les statistiques sur Amazon. CloudWatch Pour plus d'informations, voir cni-metrics-helperci-dessous GitHub.

Configuration du plug-in Amazon VPC CNI pour utiliser IRSA

Le plug-in Amazon VPC CNI pour Kubernetes est le plug-in de mise en réseau des pods dans les clusters Amazon EKS. Le plug-in est chargé d'attribuer des adresses IP VPC aux nœuds Kubernetes et de configurer la mise en réseau nécessaire pour les pods sur chaque nœud.

> **Note**
>
> Le plug-in Amazon VPC CNI prend également en charge les identités des pods Amazon EKS. Pour de plus amples informations, veuillez consulter the section called "Attribution d'un rôle IAM".

Le plugin :

- Nécessite des autorisations AWS Identity and Access Management (IAM). Si votre cluster utilise la `IPv4` famille, les autorisations sont spécifiées dans la politique `AmazonEKS_CNI_Policy` AWS gérée. Si votre cluster utilise la famille `IPv6`, les autorisations doivent être ajoutées à une politique IAM que vous créez. Pour les instructions, consultez the section called "Créer une politique IAM pour les clusters qui utilisent la famille `IPv6`". Vous pouvez attacher cette politique au rôle IAM de nœud Amazon EKS ou à un rôle IAM distinct. Pour savoir comment attacher la politique au rôle IAM du nœud Amazon EKS, consultez the section called "Rôle IAM de nœud". Nous vous recommandons de l'affecter à un rôle séparé, comme détaillé dans cette rubrique.
- Lors de son déploiement, le plug-in crée et est configuré pour utiliser un compte de service Kubernetes nommé `aws-node`. Le compte de service est lié à un `clusterrole` Kubernetes nommé `aws-node`, qui reçoit les autorisations Kubernetes requises.

> **Note**
>
> Les pods du plug-in Amazon VPC CNI pour Kubernetes héritent des autorisations du rôle IAM du nœud Amazon EKS, sauf si l'accès à IMDS est bloqué. Pour plus d'informations, consultez Restreindre l'accès au profil d'instance affecté au composant master.

- Exige un cluster Amazon EKS existant. Pour en déployer un, consultez Démarrer.
- Nécessite un fournisseur de gestion des AWS identités et des accès (IAM) (IAM) (IAM) OpenID Connect (OIDC) existant pour votre cluster. Pour déterminer si vous en avez déjà un, ou pour en créer un, consultez the section called "Fournisseur IAM OIDC".

Étape 1 : créer le rôle IAM du plug-in Amazon VPC CNI pour Kubernetes

1. Déterminez la famille d'adresses IP de votre cluster.

```
aws eks describe-cluster --name my-cluster | grep ipFamily
```

L'exemple qui suit illustre un résultat.

```
"ipFamily": "ipv4"
```

La sortie peut renvoyer `ipv6` à la place.

2. Créez le rôle IAM. Vous pouvez utiliser `eksctl` ou `kubectl` et la AWS CLI pour créer votre rôle IAM.

   eksctl

   - Créez un rôle IAM et attachez la politique IAM au rôle avec la commande qui correspond à la famille d'adresses IP de votre cluster. La commande crée et déploie une AWS CloudFormation pile qui crée un rôle IAM, y attache la politique que vous spécifiez et annote le compte de `aws-node` service Kubernetes existant avec l'ARN du rôle IAM créé.

     - `IPv4`

       Remplacez *`my-cluster`* par votre propre valeur.

       ```
       eksctl create iamserviceaccount \
           --name aws-node \
           --namespace kube-system \
           --cluster my-cluster \
           --role-name AmazonEKSVPCCNIRole \
           --attach-policy-arn arn:aws: iam::aws:policy/AmazonEKS_CNI_Policy \
           --override-existing-serviceaccounts \
           --approve
       ```

     - `IPv6`

       Remplacez *`my-cluster`* par votre propre valeur. Remplacez *`111122223333`* par l'ID de votre compte et remplacez *`AmazonEKS_CNI_IPv6_Policy`* par le nom de votre politique `IPv6`. Si vous ne disposez pas d'une politique `IPv6`, consultez the section called "Créer une politique IAM pour les clusters qui utilisent la famille `IPv6`" pour en créer une. Pour utiliser `IPv6` avec votre cluster, ce dernier doit répondre à plusieurs exigences. Pour de plus amples informations, veuillez consulter the section called "IPv6".

       ```
       eksctl create iamserviceaccount \
       ```

```
    --name aws-node \
    --namespace kube-system \
    --cluster my-cluster \
    --role-name AmazonEKSVPCCNIRole \
    --attach-policy-arn arn:aws: iam::111122223333:policy/
AmazonEKS_CNI_IPv6_Policy \
    --override-existing-serviceaccounts \
    --approve
```

kubectl et la CLI AWS

i. Affichez l'URL du fournisseur OIDC de votre cluster.

```
aws eks describe-cluster --name my-cluster --query
 "cluster.identity.oidc.issuer" --output text
```

L'exemple qui suit illustre un résultat.

```
https://oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE
```

Si aucune sortie n'est renvoyée, vous devez créer un fournisseur OIDC IAM pour votre cluster.

ii. Copiez le contenu suivant dans un fichier nommé *vpc-cni-trust-policy.json*. Remplacez *111122223333* par votre ID de compte et *EXAMPLED539D4633E53DE1B71EXAMPLE* par la sortie renvoyée à l'étape précédente. Remplacez *region-code* par la AWS région dans laquelle se trouve votre cluster.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Federated": "arn:aws:iam::111122223333:oidc-provider/
oidc.eks.us-east-1.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
            },
            "Action": "sts:AssumeRoleWithWebIdentity",
            "Condition": {
                "StringEquals": {
                    "oidc.eks.us-east-1.amazonaws.com/id/
EXAMPLED539D4633E53DE1B71EXAMPLE:aud": "sts.amazonaws.com",
```

```
                    "oidc.eks.us-east-1.amazonaws.com/id/
EXAMPLED539D4633E53DE1B71EXAMPLE:sub": "system:serviceaccount:kube-system:aws-
node"
                }
            }
        }
    ]
}
```

iii. Créez le rôle. Vous pouvez remplacer *AmazonEKSVPCCNIRole* par n'importe quel nom que vous choisissez.

```
aws iam create-role \
  --role-name AmazonEKSVPCCNIRole \
  --assume-role-policy-document file://"vpc-cni-trust-policy.json"
```

iv. Attachez la politique IAM requise au rôle. Exécutez la commande qui correspond à la famille d'adresses IP de votre cluster.

- IPv4

```
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::aws:policy/AmazonEKS_CNI_Policy \
  --role-name AmazonEKSVPCCNIRole
```

- IPv6

  Remplacez *111122223333* par l'ID de votre compte et *AmazonEKS_CNI_IPv6_Policy* par le nom de votre politique IPv6. Si vous ne disposez pas d'une politique IPv6, consultez the section called "Créer une politique IAM pour les clusters qui utilisent la famille IPv6" pour en créer une. Pour utiliser IPv6 avec votre cluster, ce dernier doit répondre à plusieurs exigences. Pour de plus amples informations, veuillez consulter the section called "IPv6".

```
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::111122223333:policy/AmazonEKS_CNI_IPv6_Policy \
  --role-name AmazonEKSVPCCNIRole
```

v. Exécutez la commande suivante pour annoter le compte de service aws-node avec l'ARN du rôle IAM que vous avez créé précédemment. Remplacez les exemples de valeurs par vos propres valeurs.

```
kubectl annotate serviceaccount \
    -n kube-system aws-node \
    eks.amazonaws.com/role-arn=arn:aws: iam::111122223333:role/
AmazonEKSVPCCNIRole
```

3. (Facultatif) Configurez le type de point de terminaison du service AWS Security Token utilisé par votre compte de service Kubernetes. Pour de plus amples informations, veuillez consulter the section called "Critères d'évaluation STS".

Étape 2 : redéployer les pods du plug-in Amazon VPC CNI pour Kubernetes

1. Supprimez et recréez tous les pods existants associés au compte de service pour appliquer les variables d'environnement des informations d'identification. L'annotation n'est pas appliquée aux pods qui sont actuellement en cours d'exécution sans annotation. La commande suivante supprime les `aws-node` DaemonSet pods existants et les déploie avec l'annotation du compte de service.

```
kubectl delete Pods -n kube-system -l k8s-app=aws-node
```

2. Vérifiez que tous les pods ont redémarré.

```
kubectl get pods -n kube-system -l k8s-app=aws-node
```

3. Décrivez l'un des pods et confirmez la présence des variables d'environnement `AWS_WEB_IDENTITY_TOKEN_FILE` et `AWS_ROLE_ARN`. *`cpjw7`*Remplacez-le par le nom de l'un de vos pods renvoyé dans le résultat de l'étape précédente.

```
kubectl describe pod -n kube-system aws-node-cpjw7 | grep 'AWS_ROLE_ARN:\|
AWS_WEB_IDENTITY_TOKEN_FILE:'
```

   L'exemple qui suit illustre un résultat.

```
AWS_ROLE_ARN: arn:aws: iam::111122223333:role/AmazonEKSVPCCNIRole
      AWS_WEB_IDENTITY_TOKEN_FILE:  /var/run/secrets/eks.amazonaws.com/
serviceaccount/token
      AWS_ROLE_ARN: arn:aws: iam::111122223333:role/AmazonEKSVPCCNIRole
      AWS_WEB_IDENTITY_TOKEN_FILE:           /var/run/secrets/eks.amazonaws.com/
serviceaccount/token
```

Deux ensembles de résultats dupliqués sont renvoyés, car le pod comprend deux conteneurs. Les deux exemples présentent les mêmes valeurs.

Si votre Pod utilise le point de terminaison AWS régional, la ligne suivante est également renvoyée dans la sortie précédente.

```
AWS_STS_REGIONAL_ENDPOINTS=regional
```

Étape 3 : suppression de la politique CNI du rôle IAM de nœud

Si le rôle IAM de votre nœud Amazon EKS est actuellement associé à la politique `AmazonEKS_CNI_Policy` IAM (`IPv4`) ou à une IPv6 politique, et que vous avez créé un rôle IAM distinct, que vous lui avez attaché la politique et que vous l'avez attribué au compte de service `aws-node` Kubernetes, nous vous recommandons de supprimer la politique de votre rôle de nœud à l'aide de la commande AWS CLI correspondant à la famille d'adresses IP de votre cluster. Remplacez *`AmazonEKSNodeRole`* par le nom de votre rôle de nœud.

- `IPv4`

```
aws iam detach-role-policy --role-name AmazonEKSNodeRole --policy-arn arn:aws:
 iam::aws:policy/AmazonEKS_CNI_Policy
```

- `IPv6`

  Remplacez *`111122223333`* par l'ID de votre compte et *`AmazonEKS_CNI_IPv6_Policy`* par le nom de votre politique `IPv6`.

```
aws iam detach-role-policy --role-name AmazonEKSNodeRole --policy-arn arn:aws:
 iam::111122223333:policy/AmazonEKS_CNI_IPv6_Policy
```

Créer une politique IAM pour les clusters qui utilisent la famille **`IPv6`**

Si vous avez créé un cluster utilisant la famille `IPv6` et que celui-ci utilise la version `1.10.1` ou ultérieure du module complémentaire plug-in Amazon VPC CNI pour Kubernetes, vous devez créer une politique IAM destinée à être attribuée à un rôle IAM. Si vous disposez d'un cluster existant qui n'a pas été configuré avec la famille `IPv6` lors de sa création, alors, pour utiliser `IPv6`, vous devez

créer un nouveau cluster. Pour plus d'informations sur l'utilisation de IPv6 avec votre cluster, veuillez consulter the section called "IPv6".

1. Copiez la politique suivante et enregistrez-la dans un fichier appelé vpc-cni-ipv6-policy.json.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AssignIpv6Addresses",
                "ec2:DescribeInstances",
                "ec2:DescribeTags",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeInstanceTypes"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:network-interface/*"
            ]
        }
    ]
}
```

2. Créez la politique IAM.

```
aws iam create-policy --policy-name AmazonEKS_CNI_IPv6_Policy --policy-document
 file://vpc-cni-ipv6-policy.json
```

En savoir plus sur les modes et la configuration CNI Amazon VPC

Le plug-in CNI Amazon VPC pour Kubernetes fournit une mise en réseau pour les pods. Consultez le tableau suivant pour en savoir plus sur les fonctionnalités de mise en réseau disponibles.

Fonctionnalité de mise en réseau	En savoir plus
Configurez votre cluster pour attribuer des adresses IPv6 aux clusters, pods et services	the section called "IPv6"
Utilisez la traduction d'adresses réseau source IPv4 pour les pods	the section called "Trafic sortant"
Limitez le trafic réseau vers et depuis vos pods	the section called "Restrictions de trafic"
Personnalisez l'interface réseau secondaire dans les nœuds	the section called "Mise en réseau personnalisée"
Augmentez le nombre d'adresses IP pour votre nœud	the section called "Augmentation du nombre d'adresses IP"
Utilisez des groupes de sécurité pour le trafic réseau des pods	the section called "Groupes de sécurité pour les pods"
Utilisez plusieurs interfaces réseau pour les pods	the section called "Plusieurs interfaces"

En savoir plus sur IPv6 les adresses des clusters, des pods et des services

S'applique à : Pods avec EC2 instances Amazon et Fargate Pods

Par défaut, Kubernetes attribue des adresses `IPv4` à vos pods et services. Au lieu d'attribuer des adresses `IPv4` à vos pods et services, vous pouvez configurer votre cluster pour leur attribuer des adresses `IPv6`. Amazon EKS ne prend pas en charge les pods ou services double pile, bien que Kubernetes le permette. En conséquence, vous ne pouvez pas attribuer à la fois des adresses `IPv4` et des adresses `IPv6` aux pods et aux services.

Vous sélectionnez la famille IP que vous souhaitez utiliser pour votre cluster lorsque vous le créez. La famille d'adresses ne peut plus être modifiée après la création du cluster.

Pour un tutoriel sur le déploiement d'un cluster `IPv6` Amazon EKS, consultezthe section called "Déploiement".

Tenez compte des points suivants pour utiliser cette fonctionnalité :

Prise en charge des fonctionnalités **IPv6**

- Absence de prise en charge Windows : les pods et services Windows ne sont pas pris en charge.
- EC2 Nœuds basés sur Nitro requis : vous ne pouvez les utiliser qu'**IPv6**avec des nœuds AWS Amazon ou EC2 Fargate basés sur Nitro.
- EC2 et nœuds Fargate pris en charge : vous pouvez les utiliser **IPv6** the section called “Groupes de sécurité pour les pods” avec les nœuds Amazon et les EC2 nœuds Fargate.
- Outposts non pris en charge : vous ne pouvez pas les utiliser `IPv6` avec Amazon EKS sur AWS Outposts.
- FSx car Lustre n'est pas pris en charge : Le n'the section called “Amazon FSx pour Lustre”est pas pris en charge.
- Réseau personnalisé non pris en charge : si vous utilisiez auparavant the section called “Mise en réseau personnalisée” pour atténuer l'épuisement des adresses IP, vous pouvez utiliser `IPv6` à la place. Vous ne pouvez pas utiliser `IPv6` avec un réseau personnalisé. Si vous utilisez un réseau personnalisé pour l'isolation du réseau, vous devrez peut-être continuer à utiliser un réseau personnalisé et la famille `IPv4` pour vos clusters.

Attribution des adresses IP

- Services Kubernetes : seules des adresses `IPv6` sont attribuées aux services Kubernetes. Aucune IPv4 adresse ne leur est attribuée.
- Pods : une IPv6 adresse et une adresse locale de l'hôte IPv4 sont attribuées aux pods. L' IPv4 adresse locale de l'hôte est attribuée à l'aide d'un plug-in CNI local lié à un VPC CNI et l'adresse n'est pas signalée au plan de contrôle Kubernetes. Il n'est utilisé que lorsqu'un pod doit communiquer avec une IPv4 ressource externe dans un autre Amazon VPC ou sur Internet. L' IPv4 adresse locale de l'hôte accède SNATed (par VPC CNI) à l' IPv4 adresse principale de l'ENI principal du nœud de travail.
- Pods et services : les pods et services reçoivent uniquement des adresses `IPv6`, jamais des adresses `IPv4`. Lorsque les pods doivent communiquer avec des points de terminaison `IPv4` externes, ils utilisent la NAT directement sur le nœud. Cette fonctionnalité NAT intégrée élimine le besoin de DNS64 et NAT64. Pour le trafic nécessitant un accès à Internet public, le trafic du pod est soumis à une conversion d'adresse réseau source vers une adresse IP publique.
- Adresses de routage : lorsqu'un pod communique en dehors du VPC, son adresse `IPv6` d'origine est préservée (elle n'est pas convertie en adresse `IPv6` du nœud.). Ce trafic est routé directement via une passerelle Internet ou une passerelle Internet sortante uniquement.

- Nœuds : tous les nœuds reçoivent une adresse `IPv4` ainsi qu'une adresse `IPv6`.
- Pods Fargate : chaque pod Fargate reçoit une adresse `IPv6` à partir du CIDR spécifié pour le sous-réseau dans lequel il est déployé. L'unité matérielle sous-jacente qui exécute les Fargate Pods obtient une adresse `IPv6` et une adresse `IPv4` uniques attribuées CIDRs au sous-réseau dans lequel l'unité matérielle est déployée.

Utilisation de **`IPv6`** avec Amazon EKS

- Création d'un nouveau cluster : vous devez créer un nouveau cluster et spécifier l'utilisation de la famille `IPv6` pour ce cluster. Vous ne pouvez pas activer la famille `IPv6` pour un cluster que vous avez mis à jour à partir d'une version précédente. Pour obtenir des instructions sur la création d'un nouveau cluster, consultez la section Considérations.
- Utilisation d'une version récente de VPC CNI : déployez la version `1.10.1` ou ultérieure du module complémentaire Amazon VPC CNI. Cette version ou une version ultérieure est déployée par défaut. Après avoir déployé le module complémentaire, vous ne pouvez pas rétrograder votre module complémentaire Amazon VPC CNI vers une version inférieure à `1.10.1` sans supprimer au préalable tous les nœuds de tous les groupes de nœuds de votre cluster.
- Configurer le VPC CNI pour **`IPv6`** : Si vous utilisez des EC2 nœuds Amazon, vous devez configurer le module complémentaire Amazon VPC CNI avec délégation de préfixe IP et. `IPv6` Si vous choisissez la famille `IPv6` lors de la création de votre cluster, la version `1.10.1` du module complémentaire correspond par défaut à cette configuration. C'est le cas pour un module complémentaire auto-géré ou Amazon EKS. Pour de plus amples informations sur la délégation de préfixes IP, veuillez consulter the section called "Augmentation du nombre d'adresses IP".
- Configuration des adresses **`IPv4`** et **`IPv6`** : lorsque vous créez un cluster, le VPC et les sous-réseaux que vous spécifiez doivent disposer d'un bloc CIDR `IPv6` attribué au VPC et aux sous-réseaux que vous spécifiez. Ils doivent également avoir un bloc d'adresse CIDR `IPv4` qui leur est affecté. En effet, même si vous souhaitez uniquement utiliser `IPv6`, un VPC nécessite toujours un bloc d'adresse CIDR `IPv4` pour fonctionner. Pour plus d'informations, consultez Associer un bloc IPv6 CIDR à votre VPC dans le guide de l'utilisateur Amazon VPC.
- Attribuer automatiquement IPv6 des adresses aux nœuds : Lorsque vous créez vos nœuds, vous devez spécifier des sous-réseaux configurés pour attribuer automatiquement `IPv6` des adresses. Sinon, le déploiement des nœuds échouera. Par défaut, cette configuration est désactivée. Pour plus d'informations, consultez Modifier l'attribut d' IPv6 adressage de votre sous-réseau dans le guide de l'utilisateur Amazon VPC.

- Configuration des tables de routage pour utilise **IPv6** : les tables de routage attribuées à vos sous-réseaux contenir des routes pour les adresses IPv6. Pour de plus amples informations, veuillez consulter la section Migrer vers IPv6 du Guide de l'utilisateur Amazon VPC.
- Configuration des groupes de sécurité pour**IPv6** : vos groupes de sécurité doivent autoriser les adresses IPv6. Pour de plus amples informations, veuillez consulter la section Migrer vers IPv6 du Guide de l'utilisateur Amazon VPC.
- Configurer l'équilibreur de charge : utilisez la version 2.3.1 ou une version ultérieure du AWS Load Balancer Controller pour équilibrer la charge des applications HTTP à l'aide the section called “Répartition de la charge des applications” du ou du trafic réseau utilisant les deux pods, the section called “Répartition de charge réseau” l'IPv6un des équilibreurs de charge étant en mode IP, mais pas en mode instance. Pour de plus amples informations, veuillez consulter the section called “ AWS Contrôleur Load Balancer”.
- Ajout de la politique IAM **IPv6** : vous devez attacher une politique IAM IPv6 au rôle IAM du nœud ou au rôle IAM du plug-in CNI. Entre les deux, nous vous recommandons de le rattacher à un rôle CNI IAM. Pour plus d'informations, consultez the section called “Créer une politique IAM pour les clusters qui utilisent la famille IPv6” et the section called “Étape 1 : créer le rôle IAM du plug-in Amazon VPC CNI pour Kubernetes”.
- Évaluez tous les composants : effectuez une évaluation approfondie de vos applications, des modules complémentaires Amazon EKS et AWS des services auxquels vous intégrez avant de déployer IPv6 des clusters. Cela permet de s'assurer que tout fonctionne comme prévu avec IPv6.

Déploiement d'un cluster Amazon EKS **IPv6** et de nœuds Amazon Linux gérés

Dans ce didacticiel, vous allez déployer un IPv6 Amazon VPC, un cluster Amazon EKS appartenant à la IPv6 famille et un groupe de nœuds géré avec des nœuds EC2 Amazon Linux. Vous ne pouvez pas déployer de nœuds Amazon EC2 Windows dans un IPv6 cluster. Vous pouvez également déployer des nœuds Fargate dans votre cluster, mais ces instructions ne sont pas fournies dans cette rubrique pour des raisons de simplicité.

Conditions préalables

Avant de commencer le tutoriel, veuillez effectuer les opérations suivantes :

Installez et configurez les outils et ressources suivants dont vous aurez besoin pour créer et gérer un cluster Amazon EKS.

- Nous vous recommandons de vous familiariser avec tous les paramètres et de déployer un cluster avec les paramètres qui répondent à vos besoins. Pour plus d'informations, consultez the section called "Créer un cluster", the section called "Groupes de nœuds gérés", et les considérations relatives à ce sujet. Vous ne pouvez activer certains paramètres que lors de la création de votre cluster.
- L'outil de ligne de commande `kubectl` est installé sur votre appareil ou AWS CloudShell. La version peut correspondre à celle utilisée par votre cluster Kubernetes, ou différer d'au plus une version mineure, qu'elle soit antérieure ou plus récente. Par exemple, si la version de votre cluster est `1.29`, vous pouvez utiliser la version `kubectl` `1.28`, `1.29` ou `1.30`. Pour installer ou mettre à niveau `kubectl`, veuillez consulter the section called "Configurer kubectl et eksctl".
- Le principal de sécurité IAM que vous utilisez doit être autorisé à utiliser les rôles IAM Amazon EKS, les rôles liés à un service AWS CloudFormation, un VPC et les ressources associées. Pour plus d'informations, consultez les sections Actions et Utilisation des rôles liés aux services dans le Guide de l'utilisateur IAM.
- Si vous utilisez eksctl, installez la version `0.215.0` ou une version ultérieure sur votre ordinateur. Pour l'installer ou le mettre à jour, veuillez consulter Installation dans la documentation de `eksctl`.
- Version `2.12.3` ou version ultérieure `1.27.160` ou version ultérieure de l'interface de ligne de AWS commande (AWS CLI) installée et configurée sur votre appareil ou AWS CloudShell. Pour vérifier votre version actuelle, utilisez `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Les gestionnaires de packages tels que `yum` Homebrew pour macOS ont souvent plusieurs versions de retard sur la dernière version de la AWS CLI. `apt-get` Pour installer la dernière version, consultez la section Installation et configuration rapide avec aws configure dans le Guide de l'utilisateur de l'interface de ligne de AWS commande. La version de la AWS CLI installée AWS CloudShell peut également avoir plusieurs versions de retard par rapport à la dernière version. Pour le mettre à jour, consultez la section Installation de la AWS CLI dans votre répertoire de base dans le guide de AWS CloudShell l'utilisateur. Si vous utilisez le AWS CloudShell, vous devrez peut-être installer la version 2.12.3 ou ultérieure ou 1.27.160 ou ultérieure de la CLI, car la version de la AWSAWS CLI par défaut installée dans le AWS CloudShell peut être une version antérieure.

Vous pouvez utiliser eksctl ou CLI pour déployer un cluster `IPv6`.

Déployer un IPv6 cluster avec eksctl

a. Créez le fichier `ipv6-cluster.yaml`. Copiez la commande qui suit sur votre appareil. Si nécessaire, apportez les modifications suivantes à la commande, puis exécutez la commande modifiée :

- Remplacez *my-cluster* par un nom pour votre cluster. Un nom ne peut contenir que des caractères alphanumériques (sensibles à la casse) et des traits d'union. Il doit commencer par un caractère alphanumérique et ne peut pas dépasser 100 caractères. Le nom doit être unique dans la AWS région et le AWS compte dans lesquels vous créez le cluster.
- Remplacez *region-code* par n'importe quelle AWS région prise en charge par Amazon EKS. Pour obtenir la liste des AWS régions, consultez la section Points de terminaison et quotas Amazon EKS dans le guide de référence AWS général.
- La valeur de `version` correspond à la version de votre cluster. Pour plus d'informations, consultez les versions prises en charge par Amazon EKS.
- Remplacer *my-nodegroup* avec un nom pour votre groupe de nœuds. Le nom du groupe de nœuds ne peut pas dépasser 63 caractères. Il doit commencer par une lettre ou un chiffre, mais peut également inclure des tirets et des traits de soulignement pour les autres caractères.
- Remplacez *t3.medium* par tout type d'instance du système AWS Nitro.

```
cat >ipv6-cluster.yaml <<EOF
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-cluster
  region: region-code
  version: "X.XX"

kubernetesNetworkConfig:
  ipFamily: IPv6

addons:
  - name: vpc-cni
    version: latest
  - name: coredns
    version: latest
  - name: kube-proxy
    version: latest

iam:
  withOIDC: true

managedNodeGroups:
  - name: my-nodegroup
```

```
    instanceType: t3.medium
EOF
```

b. Créer votre cluster.

```
eksctl create cluster -f ipv6-cluster.yaml
```

La création du cluster prend plusieurs minutes. Ne continuez pas avant d'avoir vu la dernière ligne de sortie, qui ressemble à la sortie suivante.

```
[...]
[#]  EKS cluster "my-cluster" in "region-code" region is ready
```

c. Vérifiez que les pods par défaut se voient attribuer des adresses IPv6.

```
kubectl get pods -n kube-system -o wide
```

L'exemple qui suit illustre un résultat.

```
NAME                       READY   STATUS    RESTARTS   AGE     IP
                    NODE                                             NOMINATED NODE
 READINESS GATES
aws-node-rslts             1/1     Running   1          5m36s
 2600:1f13:b66:8200:11a5:ade0:c590:6ac8   ip-192-168-34-75.region-
code.compute.internal   <none>           <none>
aws-node-t74jh             1/1     Running   0          5m32s
 2600:1f13:b66:8203:4516:2080:8ced:1ca9   ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
coredns-85d5b4454c-cw7w2   1/1     Running   0          56m
 2600:1f13:b66:8203:34e5::                ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
coredns-85d5b4454c-tx6n8   1/1     Running   0          56m
 2600:1f13:b66:8203:34e5::1               ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
kube-proxy-btpbk           1/1     Running   0          5m36s
 2600:1f13:b66:8200:11a5:ade0:c590:6ac8   ip-192-168-34-75.region-
code.compute.internal   <none>           <none>
kube-proxy-jjk2g           1/1     Running   0          5m33s
 2600:1f13:b66:8203:4516:2080:8ced:1ca9   ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
```

d. Vérifiez que les adresses IPv6 sont attribuées aux services par défaut.

```
kubectl get services -n kube-system -o wide
```

L'exemple qui suit illustre un résultat.

```
NAME       TYPE        CLUSTER-IP          EXTERNAL-IP   PORT(S)         AGE
 SELECTOR
kube-dns   ClusterIP   fd30:3087:b6c2::a   <none>        53/UDP,53/TCP   57m   k8s-
app=kube-dns
```

e. (Facultatif) Déployez un exemple d'application ou déployez le Contrôleur d'équilibreur de charge AWS et un exemple d'application pour équilibrer la charge des applications HTTP avec the section called "Répartition de la charge des applications" ou le trafic réseau vers les pods the section called "Répartition de charge réseau" ou IPv6.

f. Une fois que vous avez terminé avec le cluster et les nœuds que vous avez créés pour ce tutoriel, vous devez nettoyer les ressources que vous avez créées avec la commande suivante.

```
eksctl delete cluster my-cluster
```

Déployer un IPv6 cluster avec une AWS CLI

Important

- Vous devez effectuer toutes les étapes de cette procédure avec le même utilisateur. Exécutez la commande suivante pour vérifier l'utilisateur actuel :

  ```
  aws sts get-caller-identity
  ```

- Vous devez effectuer toutes les étapes de cette procédure dans le même shell. Plusieurs étapes utilisent des variables définies dans les étapes précédentes. Les étapes qui utilisent des variables ne fonctionneront pas correctement si les valeurs de ces variables sont définies dans un autre shell. Si vous utilisez le AWS CloudShellpour effectuer la procédure suivante, n'oubliez pas que si vous n'interagissez pas avec le clavier ou le pointeur pendant environ 20 à 30 minutes, votre session shell se termine. Les processus en cours d'exécution ne sont pas considérés comme des interactions.

- Les instructions sont écrites pour le shell Bash et peuvent nécessiter des ajustements dans d'autres shells.

Remplacez toutes les valeurs d'exemple dans les étapes de cette procédure par vos propres valeurs.

a. Exécutez les commandes suivantes pour définir certaines variables utilisées dans les étapes suivantes. Remplacez *region-code* par la AWS région dans laquelle vous souhaitez déployer vos ressources. La valeur peut être n'importe quelle AWS région prise en charge par Amazon EKS. Pour obtenir la liste des AWS régions, consultez la section Points de terminaison et quotas Amazon EKS dans le guide de référence AWS général. Remplacez *my-cluster* par un nom pour votre cluster. Un nom ne peut contenir que des caractères alphanumériques (sensibles à la casse) et des traits d'union. Il doit commencer par un caractère alphanumérique et ne peut pas dépasser 100 caractères. Le nom doit être unique dans la AWS région et le AWS compte dans lesquels vous créez le cluster. Remplacer *my-nodegroup* avec un nom pour votre groupe de nœuds. Le nom du groupe de nœuds ne peut pas dépasser 63 caractères. Il doit commencer par une lettre ou un chiffre, mais peut également inclure des tirets et des traits de soulignement pour les autres caractères. Remplacez *111122223333* par votre ID de compte.

```
export region_code=region-code
export cluster_name=my-cluster
export nodegroup_name=my-nodegroup
export account_id=111122223333
```

b. Créez un Amazon VPC avec des sous-réseaux publics et privés qui répondent aux exigences Amazon EKS et IPv6.

   i. Exécutez la commande suivante pour définir une variable pour le nom de votre AWS CloudFormation pile. Vous pouvez remplacer *my-eks-ipv6-vpc* par n'importe quel nom que vous choisissez.

```
export vpc_stack_name=my-eks-ipv6-vpc
```

   ii. Créez un IPv6 VPC à l'aide d'un AWS CloudFormation modèle.

```
aws cloudformation create-stack --region $region_code --stack-name $vpc_stack_name
 \
  --template-url https://s3.us-west-2.amazonaws.com/amazon-eks/
cloudformation/2020-10-29/amazon-eks-ipv6-vpc-public-private-subnets.yaml
```

La création de la pile prend quelques minutes. Exécutez la commande suivante. Ne passez pas à l'étape suivante tant que le résultat de la commande n'est pas `CREATE_COMPLETE`.

```
aws cloudformation describe-stacks --region $region_code --stack-name
 $vpc_stack_name --query Stacks[].StackStatus --output text
```

iii. Récupérez IDs les sous-réseaux publics qui ont été créés.

```
aws cloudformation describe-stacks --region $region_code --stack-name
 $vpc_stack_name \
    --query='Stacks[].Outputs[?OutputKey==`SubnetsPublic`].OutputValue' --output
 text
```

L'exemple qui suit illustre un résultat.

```
subnet-0a1a56c486EXAMPLE,subnet-099e6ca77aEXAMPLE
```

iv. Activez l'option d'attribution automatique d'adresses `IPv6` pour les sous-réseaux publics qui ont été créés.

```
aws ec2 modify-subnet-attribute --region $region_code --subnet-id
 subnet-0a1a56c486EXAMPLE --assign-ipv6-address-on-creation
aws ec2 modify-subnet-attribute --region $region_code --subnet-id
 subnet-099e6ca77aEXAMPLE --assign-ipv6-address-on-creation
```

v. Récupérez les noms des sous-réseaux et des groupes de sécurité créés par le modèle à partir de la AWS CloudFormation pile déployée et stockez-les dans des variables pour une utilisation ultérieure.

```
security_groups=$(aws cloudformation describe-stacks --region $region_code --
stack-name $vpc_stack_name \
    --query='Stacks[].Outputs[?OutputKey==`SecurityGroups`].OutputValue' --output
 text)

public_subnets=$(aws cloudformation describe-stacks --region $region_code --stack-
name $vpc_stack_name \
    --query='Stacks[].Outputs[?OutputKey==`SubnetsPublic`].OutputValue' --output
 text)
```

```
private_subnets=$(aws cloudformation describe-stacks --region $region_code --
stack-name $vpc_stack_name \
    --query='Stacks[].Outputs[?OutputKey==`SubnetsPrivate`].OutputValue' --output
 text)

subnets=${public_subnets},${private_subnets}
```

c. Créez un rôle IAM de cluster et attachez-y la politique gérée par Amazon EKS IAM requise. Les clusters Kubernetes gérés par Amazon EKS appellent d'autres AWS services en votre nom afin de gérer les ressources que vous utilisez avec le service.

   i. Exécutez la commande suivante pour créer un fichier `eks-cluster-role-trust-policy.json` :

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

   ii. Exécutez la commande suivante pour définir une variable pour votre nom de rôle. Vous pouvez remplacer *myAmazonEKSClusterRole* par n'importe quel nom que vous choisissez.

```
export cluster_role_name=myAmazonEKSClusterRole
```

   iii. Créez le rôle.

```
aws iam create-role --role-name $cluster_role_name --assume-role-policy-document
 file://"eks-cluster-role-trust-policy.json"
```

   iv. Récupérez l'ARN du rôle IAM et le stockez dans une variable pour une étape ultérieure.

```
CLUSTER_IAM_ROLE=$(aws iam get-role --role-name $cluster_role_name --
query="Role.Arn" --output text)
```

   v. Attachez la politique IAM gérée par Amazon EKS au rôle.

```
aws iam attach-role-policy --policy-arn arn:aws: iam::aws:policy/
AmazonEKSClusterPolicy --role-name $cluster_role_name
```

d. Créer votre cluster.

```
aws eks create-cluster --region $region_code --name $cluster_name --kubernetes-
version 1.XX \
   --role-arn $CLUSTER_IAM_ROLE --resources-vpc-config subnetIds=
$subnets,securityGroupIds=$security_groups \
   --kubernetes-network-config ipFamily=ipv6
```

i. REMARQUE : il est possible que vous receviez un message d'erreur indiquant que l'une des zones de disponibilité de votre demande ne dispose pas d'une capacité suffisante pour créer un cluster Amazon EKS. Si cela se produit, la sortie de l'erreur contient les zones de disponibilité qui peuvent prendre en charge un nouveau cluster. Essayez à nouveau de créer votre cluster avec au moins deux sous-réseaux situés dans les zones de disponibilité prises en charge pour votre compte. Pour de plus amples informations, veuillez consulter the section called "Capacité insuffisante".

   La création du cluster prend quelques minutes. Exécutez la commande suivante. Ne passez pas à l'étape suivante tant que la sortie de la commande n'est pas `ACTIVE`.

```
aws eks describe-cluster --region $region_code --name $cluster_name --query
 cluster.status
```

e. Créez ou mettez à jour un fichier `kubeconfig` pour votre cluster afin de pouvoir communiquer avec votre cluster.

```
aws eks update-kubeconfig --region $region_code --name $cluster_name
```

   Par défaut, le fichier `config` est créé dans `~/.kube` ou la configuration du nouveau cluster est ajoutée à un fichier `config` existant dans `~/.kube`.

f. Créez un rôle IAM de nœud.

   i. Exécutez la commande suivante pour créer un fichier `vpc-cni-ipv6-policy.json` :

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

```
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AssignIpv6Addresses",
                "ec2:DescribeInstances",
                "ec2:DescribeTags",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeInstanceTypes"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:network-interface/*"
            ]
        }
    ]
}
```

ii. Créez la politique IAM.

```
aws iam create-policy --policy-name AmazonEKS_CNI_IPv6_Policy --policy-document
 file://vpc-cni-ipv6-policy.json
```

iii. Exécutez la commande suivante pour créer un fichier `node-role-trust-relationship.json` :

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

iv. Exécutez la commande suivante pour définir une variable pour votre nom de rôle. Vous pouvez remplacer *AmazonEKSNodeRole* par n'importe quel nom que vous choisissez.

```
export node_role_name=AmazonEKSNodeRole
```

v. Créez le rôle IAM.

```
aws iam create-role --role-name $node_role_name --assume-role-policy-document
 file://"node-role-trust-relationship.json"
```

vi. Attachez la politique IAM au rôle IAM.

```
aws iam attach-role-policy --policy-arn arn:aws: iam::$account_id:policy/
AmazonEKS_CNI_IPv6_Policy \
    --role-name $node_role_name
```

> **Important**
>
> Pour simplifier ce tutoriel, la politique est attachée à ce rôle IAM. Toutefois, dans un cluster de production, nous recommandons de rattacher la politique à un rôle IAM distinct. Pour de plus amples informations, veuillez consulter the section called "Configuration pour IRSA".

vii. Attachez deux politiques gérées IAM requises au rôle IAM.

```
aws iam attach-role-policy --policy-arn arn:aws: iam::aws:policy/
AmazonEKSWorkerNodePolicy \
  --role-name $node_role_name
aws iam attach-role-policy --policy-arn arn:aws: iam::aws:policy/
AmazonEC2ContainerRegistryReadOnly \
  --role-name $node_role_name
```

viii. Récupérez l'ARN du rôle IAM et le stockez dans une variable pour une étape ultérieure.

```
node_iam_role=$(aws iam get-role --role-name $node_role_name --query="Role.Arn" --
output text)
```

g. Créez un groupe de nœuds gérés.

i. Affichez les IDs sous-réseaux que vous avez créés à l'étape précédente.

```
echo $subnets
```

L'exemple qui suit illustre un résultat.

```
subnet-0a1a56c486EXAMPLE,subnet-099e6ca77aEXAMPLE,subnet-0377963d69EXAMPLE,subnet-0c05f819
```

ii. Créez le groupe de nœuds. Remplacez *0a1a56c486EXAMPLE*, *099e6ca77aEXAMPLE*, *0377963d69EXAMPLE*, et *0c05f819d5EXAMPLE* par les valeurs renvoyées dans la sortie de l'étape précédente. Assurez-vous de supprimer les virgules entre les sous-réseaux IDs de la sortie précédente dans la commande suivante. Vous pouvez remplacer *t3.medium* par tout type d'instance du système AWS Nitro.

```
aws eks create-nodegroup --region $region_code --cluster-name $cluster_name --
nodegroup-name $nodegroup_name \
    --subnets subnet-0a1a56c486EXAMPLE subnet-099e6ca77aEXAMPLE
 subnet-0377963d69EXAMPLE subnet-0c05f819d5EXAMPLE \
    --instance-types t3.medium --node-role $node_iam_role
```

La création du groupe de nœuds prend quelques minutes. Exécutez la commande suivante. Ne passez pas à l'étape suivante tant que le résultat renvoyé n'est pas `ACTIVE`.

```
aws eks describe-nodegroup --region $region_code --cluster-name $cluster_name --
nodegroup-name $nodegroup_name \
    --query nodegroup.status --output text
```

h. Vérifiez que les pods par défaut se voient attribuer des adresses `IPv6` dans la colonne `IP`.

```
kubectl get pods -n kube-system -o wide
```

L'exemple qui suit illustre un résultat.

```
NAME                       READY   STATUS    RESTARTS   AGE     IP
                    NODE                                                NOMINATED NODE
 READINESS GATES
aws-node-rslts             1/1     Running   1          5m36s
 2600:1f13:b66:8200:11a5:ade0:c590:6ac8   ip-192-168-34-75.region-
code.compute.internal   <none>           <none>
```

```
aws-node-t74jh             1/1     Running   0          5m32s
 2600:1f13:b66:8203:4516:2080:8ced:1ca9   ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
coredns-85d5b4454c-cw7w2   1/1     Running   0          56m
 2600:1f13:b66:8203:34e5::                ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
coredns-85d5b4454c-tx6n8   1/1     Running   0          56m
 2600:1f13:b66:8203:34e5::1               ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
kube-proxy-btpbk           1/1     Running   0          5m36s
 2600:1f13:b66:8200:11a5:ade0:c590:6ac8   ip-192-168-34-75.region-
code.compute.internal   <none>           <none>
kube-proxy-jjk2g           1/1     Running   0          5m33s
 2600:1f13:b66:8203:4516:2080:8ced:1ca9   ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
```

i. Vérifiez que les adresses `IPv6` sont attribuées aux services par défaut dans la colonne `IP`.

```
kubectl get services -n kube-system -o wide
```

L'exemple qui suit illustre un résultat.

```
NAME       TYPE        CLUSTER-IP          EXTERNAL-IP   PORT(S)         AGE
 SELECTOR
kube-dns   ClusterIP   fd30:3087:b6c2::a   <none>        53/UDP,53/TCP   57m   k8s-
app=kube-dns
```

j. (Facultatif) Déployez un exemple d'application ou déployez le Contrôleur d'équilibreur de charge AWS et un exemple d'application pour équilibrer la charge des applications HTTP avec the section called "Répartition de la charge des applications" ou le trafic réseau vers les pods the section called "Répartition de charge réseau" ou `IPv6`.

k. Une fois que vous avez terminé avec le cluster et les nœuds que vous avez créés pour ce tutoriel, vous devez nettoyer les ressources que vous avez créées avec les commandes suivantes. Assurez-vous que vous n'utilisez aucune des ressources en dehors de ce tutoriel avant de les supprimer.

   i. Si vous effectuez cette étape dans un shell différent de celui dans lequel vous avez effectué les étapes précédentes, définissez les valeurs de toutes les variables utilisées dans les étapes précédentes, en remplaçant les valeurs d'exemple par les valeurs que vous avez spécifiées lorsque vous avez effectué les étapes précédentes. Si vous effectuez cette étape dans le

même shell que celui dans lequel vous avez effectué les étapes précédentes, passez à l'étape suivante.

```
export region_code=region-code
export vpc_stack_name=my-eks-ipv6-vpc
export cluster_name=my-cluster
export nodegroup_name=my-nodegroup
export account_id=111122223333
export node_role_name=AmazonEKSNodeRole
export cluster_role_name=myAmazonEKSClusterRole
```

ii. Supprimez votre groupe de nœuds.

```
aws eks delete-nodegroup --region $region_code --cluster-name $cluster_name --
nodegroup-name $nodegroup_name
```

La suppression prend quelques minutes. Exécutez la commande suivante. Ne pas passer à l'étape suivante si une sortie est retournée.

```
aws eks list-nodegroups --region $region_code --cluster-name $cluster_name --query
 nodegroups --output text
```

iii. Supprimez le cluster.

```
aws eks delete-cluster --region $region_code --name $cluster_name
```

La suppression du cluster prend quelques minutes. Avant de continuer, assurez-vous que le cluster est supprimé avec la commande suivante.

```
aws eks describe-cluster --region $region_code --name $cluster_name
```

Ne passez pas à l'étape suivante tant que votre résultat n'est pas similaire au résultat suivant.

```
An error occurred (ResourceNotFoundException) when calling the DescribeCluster
 operation: No cluster found for name: my-cluster.
```

iv. Supprimez les ressources IAM que vous avez créées. Remplacez *AmazonEKS_CNI_IPv6_Policy* par le nom que vous avez choisi, si vous avez choisi un nom différent de celui utilisé dans les étapes précédentes.

```
aws iam detach-role-policy --role-name $cluster_role_name --policy-arn arn:aws:
 iam::aws:policy/AmazonEKSClusterPolicy
aws iam detach-role-policy --role-name $node_role_name --policy-arn arn:aws:
 iam::aws:policy/AmazonEKSWorkerNodePolicy
aws iam detach-role-policy --role-name $node_role_name --policy-arn arn:aws:
 iam::aws:policy/AmazonEC2ContainerRegistryReadOnly
aws iam detach-role-policy --role-name $node_role_name --policy-arn arn:aws: iam::
$account_id:policy/AmazonEKS_CNI_IPv6_Policy
aws iam delete-policy --policy-arn arn:aws: iam::$account_id:policy/
AmazonEKS_CNI_IPv6_Policy
aws iam delete-role --role-name $cluster_role_name
aws iam delete-role --role-name $node_role_name
```

v. Supprimez la AWS CloudFormation pile qui a créé le VPC.

```
aws cloudformation delete-stack --region $region_code --stack-name $vpc_stack_name
```

Activer l'accès Internet sortant pour les Pods

S'applique à : nœuds Linux `IPv4` Fargate, nœuds Linux avec instances Amazon EC2

Si vous avez déployé votre cluster à l'aide de la famille `IPv6`, les informations de cette rubrique ne s'appliquent pas à votre cluster, car les adresses `IPv6` ne sont pas traduites en réseau. Pour plus d'informations sur l'utilisation des adresses `IPv6` avec votre cluster, veuillez consulter the section called "IPv6".

Par défaut, chaque pod de votre cluster se voit attribuer une adresse `IPv4` privée provenant d'un bloc CIDR (Classless Inter-Domain Routing) associé au VPC dans lequel le pod est déployé. Les pods du même VPC communiquent ensemble via l'adresse IP privée. Lorsqu'un pod communique avec une adresse `IPv4` qui ne se trouve pas dans un bloc CIDR associé à votre VPC, le plug-in Amazon VPC CNI (pour Linux ou Windows) traduit l'adresse `IPv4` du pod en adresse `IPv4` privée principale de l'interface réseau Elastic principale du nœud sur lequel le pod est exécuté, par défaut *.

> **Note**
>
> Pour les nœuds Windows, il y a des détails supplémentaires à prendre en compte. Par défaut, le plug-in CNI VPC pour Windows est défini avec une configuration réseau dans laquelle le trafic vers une destination au sein du même VPC est exclu pour la SNAT. Cela signifie que la SNAT est désactivée pour la communication interne du VPC et que

l'adresse IP attribuée à un pod est acheminable au sein du VPC. Mais le trafic vers une destination en dehors du VPC a l'adresse IP source du pod SNAT vers l'adresse IP principale de l'ENI de l'instance. Cette configuration par défaut pour Windows garantit que le pod peut accéder à des réseaux extérieurs à votre VPC de la même manière que l'instance hôte.

Compte tenu de ce comportement :

- Vos pods ne peuvent communiquer avec des ressources Internet que si le nœud sur lequel ils s'exécutent dispose d'une adresse IP publique ou Elastic qui lui est attribuée et se trouve dans un sous-réseau public. Une table de routage associée d'un sous-réseau public comporte une route vers une passerelle Internet. Nous recommandons de déployer les nœuds dans des sous-réseaux privés, dans la mesure du possible.
- Pour les versions antérieures à `1.8.0` du plug-in, les ressources qui se trouvent dans des réseaux ou des VPC connectés au VPC de votre cluster à l'aide du peering VPC, d'un VPC de transit ou de AWS Direct Connect ne peuvent pas initier de communication avec vos pods derrière des interfaces réseau Elastic secondaires. Cependant, vos pods peuvent initier une communication avec ces ressources et recevoir des réponses de leur part.

Si l'une des affirmations suivantes est vraie dans votre environnement, modifiez la configuration par défaut à l'aide de la commande suivante.

- Vous disposez de ressources dans des réseaux ou des VPC connectés au VPC de votre cluster à l'aide d'un peering VPC, d'un VPC de transit ou de AWS Direct Connect qui doivent initier la communication avec vos pods à l'aide d'une adresse `IPv4`, et la version de votre plug-in est antérieure à `1.8.0`.
- Vos pods se trouvent dans un sous-réseau privé et doivent communiquer en sortie vers Internet. Le sous-réseau dispose d'une route vers une passerelle NAT.

```
kubectl set env daemonset -n kube-system aws-node AWS_VPC_K8S_CNI_EXTERNALSNAT=true
```

> **Note**
>
> Les variables de configuration `AWS_VPC_K8S_CNI_EXTERNALSNAT` et `AWS_VPC_K8S_CNI_EXCLUDE_SNAT_CIDRS` CNI ne s'appliquent pas aux nœuds Windows. La désactivation de la SNAT n'est pas prise en charge pour Windows. Pour exclure

une liste de `IPv4` CIDR du SNAT, vous pouvez définir cela en spécifiant le paramètre `ExcludedSnatCIDRs` dans le script de démarrage Windows. Pour plus d'informations sur ce paramètre, veuillez consulter la rubrique the section called “Paramètres de configuration du script d'amorçage”.

Mise en réseau de l'hôte

* Si les spécifications d'un pod contiennent `hostNetwork=true` (la valeur par défaut est `false`), son adresse IP n'est pas traduite en une autre adresse. C'est le cas, par défaut, du plug-in `kube-proxy` et Amazon VPC CNI pour les pods Kubernetes qui s'exécutent sur votre cluster. Pour ces pods, l'adresse IP est la même que l'adresse IP principale du nœud, donc l'adresse IP du pod n'est pas traduite. Pour plus d'informations sur la configuration `hostNetwork` d'un pod, consultez PodSpec v1 core dans la référence API Kubernetes.

Limitation du trafic des pods à l'aide des politiques réseau Kubernetes

Présentation de

Par défaut, Kubernetes n'impose aucune restriction concernant les adresses IP, les ports ou les connexions entre les pods d'un même cluster, ni entre vos pods et des ressources situées dans d'autres réseaux. Vous pouvez utiliser une politique réseau Kubernetes pour limiter le trafic réseau vers et depuis vos pods. Pour plus d'informations, consultez Politiques réseau dans la documentation Kubernetes.

Politique réseau standard

Vous pouvez utiliser la norme `NetworkPolicy` pour segmenter le pod-to-pod trafic dans le cluster. Ces politiques réseau s'appliquent aux couches 3 et 4 du modèle de réseau OSI, ce qui vous permet de contrôler le flux de trafic au niveau de l'adresse IP ou du port au sein de votre cluster Amazon EKS. Les politiques réseau standard sont limitées au niveau de l'espace de noms.

Cas d'utilisation

- Segmentez le trafic réseau entre les charges de travail afin de garantir que seules les applications associées puissent communiquer entre elles.
- Isolez les locataires au niveau de l'espace de noms à l'aide de politiques visant à appliquer la séparation des réseaux.

Exemple

Dans la politique ci-dessous, le trafic sortant des pods d'applications Web dans l'espace de noms Sun est restreint.

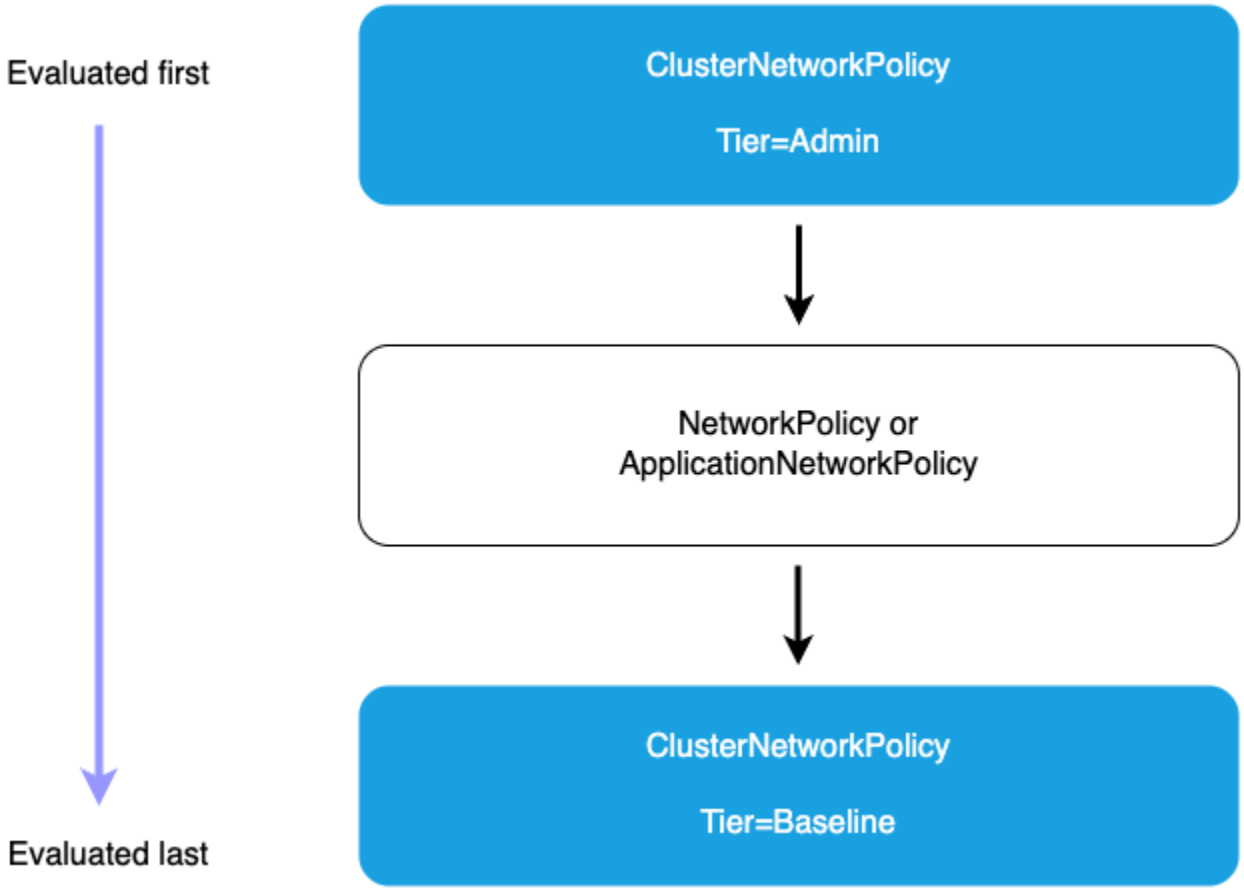
```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: webapp-egress-policy
  namespace: sun
spec:
  podSelector:
    matchLabels:
      role: webapp
  policyTypes:
  - Egress
  egress:
  - to:
    - namespaceSelector:
        matchLabels:
          name: moon
      podSelector:
        matchLabels:
          role: frontend
    ports:
    - protocol: TCP
      port: 8080
  - to:
    - namespaceSelector:
        matchLabels:
          name: stars
      podSelector:
        matchLabels:
          role: frontend
    ports:
    - protocol: TCP
      port: 8080
```

La politique s'applique aux modules dont l'étiquette figure `role: webapp` dans l'espace de `sun` noms.

- Trafic autorisé : pods dont l'étiquette figure `role: frontend` dans l'espace de `moon` noms du port TCP `8080`

- Trafic autorisé : Pods portant le label role : frontend dans l'espace de `stars` noms du port TCP `8080`
- Trafic bloqué : tout autre trafic sortant provenant `webapp` des pods est implicitement refusé

Politique réseau de l'administrateur (ou du cluster)



Vous pouvez utiliser le `ClusterNetworkPolicy` pour appliquer une norme de sécurité réseau qui s'applique à l'ensemble du cluster. Au lieu de définir et de gérer de manière répétitive une politique distincte pour chaque espace de noms, vous pouvez utiliser une seule stratégie pour gérer de manière centralisée les contrôles d'accès au réseau pour les différentes charges de travail du cluster, quel que soit leur espace de noms.

Cas d'utilisation

- Gérez de manière centralisée les contrôles d'accès au réseau pour toutes les charges de travail (ou un sous-ensemble de celles-ci) de votre cluster EKS.
- Définissez une posture de sécurité réseau par défaut sur l'ensemble du cluster.

- Étendez les normes de sécurité organisationnelles au périmètre du cluster de manière plus efficace sur le plan opérationnel.

Exemple

Dans la politique ci-dessous, vous pouvez explicitement bloquer le trafic du cluster en provenance d'autres espaces de noms afin d'empêcher l'accès réseau à un espace de noms de charge de travail sensible.

```
apiVersion: networking.k8s.aws/v1alpha1
kind: ClusterNetworkPolicy
metadata:
  name: protect-sensitive-workload
spec:
  tier: Admin
  priority: 10
  subject:
    namespaces:
      matchLabels:
        kubernetes.io/metadata.name: earth
  ingress:
    - action: Deny
      from:
      - namespaces:
          matchLabels: {} # Match all namespaces.
      name: select-all-deny-all
```

Remarques importantes

Les politiques réseau du plug-in Amazon VPC CNI pour Kubernetes sont prises en charge dans les configurations répertoriées ci-dessous.

- Version 1.21.0 (ou ultérieure) du plugin Amazon VPC CNI pour les politiques réseau standard et d'administration.
- Cluster configuré pour les adresses `IPv4` ou `IPv6`.
- Vous pouvez utiliser les politiques réseau avec des groupes de sécurité pour les pods. Grâce aux stratégies réseau, vous pouvez contrôler toutes les communications au sein du cluster. Avec les groupes de sécurité pour Pods, vous pouvez contrôler l'accès aux AWS services depuis les applications d'un Pod.

- Vous pouvez utiliser les stratégies réseau avec mise en réseau personnalisée et délégation de préfixes.

Considérations

Architecture

- Lorsque vous appliquez les politiques réseau du plug-in Amazon VPC CNI pour Kubernetes à votre cluster à l'aide du plug-in Amazon VPC CNI pour Kubernetes, vous pouvez appliquer les politiques aux nœuds Amazon Linux uniquement. EC2 Il n'est pas possible d'appliquer ces politiques aux nœuds Fargate ni aux nœuds Windows.
- Les politiques réseau ne s'appliquent qu'à une seule famille d'adresses IP, soit `IPv4`, soit `IPv6`, mais pas aux deux en même temps. Dans un cluster `IPv4`, le plug-in VPC CNI attribue des adresses `IPv4` aux pods et applique les politiques `IPv4` correspondantes. Dans un cluster `IPv6`, le plug-in VPC CNI attribue des adresses `IPv6` aux pods et applique les politiques `IPv6` correspondantes. Toute règle de politique réseau `IPv4` appliquée à un cluster `IPv6` est ignorée. Toute règle de politique réseau `IPv6` appliquée à un cluster `IPv4` est ignorée.

Politiques réseau

- Les politiques réseau s'appliquent uniquement aux pods faisant partie d'un déploiement. Les pods autonomes qui n'ont pas de `metadata.ownerReferences` définies ne peuvent pas se voir appliquer de politique réseau.
- Il est possible d'appliquer plusieurs politiques réseau à un même pod. Lorsque plusieurs politiques ciblent le même pod, toutes les politiques sont appliquées simultanément.
- Le nombre maximal de combinaisons de ports et de protocoles pour une même plage d'adresses IP (CIDR) est de 24, toutes politiques réseau confondues. Des sélecteurs tels que `namespaceSelector` Resolve to a un ou plusieurs CIDRs. Si plusieurs sélecteurs aboutissent à la même plage CIDR ou si une même plage CIDR directe est spécifiée plusieurs fois, dans une ou plusieurs politiques réseau, chacune de ces occurrences compte dans cette limite.
- Pour tous les services Kubernetes, le port du service doit être identique au port du conteneur. Si vous utilisez des ports nommés, vous devez employer le même nom dans la spécification du service.

Politiques du réseau d'administration

1. Politiques du niveau administrateur (évaluées en premier) : toutes les politiques du niveau administrateur ClusterNetworkPolicies sont évaluées avant les autres politiques. Au niveau administrateur, les politiques sont traitées par ordre de priorité (le numéro de priorité le plus bas en premier). Le type d'action détermine ce qui se passe ensuite.
   - Refuser l'action (priorité la plus élevée) : lorsqu'une politique d'administration associée à une action de refus correspond au trafic, ce trafic est immédiatement bloqué, quelles que soient les autres politiques. Aucune autre ClusterNetworkPolicy NetworkPolicy règle n'est traitée. Cela garantit que les contrôles de sécurité à l'échelle de l'organisation ne peuvent pas être remplacés par des politiques au niveau de l'espace de noms.
   - Autoriser l'action : une fois les règles de refus évaluées, les politiques d'administration comportant des actions d'autorisation sont traitées par ordre de priorité (le numéro de priorité le plus bas en premier). Lorsqu'une action Autoriser correspond, le trafic est accepté et aucune autre évaluation de politique n'est effectuée. Ces politiques peuvent accorder l'accès à plusieurs espaces de noms en fonction de sélecteurs d'étiquettes, offrant ainsi un contrôle centralisé des charges de travail pouvant accéder à des ressources spécifiques.
   - Action de transfert : les actions de transfert prévues dans les politiques du niveau administrateur délèguent la prise de décision aux niveaux inférieurs. Lorsque le trafic correspond à une règle Pass, l'évaluation ignore toutes les règles de niveau administrateur restantes pour ce trafic et passe directement au NetworkPolicy niveau. Cela permet aux administrateurs de déléguer explicitement le contrôle de certains modèles de trafic aux équipes d'application. Par exemple, vous pouvez utiliser les règles Pass pour déléguer la gestion du trafic intra-espace à des administrateurs d'espaces de noms tout en maintenant des contrôles stricts sur l'accès externe.
2. Niveau de politique réseau : si aucune politique de niveau administrateur ne correspond à Deny ou Allow, ou si une action Pass correspond, les NetworkPolicy ressources délimitées par un espace de noms sont ensuite évaluées. Ces politiques fournissent un contrôle précis au sein des espaces de noms individuels et sont gérées par les équipes d'application. Les politiques étendues à l'espace de noms ne peuvent être que plus restrictives que les politiques d'administration. Ils ne peuvent pas annuler la décision de refus d'une politique d'administration, mais ils peuvent restreindre davantage le trafic autorisé ou approuvé par les politiques d'administration.
3. Politiques d'administration du niveau de référence : si aucune politique d'administration ou limitée à un espace de noms ne correspond au trafic, le niveau de référence est évalué. ClusterNetworkPolicies Ils fournissent des postures de sécurité par défaut qui peuvent être remplacées par des politiques délimitées par des espaces de noms, permettant aux administrateurs de définir des valeurs par défaut à l'échelle de l'organisation tout en donnant aux

équipes la flexibilité de les personnaliser selon les besoins. Les politiques de base sont évaluées par ordre de priorité (le numéro de priorité le plus bas en premier).

4. Refus par défaut (si aucune politique ne correspond) : ce deny-by-default comportement garantit que seules les connexions explicitement autorisées sont autorisées, garantissant ainsi un niveau de sécurité élevé.

Migration

- Si votre cluster utilise actuellement une solution tierce pour gérer les politiques réseau Kubernetes, vous pouvez réutiliser ces mêmes politiques avec le plug-in Amazon VPC CNI pour Kubernetes. Cependant, vous devez supprimer la solution existante afin d'éviter qu'elle ne gère les mêmes politiques en parallèle.

> **Warning**
>
> Il est recommandé, après avoir supprimé la solution de politique réseau, de remplacer tous les nœuds sur lesquels cette solution était appliquée. Cette mesure permet d'éviter que des règles de trafic résiduelles laissées par un pod de la solution, notamment si celui-ci s'arrête brusquement, ne restent actives sur les nœuds.

Installation

- La fonctionnalité de stratégie réseau crée et nécessite une définition de ressource personnalisée (CRD) de `PolicyEndpoint` appelée `policyendpoints.networking.k8s.aws`. Les objets du `PolicyEndpoint` de la ressource personnalisée sont gérés par Amazon EKS. Vous ne devez ni modifier ni supprimer ces ressources.
- Si vous exécutez des pods qui utilisent les informations d'identification IAM du rôle d'instance ou si vous vous connectez à l' EC2 IMDS, veillez à vérifier les politiques réseau susceptibles de bloquer l'accès à l' EC2 IMDS. Vous devrez peut-être ajouter une politique réseau pour autoriser l'accès à l' EC2 IMDS. Pour plus d'informations, consultez la section Métadonnées de l'instance et données utilisateur dans le guide de EC2 l'utilisateur Amazon.

  Les pods qui utilisent des rôles IAM pour les comptes de service ou EKS Pod Identity n'accèdent pas à EC2 IMDS.

- Le plug-in Amazon VPC CNI pour Kubernetes n'applique pas les politiques réseau aux interfaces réseau supplémentaires pour chaque pod, mais uniquement à l'interface principale de chaque pod (`eth0`). Cela concerne les architectures suivantes :
  - `IPv6` pods dont la variable `ENABLE_V4_EGRESS` est définie sur `true`. Cette variable permet à la fonction de `IPv4` sortie de connecter les IPv6 pods à des `IPv4` points de terminaison tels que ceux situés en dehors du cluster. La fonction de `IPv4` sortie fonctionne en créant une interface réseau supplémentaire avec une adresse de boucle locale. IPv4
  - Lorsque vous utilisez des plugins réseau chaînés tels que Multus. Comme ces plug-ins ajoutent des interfaces réseau à chaque pod, les politiques réseau ne sont pas appliquées aux plug-ins réseau chaînés.

Restriction du trafic réseau des pods à l'aide des politiques réseau Kubernetes

Les politiques réseau Kubernetes permettent de restreindre le trafic réseau entrant et sortant de vos pods. Pour plus d'informations, consultez Politiques réseau dans la documentation Kubernetes.

Pour utiliser cette fonctionnalité, vous devez effectuer les configurations suivantes :

1. Configurez l'application des politiques au démarrage du pod. Cette opération s'effectue dans le conteneur `aws-node` du `DaemonSet` VPC CNI.
2. Activez le paramètre de politique réseau pour le module complémentaire.
3. Configuration de votre cluster pour qu'il utilise la politique réseau Kubernetes

Avant de commencer, consultez les points à prendre en compte. Pour de plus amples informations, veuillez consulter the section called "Considérations".

Conditions préalables

Prérequis pour l'utilisation de cette fonctionnalité :

Version minimale du cluster

Un cluster Amazon EKS existant. Pour en déployer un, consultez Démarrer. Le cluster doit exécuter l'une des versions de Kubernetes et de la plateforme répertoriées dans le tableau suivant. Toute version de Kubernetes et de la plateforme ultérieure à celles indiquées est également prise en charge. Vous pouvez vérifier votre version actuelle de Kubernetes *`my-cluster`* en remplaçant la commande suivante par le nom de votre cluster, puis en exécutant la commande modifiée :

```
aws eks describe-cluster --name my-cluster --query cluster.version --output text
```

Version de Kubernetes	Version de la plateforme
1.27.4	eks.5
1.26.7	eks.6

Version minimale du VPC CNI

Pour créer à la fois des politiques réseau Kubernetes standard et des politiques réseau d'administration, vous devez exécuter 1.21 la version du plugin VPC CNI. Vous pouvez voir la version que vous utilisez actuellement à l'aide de la commande suivante.

```
kubectl describe daemonset aws-node --namespace kube-system | grep amazon-k8s-cni: |
 cut -d : -f 3
```

Si votre version est antérieure à 1.21, consultez la section the section called "Mise à jour (module complémentaire EKS)" pour mettre à jour vers la version 1.21 ou ultérieure.

Version minimale du noyau Linux

Vos nœuds doivent avoir la version du noyau Linux 5.10 ou ultérieure. Vous pouvez vérifier votre version du noyau avec uname -r. Si vous utilisez les dernières versions d'Amazon Linux optimisé pour Amazon EKS, d'Amazon Linux accéléré optimisé pour Amazon EKS et de Bottlerocket AMIs, ils disposent déjà de la version de noyau requise. AMIs

La version v20231116 ou ultérieure de l'AMI Amazon Linux accélérée et optimisée pour Amazon EKS dispose de la version 5.10 du noyau.

Étape 1 : configurer l'application des politiques au démarrage du pod

Le plug-in Amazon VPC CNI pour Kubernetes configure les politiques réseau pour les pods en parallèle du processus de provisionnement des pods. Tant que toutes les politiques ne sont pas appliquées au nouveau pod, les conteneurs de ce pod démarrent avec une politique d'autorisation par défaut. Ce mode de fonctionnement est appelé mode standard. Une politique d'autorisation par défaut signifie que tout le trafic entrant et sortant est autorisé vers et depuis les nouveaux pods. Par

exemple, aucune règle de pare-feu ne sera appliquée aux pods (tout le trafic est autorisé) tant que le nouveau pod ne sera pas mis à jour avec les politiques actives.

Lorsque la variable `NETWORK_POLICY_ENFORCING_MODE` est définie sur `strict`, les pods qui utilisent le plug-in VPC CNI commencent par une politique de refus par défaut, puis les politiques sont configurées. Ce mode de fonctionnement est appelé mode strict. En mode strict, vous devez définir une politique réseau pour chaque point de terminaison auquel vos pods doivent accéder dans le cluster. Cette exigence s'applique également aux pods CoreDNS. La politique de refus par défaut n'est pas configurée pour les pods disposant d'un réseau hôte.

Vous pouvez modifier la politique réseau par défaut en définissant la variable d'environnement `NETWORK_POLICY_ENFORCING_MODE` sur `strict` dans le conteneur `aws-node` du `DaemonSet` VPC CNI.

```
env:
  - name: NETWORK_POLICY_ENFORCING_MODE
    value: "strict"
```

Étape 2◦: activer le paramètre de politique réseau pour le module complémentaire

La fonctionnalité de politique réseau utilise le port `8162` du nœud pour les métriques par défaut. En outre, la fonctionnalité utilise le port `8163` pour les sondes de santé. Si vous exécutez une autre application sur les nœuds ou dans les pods qui doivent utiliser ces ports, l'application ne s'exécutera pas. À partir de la version VPC CNI `v1.14.1` ou ultérieure, vous pouvez modifier ces ports.

Suivez la procédure ci-dessous pour activer le paramètre de politique réseau du module complémentaire.

AWS Management Console

1. Ouvrez la console Amazon EKS.
2. Dans le panneau de navigation de gauche, sélectionnez Clusters, puis sélectionnez le nom du cluster pour lequel vous souhaitez configurer le module complémentaire Amazon VPC CNI.
3. Choisissez l'onglet Modules complémentaires.
4. Cochez la case en haut à droite de la zone du module complémentaire, puis sélectionnez Edit (Modifier).
5. Sur la page Configurer **`Amazon VPC CNI`** :
   a. Sélectionnez une version `v1.14.0-eksbuild.3` ou ultérieure dans la liste Version.

b. Sélectionnez Paramètres de configuration facultatifs.

c. Saisissez la clé JSON `"enableNetworkPolicy":` et valeur `"true"` dans Valeurs de configuration. Le texte obtenu doit être un objet JSON valide. Si cette clé et cette valeur sont les seules données de la zone de texte, entourez-les d'accolades `{ }`.

   L'exemple suivant présente une configuration où la fonctionnalité de politique réseau est activée, et où les sondes de surveillance et métriques utilisent les numéros de port par défaut :

```
{
    "enableNetworkPolicy": "true",
    "nodeAgent": {
        "healthProbeBindAddr": "8163",
        "metricsBindAddr": "8162"
    }
}
```

Helm

Si vous avez installé le plug-in Amazon VPC CNI pour Kubernetes via `helm`, vous pouvez mettre à jour la configuration pour modifier les ports.

1. Exécutez la commande suivante pour modifier les ports. Définissez le numéro de port dans la valeur de la clé `nodeAgent.metricsBindAddr` ou de la clé `nodeAgent.healthProbeBindAddr`.

```
helm upgrade --set nodeAgent.metricsBindAddr=8162 --set
 nodeAgent.healthProbeBindAddr=8163 aws-vpc-cni --namespace kube-system eks/aws-vpc-
cni
```

kubectl

1. Ouvrez le `aws-node DaemonSet` dans votre éditeur.

```
kubectl edit daemonset -n kube-system aws-node
```

2. Remplacez les numéros de port dans les arguments de commande suivants pour le `args:` du conteneur `aws-network-policy-agent` dans le manifeste daemonset `aws-node` du VPC CNI.

```
    - args:
            - --metrics-bind-addr=:8162
            - --health-probe-bind-addr=:8163
```

Étape 3 : configurer votre cluster pour qu'il utilise la politique réseau Kubernetes

Cette configuration peut être appliquée à un module complémentaire Amazon EKS ou à un module complémentaire autogéré.

Module complémentaire d'Amazon EKS

À l'aide de la AWS CLI, vous pouvez configurer le cluster pour utiliser les politiques réseau Kubernetes en exécutant la commande suivante. Remplacez `my-cluster` par le nom de votre cluster et l'ARN du rôle IAM par le rôle que vous utilisez.

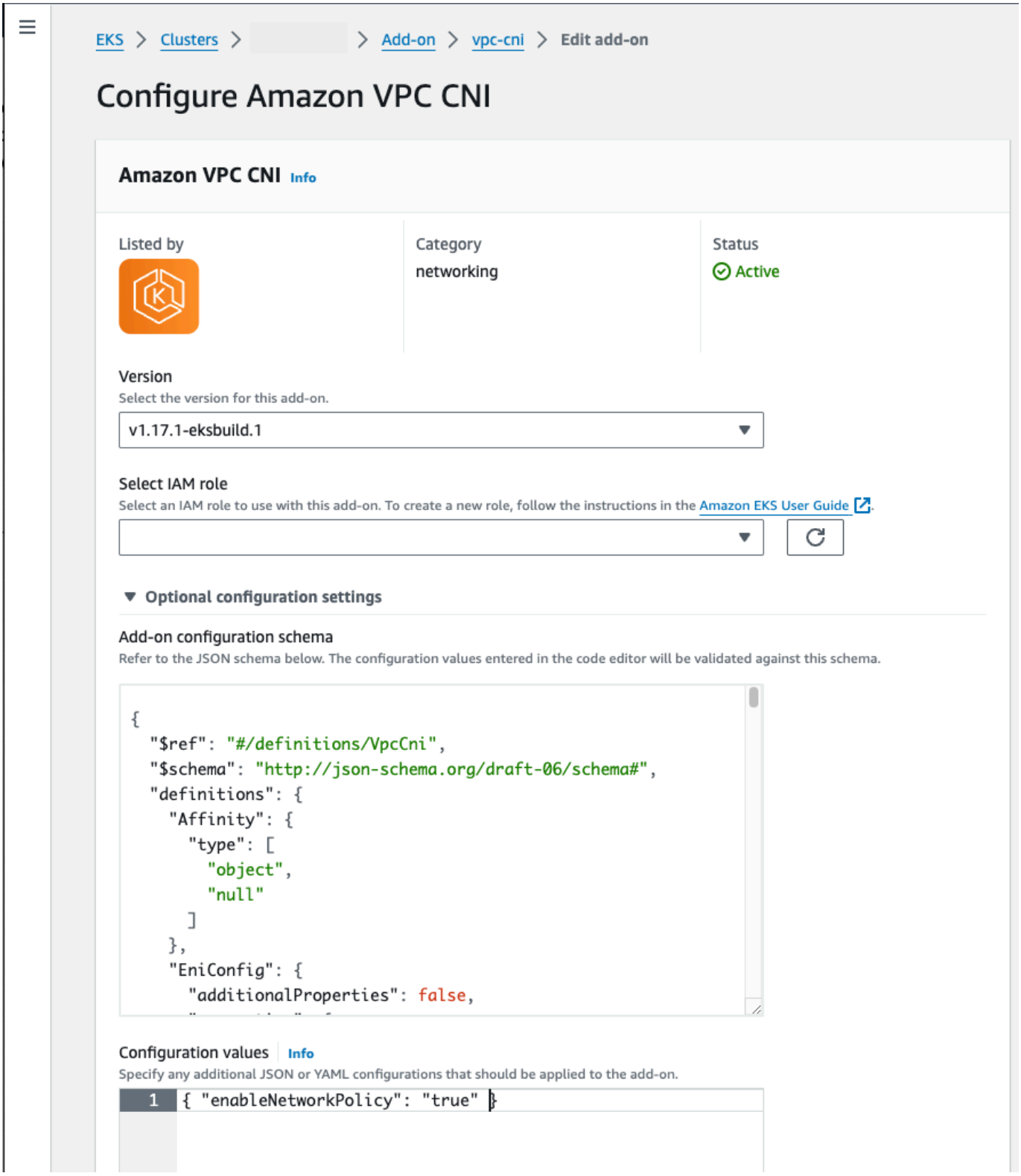
```
aws eks update-addon --cluster-name my-cluster --addon-name vpc-cni --addon-version
 v1.14.0-eksbuild.3 \
    --service-account-role-arn arn:aws: iam::123456789012:role/AmazonEKSVPCCNIRole \
    --resolve-conflicts PRESERVE --configuration-values '{"enableNetworkPolicy":
 "true"}'
```

Pour le configurer à l'aide AWS de la console de gestion, suivez les étapes ci-dessous :

1. Ouvrez la console Amazon EKS.
2. Dans le panneau de navigation de gauche, sélectionnez Clusters, puis sélectionnez le nom du cluster pour lequel vous souhaitez configurer le module complémentaire Amazon VPC CNI.
3. Choisissez l'onglet Modules complémentaires.
4. Cochez la case en haut à droite de la zone du module complémentaire, puis sélectionnez Edit (Modifier).
5. Sur la page Configurer **`Amazon VPC CNI`** :
   a. Sélectionnez une version `v1.14.0-eksbuild.3` ou ultérieure dans la liste Version.
   b. Sélectionnez Paramètres de configuration facultatifs.
   c. Saisissez la clé JSON `"enableNetworkPolicy":` et valeur `"true"` dans Valeurs de configuration. Le texte obtenu doit être un objet JSON valide. Si cette clé et cette valeur sont les seules données de la zone de texte, entourez-les d'accolades `{ }`. L'exemple suivant montre que la politique réseau est activée :

```
{ "enableNetworkPolicy": "true" }
```

La capture d'écran suivante montre un exemple de ce scénario.


EKS > Clusters > > Add-on > vpc-cni > Edit add-on
Configure Amazon VPC CNI
Amazon VPC CNI Info
Listed by
Category
networking
Status
Active
Version
Select the version for this add-on.
v1.17.1-eksbuild.1
Select IAM role
Select an IAM role to use with this add-on. To create a new role, follow the instructions in the Amazon EKS User Guide.
Optional configuration settings
Add-on configuration schema
Refer to the JSON schema below. The configuration values entered in the code editor will be validated against this schema.
{
"$ref": "#/definitions/VpcCni",
"$schema": "http://json-schema.org/draft-06/schema#",
"definitions": {
"Affinity": {
"type": [
"object",
"null"
]
},
"EniConfig": {
"additionalProperties": false,
Configuration values Info
Specify any additional JSON or YAML configurations that should be applied to the add-on.
1 { "enableNetworkPolicy": "true" }

Module complémentaire autogéré

Helm

Si vous avez installé le plug-in Amazon VPC CNI pour Kubernetes via `helm`, vous pouvez mettre à jour la configuration afin d'activer la politique réseau.

1. Exécutez la commande suivante pour activer la stratégie réseau.

```
helm upgrade --set enableNetworkPolicy=true aws-vpc-cni --namespace kube-system eks/
aws-vpc-cni
```

kubectl

1. Ouvrez le `amazon-vpc-cni` `ConfigMap` dans votre éditeur.

```
kubectl edit configmap -n kube-system amazon-vpc-cni -o yaml
```

2. Ajoutez la ligne suivante dans les `data` de `ConfigMap`.

```
enable-network-policy-controller: "true"
```

   Une fois la ligne ajoutée, votre `ConfigMap` doit ressembler à l'exemple ci-dessous.

```
apiVersion: v1
 kind: ConfigMap
 metadata:
  name: amazon-vpc-cni
  namespace: kube-system
 data:
  enable-network-policy-controller: "true"
```

3. Ouvrez le `aws-node` `DaemonSet` dans votre éditeur.

```
kubectl edit daemonset -n kube-system aws-node
```

   a. Remplacez `false` par `true` dans l'argument de commande `--enable-network-policy=false` pour le `args:` du conteneur `aws-network-policy-agent` dans le manifeste daemonset du `aws-node` VPC CNI.

```
    - args:
       - --enable-network-policy=true
```

Étape 4 : Étapes suivantes

Après avoir terminé la configuration, vérifiez que les pods `aws-node` sont en cours d'exécution dans votre cluster.

```
kubectl get pods -n kube-system | grep 'aws-node\|amazon'
```

L'exemple qui suit illustre un résultat.

```
aws-node-gmqp7                                          2/2     Running   1 (24h ago)
 24h
aws-node-prnsh                                          2/2     Running   1 (24h ago)
 24h
```

Les pods `aws-node` comportent deux conteneurs à partir des versions `1.14` et ultérieures. Dans les versions précédentes et si la stratégie réseau est désactivée, il n'y a qu'un seul conteneur dans les pods du `aws-node`.

Vous pouvez maintenant déployer des politiques réseau Kubernetes dans votre cluster.

Pour implémenter les politiques réseau Kubernetes, vous pouvez créer des Kubernetes `NetworkPolicy` ou des `ClusterNetworkPolicy` objets et les déployer sur votre cluster. `NetworkPolicy`les objets sont limités à un espace de noms, tandis que les `ClusterNetworkPolicy` objets peuvent être étendus à l'ensemble du cluster ou à plusieurs espaces de noms. Vous mettez en œuvre des politiques pour autoriser ou refuser le trafic entre les pods en fonction des sélecteurs d'étiquettes, des espaces de noms et des plages d'adresses IP. Pour plus d'informations sur la création des objets `NetworkPolicy`, consultez Politiques réseau dans la documentation Kubernetes.

L'application des objets `NetworkPolicy` Kubernetes repose sur l'Extended Berkeley Packet Filter (eBPF). Concernant les implémentations basées sur les `iptables`, il offre des caractéristiques de latence et de performance plus faibles, notamment une utilisation réduite du processeur et l'évitement des recherches séquentielles. De plus, les sondes eBPF fournissent des données contextuelles détaillées permettant de déboguer les problèmes complexes au niveau du noyau et d'améliorer

l'observabilité. Amazon EKS prend en charge un exportateur basé sur eBPF qui exploite ces sondes pour enregistrer les résultats des politiques sur chaque nœud et exporter les données vers des collecteurs de journaux externes afin de faciliter le débogage. Pour plus d'informations, consultez documentation eBPF.

Désactiver les politiques réseau Kubernetes pour le trafic réseau des pods Amazon EKS

Désactiver les politiques réseau Kubernetes pour ne plus restreindre le trafic réseau des pods Amazon EKS

1. Répertoriez toutes les politiques réseau Kubernetes.

```
kubectl get netpol -A
```

2. Supprimez chaque politique réseau Kubernetes. Vous devez supprimer toutes les politiques réseau avant de les désactiver.

```
kubectl delete netpol <policy-name>
```

3. Ouvrez le DaemonSet aws-node dans votre éditeur.

```
kubectl edit daemonset -n kube-system aws-node
```

4. Remplacez `true` par `false` dans l'argument de commande `--enable-network-policy=true` pour le `args:` du conteneur `aws-network-policy-agent` dans le manifeste daemonset du `aws-node` VPC CNI.

```
    - args:
       - --enable-network-policy=true
```

Dépannage des politiques réseau Kubernetes pour Amazon EKS

Ce guide de dépannage concerne la fonctionnalité de politique réseau d'Amazon VPC CNI.

Ce guide couvre les points suivants :

- Informations d'installation, autorisations CRD et RBAC the section called "Nouveaux CRD et autorisations `policyendpoints`"
- Journaliser les journaux lors du diagnostic des problèmes de politique réseau the section called "Journaux de stratégie réseau"

- Exécution de la collection d'outils du kit SDK eBPF pour le dépannage
- Problèmes connus et solutions the section called "Problèmes connus et solutions"

> **Note**
>
> Veuillez noter que les politiques réseau ne s'appliquent qu'aux pods créés par les déploiements Kubernetes. Pour plus d'informations sur les limitations des politiques réseau dans le CNI VPC, consultez the section called "Considérations".

Vous pouvez dépanner et analyser les connexions réseau qui utilisent des politiques réseau en lisant les journaux des politiques réseau et en exécutant les outils du kit SDK eBPF.

Nouveaux CRD et autorisations **`policyendpoints`**

- CRD : `policyendpoints.networking.k8s.aws`
- API Kubernetes : `apiservice` appelé `v1.networking.k8s.io`
- Ressource Kubernetes : `Kind: NetworkPolicy`
- RBAC : `ClusterRole` appelé `aws-node` (CNI VPC), `ClusterRole` appelé `eks:network-policy-controller` (contrôleur de politique réseau dans le plan de contrôle du cluster EKS)

Pour la politique réseau, le CNI VPC crée une nouvelle `CustomResourceDefinition` (CRD) appelée `policyendpoints.networking.k8s.aws`. Le CNI du VPC doit être autorisé à créer le CRD et à créer CustomResources (CR) celui-ci et l'autre CRD installé par le VPC CNI (). `eniconfigs.crd.k8s.amazonaws.com` Les deux CRDs sont disponibles dans le `crds.yaml`fichier sur GitHub. Plus précisément, le CNI VPC doit disposer des autorisations « get », « list » et « watch » pour `policyendpoints`.

La politique réseau Kubernetes fait partie du `apiservice` appelé `v1.networking.k8s.io`, et il s'agit de `apiversion: networking.k8s.io/v1` dans vos fichiers YAML de politique. Le `DaemonSet` CNI VPC doit disposer des autorisations nécessaires pour utiliser cette partie de l'API Kubernetes.

Les autorisations CNI VPC se trouvent dans un `ClusterRole` appelé `aws-node`. Veuillez noter que les objets `ClusterRole` ne sont pas regroupés dans des espaces de noms. Voici l'`aws-node` d'un cluster :

```
kubectl get clusterrole aws-node -o yaml
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  labels:
    app.kubernetes.io/instance: aws-vpc-cni
    app.kubernetes.io/managed-by: Helm
    app.kubernetes.io/name: aws-node
    app.kubernetes.io/version: v1.19.4
    helm.sh/chart: aws-vpc-cni-1.19.4
    k8s-app: aws-node
  name: aws-node
rules:
- apiGroups:
  - crd.k8s.amazonaws.com
  resources:
  - eniconfigs
  verbs:
  - list
  - watch
  - get
- apiGroups:
  - ""
  resources:
  - namespaces
  verbs:
  - list
  - watch
  - get
- apiGroups:
  - ""
  resources:
  - pods
  verbs:
  - list
  - watch
  - get
- apiGroups:
  - ""
  resources:
  - nodes
  verbs:
```

```
  - list
  - watch
  - get
- apiGroups:
  - ""
  - events.k8s.io
  resources:
  - events
  verbs:
  - create
  - patch
  - list
- apiGroups:
  - networking.k8s.aws
  resources:
  - policyendpoints
  verbs:
  - get
  - list
  - watch
- apiGroups:
  - networking.k8s.aws
  resources:
  - policyendpoints/status
  verbs:
  - get
- apiGroups:
  - vpcresources.k8s.aws
  resources:
  - cninodes
  verbs:
  - get
  - list
  - watch
  - patch
```

De plus, un nouveau contrôleur s'exécute dans le plan de contrôle de chaque cluster EKS. Le contrôleur utilise les autorisations du `ClusterRole` appelé `eks:network-policy-controller`. Voici l'`eks:network-policy-controller` d'un cluster :

```
kubectl get clusterrole eks:network-policy-controller -o yaml
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  labels:
    app.kubernetes.io/name: amazon-network-policy-controller-k8s
  name: eks:network-policy-controller
rules:
- apiGroups:
  - ""
  resources:
  - namespaces
  verbs:
  - get
  - list
  - watch
- apiGroups:
  - ""
  resources:
  - pods
  verbs:
  - get
  - list
  - watch
- apiGroups:
  - ""
  resources:
  - services
  verbs:
  - get
  - list
  - watch
- apiGroups:
  - networking.k8s.aws
  resources:
  - policyendpoints
  verbs:
  - create
  - delete
  - get
  - list
  - patch
  - update
  - watch
```

```
- apiGroups:
  - networking.k8s.aws
  resources:
  - policyendpoints/finalizers
  verbs:
  - update
- apiGroups:
  - networking.k8s.aws
  resources:
  - policyendpoints/status
  verbs:
  - get
  - patch
  - update
- apiGroups:
  - networking.k8s.io
  resources:
  - networkpolicies
  verbs:
  - get
  - list
  - patch
  - update
  - watch
```

Journaux de stratégie réseau

Chaque décision prise par le VPC CNI concernant l'autorisation ou le refus des connexions par les politiques réseau est journalisée dans les journaux de flux. Les journaux de stratégie réseau de chaque nœud incluent les journaux de flux pour chaque pod doté d'une stratégie réseau. Les journaux de stratégie réseau sont stockés sur `/var/log/aws-routed-eni/network-policy-agent.log`. L'exemple suivant est extrait d'un fichier `network-policy-agent.log` :

```
{"level":"info","timestamp":"2023-05-30T16:05:32.573Z","logger":"ebpf-
client","msg":"Flow Info: ","Src
IP":"192.168.87.155","Src Port":38971,"Dest IP":"64.6.160","Dest
Port":53,"Proto":"UDP","Verdict":"ACCEPT"}
```

Les journaux de politique réseau sont désactivés par défaut. Pour activer les journaux de stratégie réseau, procédez comme suit :

> **Note**
>
> Les journaux de politique réseau nécessitent 1 vCPU supplémentaire pour le conteneur `aws-network-policy-agent` dans le manifeste `aws-node DaemonSet` du CNI Amazon VPC.

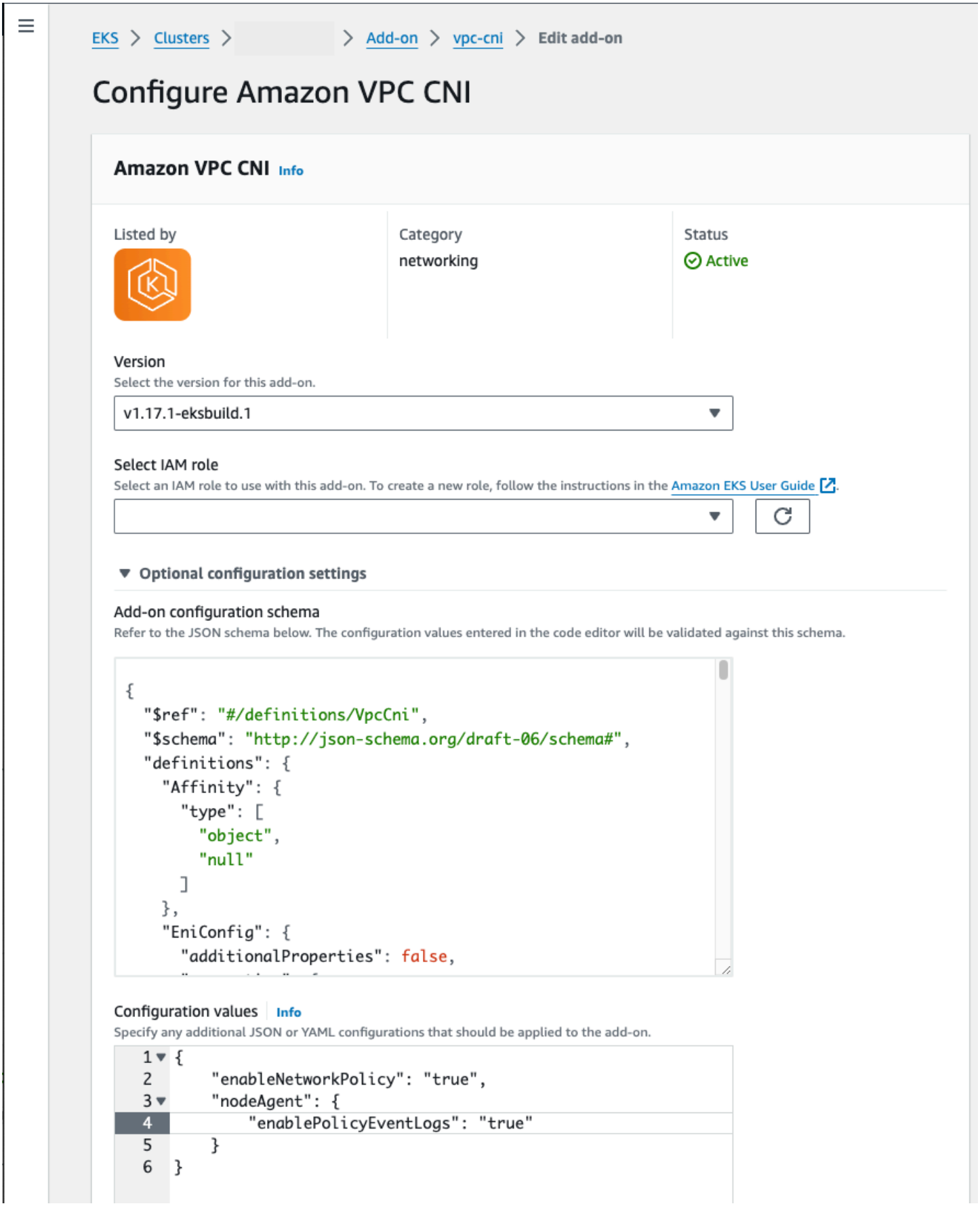
Module complémentaire d'Amazon EKS

AWS Management Console

a. Ouvrez la console Amazon EKS.
b. Dans le panneau de navigation de gauche, sélectionnez Clusters, puis sélectionnez le nom du cluster pour lequel vous souhaitez configurer le module complémentaire Amazon VPC CNI.
c. Choisissez l'onglet Modules complémentaires.
d. Cochez la case en haut à droite de la zone du module complémentaire, puis sélectionnez Edit (Modifier).
e. Sur la page Configurer ***Amazon VPC CNI*** :
   i. Sélectionnez une version `v1.14.0-eksbuild.3` ou ultérieure dans la liste déroulante Version.
   ii. Sélectionnez Paramètres de configuration facultatifs.
   iii. Saisissez la clé JSON de niveau supérieur de l'`"nodeAgent":` et la valeur est un objet avec une clé `"enablePolicyEventLogs":` et une valeur de `"true"` dansValeurs de configuration. Le texte obtenu doit être un objet JSON valide. L'exemple suivant montre la politique réseau et les journaux de stratégie réseau sont activés, et les journaux de stratégie réseau sont envoyés à CloudWatch Logs :

```
{
    "enableNetworkPolicy": "true",
    "nodeAgent": {
        "enablePolicyEventLogs": "true"
    }
}
```

La capture d'écran suivante montre un exemple de ce scénario.


EKS > Clusters > > Add-on > vpc-cni > Edit add-on
Configure Amazon VPC CNI
Amazon VPC CNI Info
Listed by
Category
networking
Status
Active
Version
Select the version for this add-on.
v1.17.1-eksbuild.1
Select IAM role
Select an IAM role to use with this add-on. To create a new role, follow the instructions in the Amazon EKS User Guide.
Optional configuration settings
Add-on configuration schema
Refer to the JSON schema below. The configuration values entered in the code editor will be validated against this schema.
{
"$ref": "#/definitions/VpcCni",
"$schema": "http://json-schema.org/draft-06/schema#",
"definitions": {
"Affinity": {
"type": [
"object",
"null"
]
},
"EniConfig": {
"additionalProperties": false,
Configuration values Info
Specify any additional JSON or YAML configurations that should be applied to the add-on.
1 {
2 "enableNetworkPolicy": "true",
3 "nodeAgent": {
4 "enablePolicyEventLogs": "true"
5 }
6 }

AWS CLI

a. Exécutez la commande AWS CLI suivante. Remplacez `my-cluster` par le nom de votre cluster et l'ARN du rôle IAM par le rôle que vous utilisez.

```
aws eks update-addon --cluster-name my-cluster --addon-name vpc-cni --addon-
version v1.14.0-eksbuild.3 \
    --service-account-role-arn arn:aws: iam::123456789012:role/AmazonEKSVPCCNIRole
 \
    --resolve-conflicts PRESERVE --configuration-values '{"nodeAgent":
 {"enablePolicyEventLogs": "true"}}'
```

Module complémentaire autogéré

Helm

Si vous avez installé le plug-in CNI Amazon VPC pour Kubernetes via `helm`, vous pouvez mettre à jour la configuration pour écrire les journaux de politique réseau.

a. Exécutez la commande suivante pour activer la stratégie réseau.

```
helm upgrade --set nodeAgent.enablePolicyEventLogs=true aws-vpc-cni --namespace
 kube-system eks/aws-vpc-cni
```

kubectl

Si vous avez installé le plug-in CNI Amazon VPC pour Kubernetes via `kubectl`, vous pouvez mettre à jour la configuration pour écrire les journaux de politique réseau.

a. Ouvrez le `aws-node DaemonSet` dans votre éditeur.

```
kubectl edit daemonset -n kube-system aws-node
```

b. Remplacez `false` par `true` dans l'argument de commande `--enable-policy-event-logs=false` pour le `args:` du conteneur `aws-network-policy-agent` dans le manifeste CNI VPC `aws-node DaemonSet`.

```
    - args:
       - --enable-policy-event-logs=true
```

Envoyer des journaux de politique réseau à Amazon CloudWatch Logs

Vous pouvez surveiller les journaux de politique réseau à l'aide de services tels qu'Amazon CloudWatch Logs. Vous pouvez utiliser les méthodes suivantes pour envoyer les journaux de politique réseau à CloudWatch Logs.

Pour les clusters EKS, les journaux de politique seront situés sous `/aws/eks/`*`cluster-name`*`/cluster/` et pour les clusters K8S autogérés, les journaux seront placés sous `/aws/k8s-cluster/cluster/`.

Envoyer les journaux de politique réseau avec le plug-in CNI Amazon VPC pour Kubernetes

Si vous activez la stratégie réseau, un deuxième conteneur est ajouté aux pods du `aws-node` pour unagent de nœud. Cet agent de nœud peut envoyer les journaux de politique réseau à CloudWatch Logs.

> **Note**
>
> Seuls les journaux de stratégie réseau sont envoyés par l'agent de nœud. Les autres journaux créés par le CNI Amazon VPC ne sont pas inclus.

Conditions préalables

- Ajoutez les autorisations suivantes sous forme de strophe ou de stratégie distincte au rôle IAM que vous utilisez pour le VPC CNI.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups",
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": "*"
        }
```

```
    ]
}
```

Module complémentaire d'Amazon EKS

AWS Management Console

a. Ouvrez la console Amazon EKS.

b. Dans le panneau de navigation de gauche, sélectionnez Clusters, puis sélectionnez le nom du cluster pour lequel vous souhaitez configurer le module complémentaire Amazon VPC CNI.

c. Choisissez l'onglet Modules complémentaires.

d. Cochez la case en haut à droite de la zone du module complémentaire, puis sélectionnez Edit (Modifier).

e. Sur la page Configurer ***Amazon VPC CNI*** :

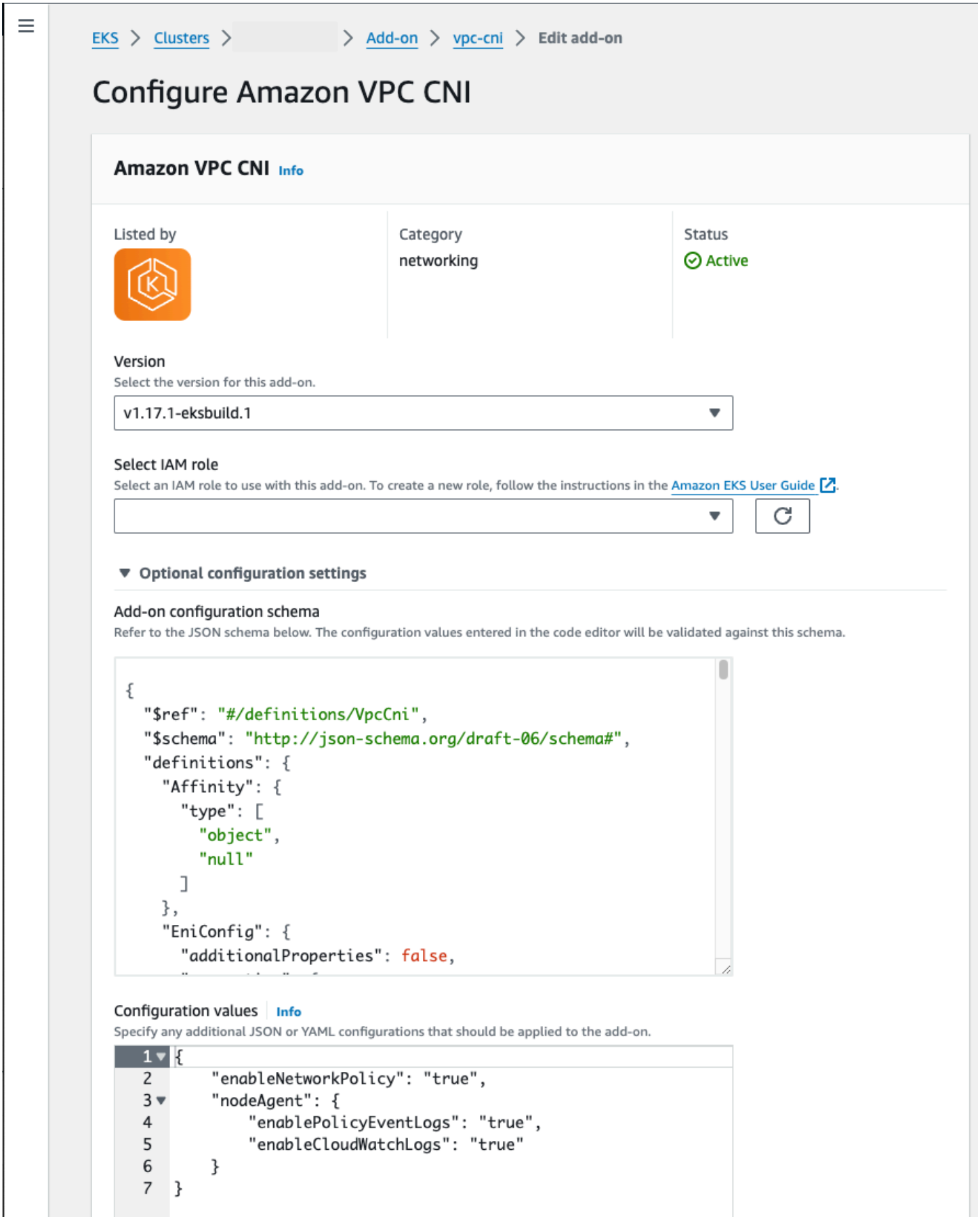
   i. Sélectionnez une version `v1.14.0-eksbuild.3` ou ultérieure dans la liste déroulante Version.

   ii. Sélectionnez Paramètres de configuration facultatifs.

   iii. Saisissez la clé JSON de niveau supérieur de l'`"nodeAgent":` et la valeur est un objet avec une clé `"enableCloudWatchLogs":` et une valeur de `"true"` dansValeurs de configuration. Le texte obtenu doit être un objet JSON valide. L'exemple suivant montre la politique réseau, les journaux de stratégie réseau sont activés et les journaux sont envoyés à CloudWatch Logs :

```
{
    "enableNetworkPolicy": "true",
    "nodeAgent": {
        "enablePolicyEventLogs": "true",
        "enableCloudWatchLogs": "true",
    }
}
```

La capture d'écran suivante montre un exemple de ce scénario.


EKS > Clusters > > Add-on > vpc-cni > Edit add-on
Configure Amazon VPC CNI
Amazon VPC CNI Info
Listed by
Category
networking
Status
Active
Version
Select the version for this add-on.
v1.17.1-eksbuild.1
Select IAM role
Select an IAM role to use with this add-on. To create a new role, follow the instructions in the Amazon EKS User Guide.
Optional configuration settings
Add-on configuration schema
Refer to the JSON schema below. The configuration values entered in the code editor will be validated against this schema.
{
"$ref": "#/definitions/VpcCni",
"$schema": "http://json-schema.org/draft-06/schema#",
"definitions": {
"Affinity": {
"type": [
"object",
"null"
]
},
"EniConfig": {
"additionalProperties": false,
Configuration values Info
Specify any additional JSON or YAML configurations that should be applied to the add-on.
1 {
2 "enableNetworkPolicy": "true",
3 "nodeAgent": {
4 "enablePolicyEventLogs": "true",
5 "enableCloudWatchLogs": "true"
6 }
7 }

AWS CLI

a. Exécutez la commande AWS CLI suivante. Remplacez `my-cluster` par le nom de votre cluster et l'ARN du rôle IAM par le rôle que vous utilisez.

```
aws eks update-addon --cluster-name my-cluster --addon-name vpc-cni --addon-
version v1.14.0-eksbuild.3 \
    --service-account-role-arn arn:aws: iam::123456789012:role/AmazonEKSVPCCNIRole
 \
    --resolve-conflicts PRESERVE --configuration-values '{"nodeAgent":
 {"enablePolicyEventLogs": "true", "enableCloudWatchLogs": "true"}}'
```

Module complémentaire autogéré

Casque

Si vous avez installé le plug-in Amazon VPC CNI pour Kubernetes via`helm`, vous pouvez mettre à jour la configuration pour envoyer les journaux de politique réseau à Logs. CloudWatch

a. Exécutez la commande suivante pour activer les journaux de politique réseau et les envoyer à CloudWatch Logs.

```
helm upgrade --set nodeAgent.enablePolicyEventLogs=true --set
 nodeAgent.enableCloudWatchLogs=true aws-vpc-cni --namespace kube-system eks/aws-
vpc-cni
```

kubectl

a. Ouvrez le `aws-node` `DaemonSet` dans votre éditeur.

```
kubectl edit daemonset -n kube-system aws-node
```

b. Remplacez `false` par `true` dans les deux arguments de commande `--enable-policy-event-logs=false` et `--enable-cloudwatch-logs=false` dans `args:` dans le conteneur `aws-network-policy-agent` du manifeste `aws-node` `DaemonSet` de VPC CNI.

```
    - args:
       - --enable-policy-event-logs=true
       - --enable-cloudwatch-logs=true
```

Envoyer les journaux de politique réseau avec un **`DaemonSet`** Fluent Bit

Si vous utilisez Fluent Bit dans un `DaemonSet` pour envoyer des journaux depuis vos nœuds, vous pouvez ajouter une configuration pour inclure les journaux de politique réseau provenant des politiques réseau. Vous pouvez utiliser l'exemple de configuration suivant :

```
    [INPUT]
        Name              tail
        Tag               eksnp.*
        Path              /var/log/aws-routed-eni/network-policy-agent*.log
        Parser            json
        DB                /var/log/aws-routed-eni/flb_npagent.db
        Mem_Buf_Limit     5MB
        Skip_Long_Lines   On
        Refresh_Interval  10
```

Kit SDK eBPF inclus

Le plug-in CNI Amazon VPC pour Kubernetes installe la collection d'outils du kit SDK eBPF sur les nœuds. Vous pouvez utiliser les outils du kit SDK eBPF pour identifier les problèmes liés aux politiques réseau. Par exemple, la commande suivante répertorie les programmes qui s'exécutent sur le nœud.

```
sudo /opt/cni/bin/aws-eks-na-cli ebpf progs
```

Pour exécuter cette commande, vous pouvez utiliser n'importe quelle méthode pour vous connecter au nœud.

Problèmes connus et solutions

Les sections suivantes décrivent les problèmes connus liés à la fonctionnalité de politique réseau du CNI Amazon VPC et leurs solutions.

Journaux de politique réseau générés même si la valeur enable-policy-event-logs est définie sur false

Problème : le CNI VPC EKS génère des journaux de politique réseau même lorsque le paramètre `enable-policy-event-logs` est défini sur `false`.

Solution : le paramètre `enable-policy-event-logs` désactive uniquement les journaux de « décision » de politique, mais il ne désactive pas tous les journaux de l'agent de politique réseau. Ce comportement est documenté dans le aws-network-policy-agent fichier README sur GitHub. Pour

désactiver complètement la journalisation, vous devrez peut-être ajuster d'autres configurations de journalisation.

Problèmes liés au nettoyage du mappage des politiques réseau

Problème : des problèmes liés au `policyendpoint` réseau persistent et ne sont pas résolus après la suppression des pods.

Solution : ce problème était dû à un dysfonctionnement du module complémentaire CNI Amazon VPC version 1.19.3-eksbuild.1. Veuillez mettre à jour le module complémentaire CNI Amazon VPC vers une version plus récente pour résoudre ce problème.

Les politiques réseau ne sont pas appliquées

Problème : la fonctionnalité de politique réseau est activée dans le module complémentaire CNI Amazon VPC, mais les politiques réseau ne sont pas appliquées correctement.

Si vous créez une politique réseau `kind: NetworkPolicy` et qu'elle n'a aucun effet sur le pod, vérifiez que l'objet policyendpoint a été créé dans le même espace de noms que le pod. S'il n'y a pas d'objets `policyendpoint` dans les espaces de noms, le contrôleur de politique réseau (qui fait partie du cluster EKS) n'a pas pu créer de règles de politique réseau à appliquer par l'agent de politique réseau (qui fait partie du VPC CNI).

Solution : la solution consiste à corriger les autorisations du CNI VPC (`ClusterRole` : `aws-node`) et du contrôleur de politique réseau (`ClusterRole` : `eks:network-policy-controller`) et à autoriser ces actions dans tout outil d'application de politique tel que Kyverno. Assurez-vous que les politiques Kyverno ne bloquent pas la création d'objets `policyendpoint`. Consultez la section précédente pour connaître les autorisations nécessaires dans the section called "Nouveaux CRD et autorisations `policyendpoints`".

Les pods ne reviennent pas à l'état de refus par défaut après la suppression de la politique en mode strict

Problème : lorsque les politiques réseau sont activées en mode strict, les pods démarrent avec une politique de refus par défaut. Une fois les politiques appliquées, le trafic est autorisé vers les points de terminaison spécifiés. Cependant, lorsque les politiques sont supprimées, le pod ne revient pas à l'état de refus par défaut, mais passe à un état d'autorisation par défaut.

Solution : ce problème a été corrigé dans la version 1.19.3 du CNI VPC, qui incluait la version 1.2.0 de l'agent de politique réseau. Après la correction, lorsque le mode strict est activé, une fois les politiques supprimées, le pod revient à l'état de refus par défaut comme prévu.

Latence de démarrage des groupes de sécurité pour les pods

Problème : lorsque vous utilisez la fonctionnalité Groupes de sécurité pour les pods dans EKS, la latence de démarrage des pods augmente.

Solution : La latence est due à la limitation du débit dans le contrôleur de ressources due à la limitation de l'`CreateNetworkInterface`API, que le contrôleur de ressources VPC utilise pour créer une branche ENIs pour les pods. Veuillez vérifier les limites API de votre compte pour cette opération et envisager de demander une augmentation de la limite si nécessaire.

FailedScheduling en raison d'une insuffisance de vpc.amazonaws.com/pod-eni

Problème : les pods ne parviennent pas à se planifier et l'erreur suivante s'affiche : `FailedScheduling 2m53s (x28 over 137m) default-scheduler 0/5 nodes are available: 5 Insufficient vpc.amazonaws.com/pod-eni. preemption: 0/5 nodes are available: 5 No preemption victims found for incoming pod.`

Solution : comme pour le problème précédent, l'attribution de groupes de sécurité aux pods augmente la latence de planification des pods et peut dépasser le seuil CNI pour le temps nécessaire à l'ajout de chaque ENI, ce qui entraîne des échecs au démarrage des pods. Il s'agit d'un comportement normal lorsque vous utilisez des groupes de sécurité pour les pods. Tenez compte des implications en matière de planification lorsque vous concevez l'architecture de votre charge de travail.

Problèmes de connectivité IPAM et erreurs de segmentation

Problème : plusieurs erreurs se produisent, notamment des problèmes de connectivité IPAM, des demandes de limitation et des erreurs de segmentation :

- `Checking for IPAM connectivity …`
- `Throttling request took 1.047064274s`
- `Retrying waiting for IPAM-D`
- `panic: runtime error: invalid memory address or nil pointer dereference`

Solution : Ce problème se produit si vous installez `systemd-udev` le fichier AL2 023, car le fichier est réécrit avec une politique d'interruption. Cela peut se produire lors de la mise à jour vers une autre `releasever` qui dispose d'un package mis à jour ou lors de la mise à jour manuelle du package lui-même. Évitez d'installer ou de mettre à jour `systemd-udev` sur les nœuds AL2 023.

Erreur : impossible de trouver le dispositif par son nom

Problème : message d'erreur :
`{"level":"error","ts":"2025-02-05T20:27:18.669Z","caller":"ebpf/bpf_client.go:578","msg":"failed to find device by name eni9ea69618bf0: %!w(netlink.LinkNotFoundError={0xc000115310})"}`

Solution : les dernières versions de l'agent de politique réseau CNI Amazon VPC (v1.2.0) identifient et corrigent ce problème. Veuillez mettre à jour vers la dernière version du CNI VPC pour résoudre ce problème.

Vulnérabilités CVE dans l'image CNI Multus

Problème : le rapport EKS ImageScan CVE amélioré identifie des vulnérabilités dans la version v4.1.4-eksbuild.2_thick de l'image Multus CNI.

Solution : veuillez mettre à jour vers la nouvelle version de l'image CNI Multus et la nouvelle image du contrôleur de politique réseau, qui ne présentent aucune vulnérabilité. Vous pouvez mettre à jour l'analyseur pour corriger les vulnérabilités détectées dans la version précédente.

Verdicts DENY des informations de flux dans les journaux

Problème : les journaux de politique réseau affichent des verdicts DENY :
`{"level":"info","ts":"2024-11-25T13:34:24.808Z","logger":"ebpf-client","caller":"events/events.go:193","msg":"Flow Info: ","Src IP":"","Src Port":9096,"Dest IP":"","Dest Port":56830,"Proto":"TCP","Verdict":"DENY"}`

Solution : le problème a été résolu dans la nouvelle version du contrôleur de politique réseau. Veuillez mettre à jour vers la dernière version de plateforme EKS pour résoudre les problèmes de journalisation.

Pod-to-pod problèmes de communication après la migration depuis Calico

Problème : Après la mise à niveau d'un cluster EKS vers la version 1.30 et le passage de Calico à Amazon VPC CNI pour la politique réseau, la pod-to-pod communication échoue lorsque les politiques réseau sont appliquées. La communication est rétablie lorsque les politiques réseau sont supprimées.

Solution : l'agent de politique réseau dans le CNI VPC ne peut pas avoir autant de ports spécifiés que Calico. Utilisez plutôt des plages de ports dans les politiques réseau. Le nombre maximal

de combinaisons uniques de ports pour chaque protocole dans chaque sélecteur `ingress:` ou `egress:` d'une politique réseau est de 24. Utilisez des plages de ports pour réduire le nombre de ports uniques et éviter cette limitation.

L'agent de politique réseau ne prend pas en charge les pods autonomes

Problème : les politiques réseau appliquées à des pods autonomes peuvent avoir un comportement incohérent.

Solution : l'agent de politique réseau ne prend actuellement en charge que les pods déployés dans le cadre d'un déploiement/réplicaset. Si des politiques réseau sont appliquées à des pods autonomes, leur comportement peut présenter certaines incohérences. Ceci est documenté en haut de cette page, dans le the section called "Considérations" numéro et dans le aws-network-policy-agent GitHub numéro #327 sur GitHub. Déployez les pods dans le cadre d'un déploiement ou d'un réplicaset pour garantir un comportement cohérent des politiques réseau.

Démonstration des politiques réseau pour Amazon EKS

Cette démonstration permet de créer un service client, un service frontal et un service backend sur votre cluster Amazon EKS. La démonstration crée également une interface utilisateur graphique de gestion qui affiche les chemins d'entrée et de sortie disponibles entre chaque service. Nous vous recommandons d'effectuer la démonstration sur un cluster sur lequel vous n'exécutez pas de charges de travail de production.

Avant de créer des politiques réseau, tous les services peuvent communiquer de façon bidirectionnelle. Une fois que vous avez appliqué les politiques réseau, vous constatez que le client peut uniquement communiquer avec le service frontend et que le service backend n'accepte que le trafic du frontend.

1. Appliquez les services frontal, backend et d'interface utilisateur de gestion :

```
kubectl apply -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
policies/calico/stars_policy_demo/create_resources.files/namespace.yaml
kubectl apply -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
policies/calico/stars_policy_demo/create_resources.files/management-ui.yaml
kubectl apply -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
policies/calico/stars_policy_demo/create_resources.files/backend.yaml
```

```
kubectl apply -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
policies/calico/stars_policy_demo/create_resources.files/frontend.yaml
kubectl apply -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
policies/calico/stars_policy_demo/create_resources.files/client.yaml
```

2. Affichez tous les pods du cluster.

```
kubectl get pods -A
```

   L'exemple qui suit illustre un résultat.

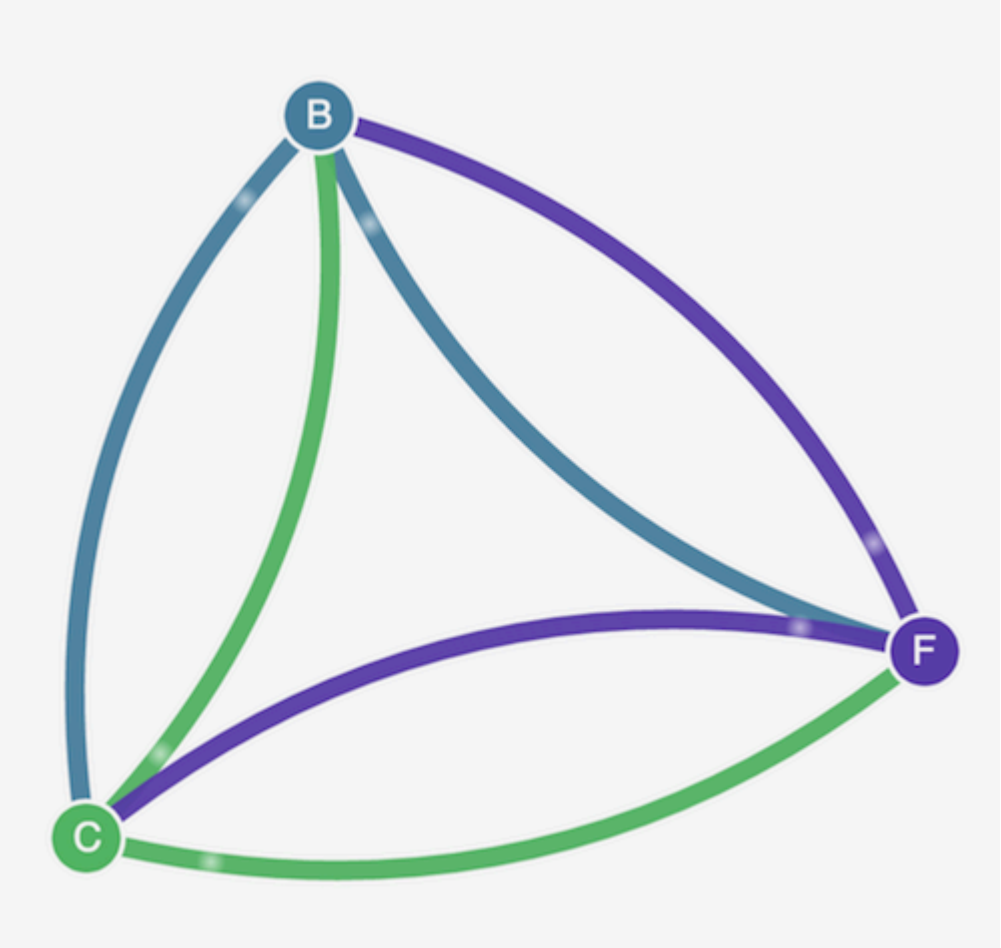
   Dans votre sortie, vous devriez voir des pods dans les espaces de noms affichés dans la sortie suivante. Les *NOMS* de vos pods et le nombre de pods dans la colonne READY sont différents de ceux de la sortie suivante. Ne continuez pas tant que vous ne voyez pas des pods avec des noms similaires et que tous ont la mention Running dans la colonne STATUS.

```
NAMESPACE         NAME                                       READY   STATUS
 RESTARTS   AGE
[...]
client            client-xlffc                               1/1     Running   0
     5m19s
[...]
management-ui     management-ui-qrb2g                        1/1     Running   0
     5m24s
stars             backend-sz87q                              1/1     Running   0
     5m23s
stars             frontend-cscnf                             1/1     Running   0
     5m21s
[...]
```

3. Pour vous connecter à l'interface utilisateur de gestion, connectez-vous à l'EXTERNAL-IP du service qui s'exécute sur votre cluster :

```
kubectl get service/management-ui -n management-ui
```

4. Ouvrez le navigateur à l'emplacement indiqué à l'étape précédente. Vous devez voir l'interface utilisateur de gestion. Le nœud C est le service client, le nœud F est le service frontal et le nœud B est le service backend. Chaque nœud a un accès en communication complet à tous les autres nœuds, comme indiqué par les lignes colorées en gras.



5. Appliquez les stratégies réseau suivantes tant dans les espaces de noms `stars` que `client` pour isoler les services les uns des autres :

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: default-deny
spec:
  podSelector:
    matchLabels: {}
```

Vous pouvez utiliser les commandes suivantes pour appliquer la stratégie aux deux espaces de noms :

```
kubectl apply -n stars -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
policies/calico/stars_policy_demo/apply_network_policies.files/default-deny.yaml
```

```
kubectl apply -n client -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
policies/calico/stars_policy_demo/apply_network_policies.files/default-deny.yaml
```

6. Actualisez votre navigateur. Vous constatez que l'interface utilisateur de gestion ne peut plus atteindre aucun des nœuds, qui n'apparaissent donc pas dans l'interface utilisateur.
7. Appliquez les stratégies réseau différentes suivantes pour permettre à l'interface utilisateur de gestion d'accéder aux services. Appliquez cette stratégie pour autoriser l'interface utilisateur :

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  namespace: stars
  name: allow-ui
spec:
  podSelector:
    matchLabels: {}
  ingress:
    - from:
        - namespaceSelector:
            matchLabels:
              role: management-ui
```

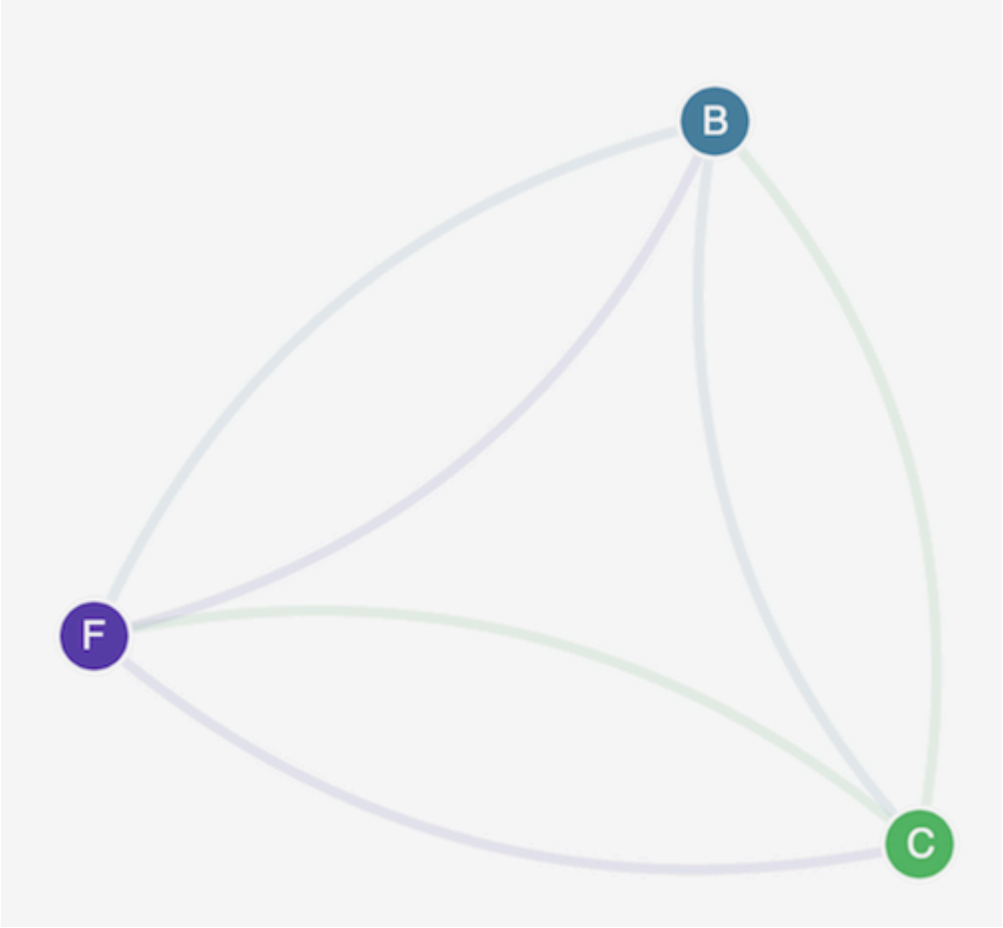
Appliquez cette stratégie pour autoriser le client :

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  namespace: client
  name: allow-ui
spec:
  podSelector:
    matchLabels: {}
  ingress:
    - from:
        - namespaceSelector:
            matchLabels:
              role: management-ui
```

Vous pouvez utiliser les commandes suivantes pour appliquer les deux stratégies :

```
kubectl apply -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
policies/calico/stars_policy_demo/apply_network_policies.files/allow-ui.yaml
kubectl apply -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
policies/calico/stars_policy_demo/apply_network_policies.files/allow-ui-client.yaml
```

8. Actualisez votre navigateur. Vous pouvez alors constater que l'interface utilisateur graphique de gestion peut à nouveau atteindre les nœuds, mais que les nœuds ne peuvent pas communiquer entre eux.

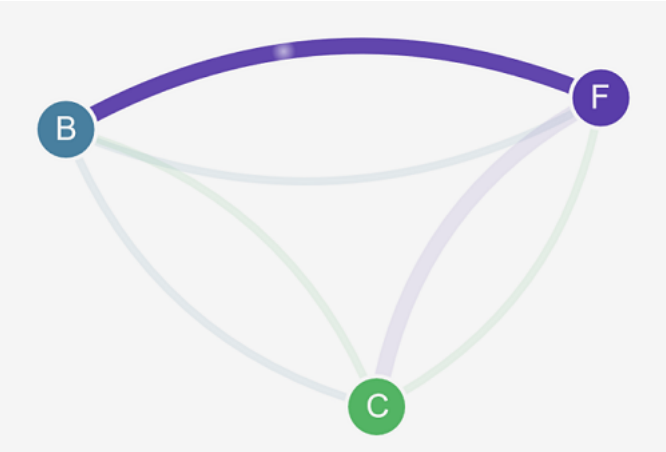


9. Appliquez la politique réseau suivante pour autoriser le trafic du service frontal au service backend :

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  namespace: stars
  name: backend-policy
```

```
spec:
  podSelector:
    matchLabels:
      role: backend
  ingress:
    - from:
        - podSelector:
            matchLabels:
              role: frontend
      ports:
        - protocol: TCP
          port: 6379
```

10 Actualisez votre navigateur. Vous voyez que le frontend peut communiquer avec le backend.

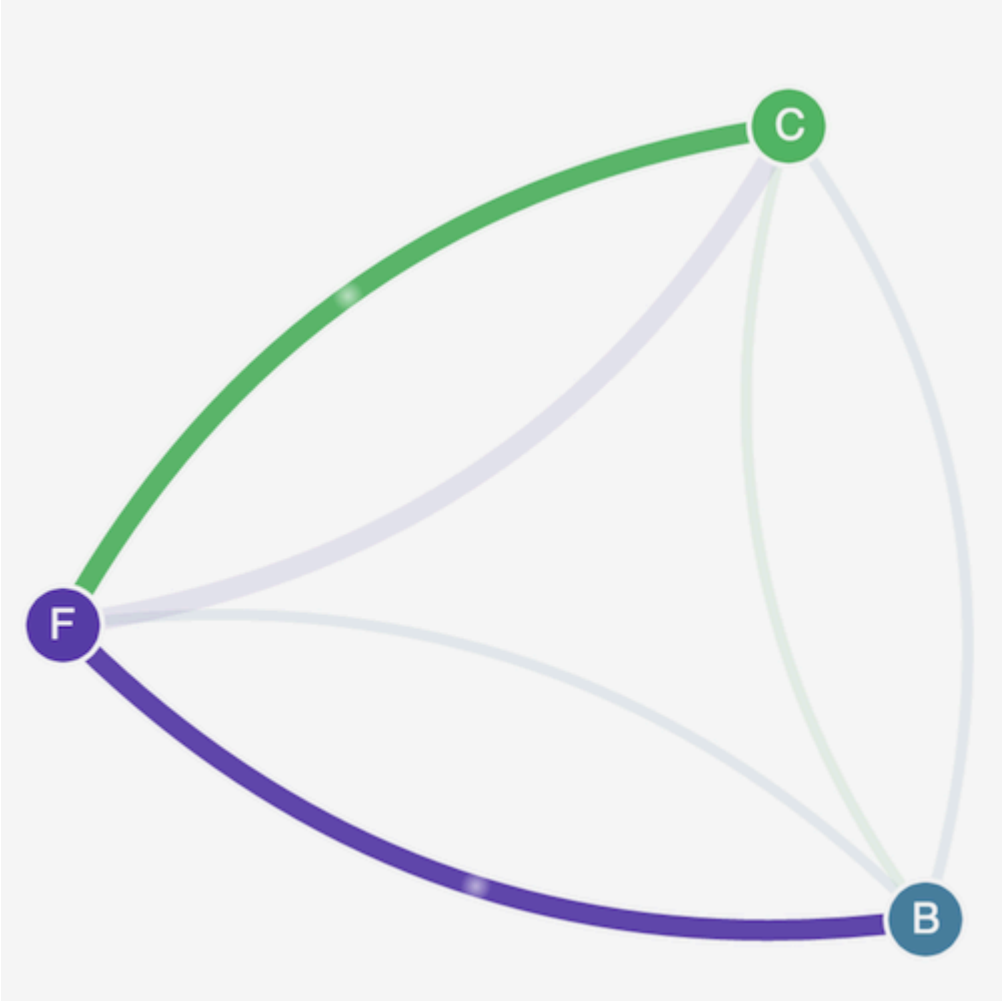


11 Appliquez la stratégie réseau suivante pour autoriser le trafic du client vers le service frontend :

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  namespace: stars
  name: frontend-policy
spec:
  podSelector:
    matchLabels:
      role: frontend
  ingress:
    - from:
        - namespaceSelector:
            matchLabels:
              role: client
      ports:
```

```
    - protocol: TCP
      port: 80
```

12. Actualisez votre navigateur. Vous voyez que le client peut communiquer avec le service frontend. Le service frontend peut toujours communiquer avec le service backend.



13. (Facultatif) Une fois que vous avez terminé avec la démonstration, vous pouvez supprimer ses ressources.

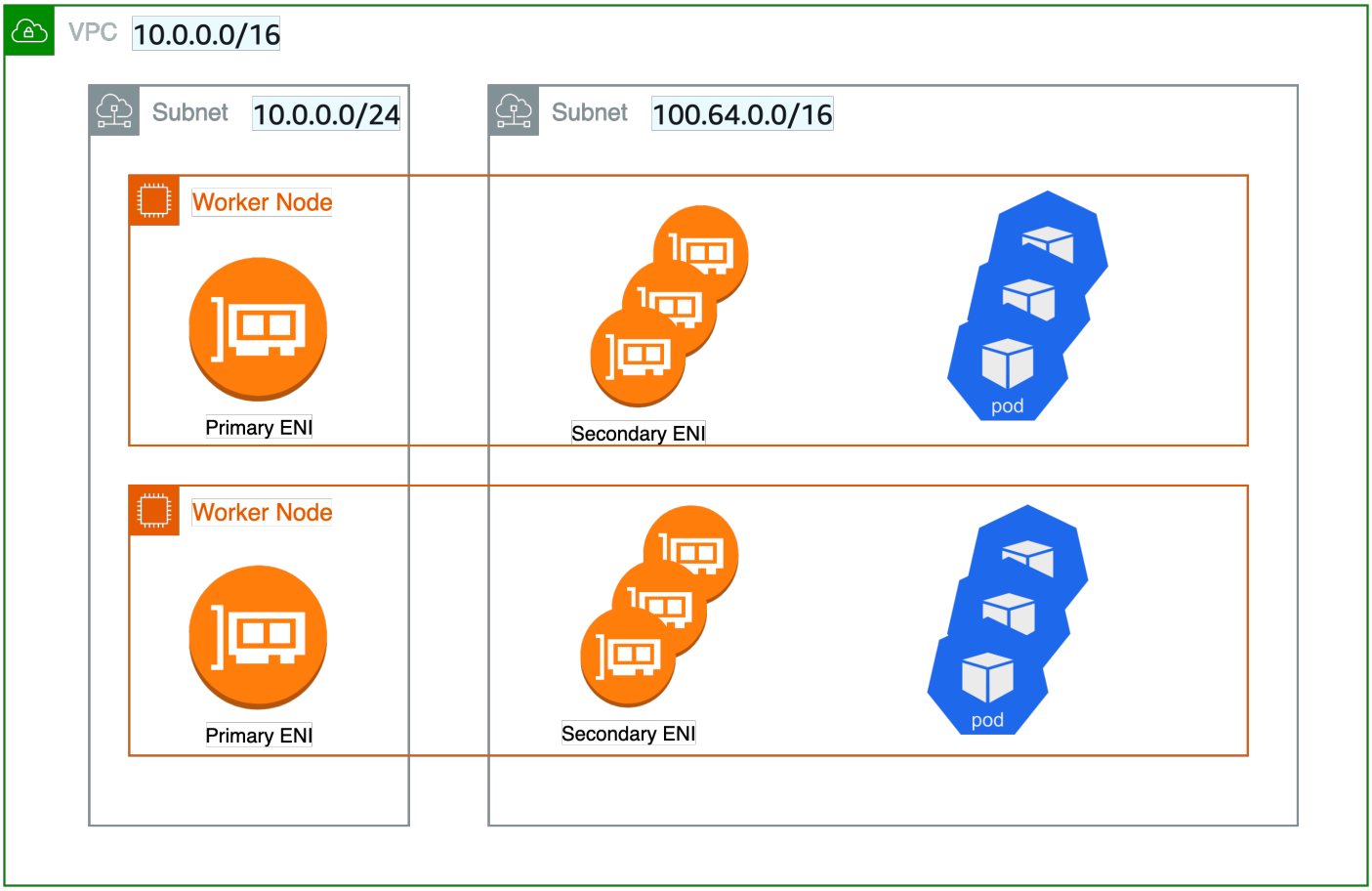
```
kubectl delete -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
policies/calico/stars_policy_demo/create_resources.files/client.yaml
kubectl delete -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
policies/calico/stars_policy_demo/create_resources.files/frontend.yaml
kubectl delete -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
policies/calico/stars_policy_demo/create_resources.files/backend.yaml
```

```
kubectl delete -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
policies/calico/stars_policy_demo/create_resources.files/management-ui.yaml
kubectl delete -f https://raw.githubusercontent.com/aws-samples/eks-
workshop/2f9d29ed3f82ed6b083649e975a0e574fb8a4058/content/beginner/120_network-
policies/calico/stars_policy_demo/create_resources.files/namespace.yaml
```

Même après la suppression des ressources, il peut toujours y avoir des points de terminaison de stratégie réseau sur les nœuds qui peuvent interférer de manière inattendue avec la mise en réseau dans votre cluster. La seule façon sûre de supprimer ces règles est de redémarrer les nœuds ou de terminer tous les nœuds et de les recycler. Pour mettre fin à tous les nœuds, définissez le nombre souhaité du groupe Auto Scaling sur 0, puis sauvegardez jusqu'au nombre souhaité, ou terminez simplement les nœuds.

Déploiement de pods dans des sous-réseaux alternatifs avec réseau personnalisé

S'applique à : nœuds Linux `IPv4` Fargate, nœuds Linux avec instances Amazon EC2



Par défaut, lorsque le plug-in Amazon VPC CNI pour Kubernetes crée des interfaces réseau Elastic (interfaces réseau) secondaires pour votre nœud Amazon EC2, il les crée dans le même sous-réseau que l'interface réseau principale du nœud. Il associe également les mêmes groupes de sécurité à l'interface réseau secondaire, qui sont associés à l'interface réseau principale. Pour une ou plusieurs des raisons suivantes, vous voudrez peut-être que le plugin crée des interfaces réseau secondaires dans un sous-réseau différent ou associe différents groupes de sécurité aux interfaces réseau secondaires, ou les deux :

- Le nombre d'adresses `IPv4` disponibles dans ce sous-réseau est limité, ce qui peut réduire le nombre de pods pouvant être créés dans ce sous-réseau. Cela peut limiter le nombre de pods que vous pouvez créer dans le sous-réseau. En utilisant un sous-réseau différent pour les interfaces réseau secondaires, vous pouvez augmenter le nombre d'adresses `IPv4` disponibles pour les pods.
- Pour des raisons de sécurité, vos pods doivent utiliser d'autres groupes de sécurité ou sous-réseaux que l'interface réseau principale du nœud.
- Les nœuds sont configurés dans des sous-réseaux publics et vous souhaitez placer les pods dans des sous-réseaux privés. La table de routage associée à un sous-réseau public comprend une route vers une passerelle Internet. La table de routage associée à un sous-réseau privé ne comprend pas une route vers une passerelle Internet.

> **Tip**
>
> Vous pouvez également ajouter un sous-réseau, nouveau ou existant, directement à votre cluster Amazon EKS sans utiliser le réseau personnalisé. Pour de plus amples informations, consultez the section called "Ajouter un sous-réseau VPC existant à un cluster Amazon EKS à partir de la console de gestion".

Considérations

Tenez compte des points suivants pour utiliser cette fonctionnalité.

- Avec le réseau personnalisé activé, aucune adresse IP attribuée à l'interface réseau principale n'est attribuée aux pods. Seules les adresses IP provenant des interfaces réseau secondaires sont attribuées aux pods.
- Si votre cluster utilise la famille `IPv6`, vous ne pouvez pas utiliser le réseau personnalisé.

- Si vous prévoyez d'utiliser une mise en réseau personnalisée uniquement pour pallier à l'épuisement des adresses `IPv4`, vous pouvez plutôt créer un cluster à l'aide de la famille `IPv6`. Pour de plus amples informations, consultez the section called "IPv6".
- Même si les pods déployés dans des sous-réseaux définis pour les interfaces réseau secondaires peuvent utiliser un sous-réseau et des groupes de sécurité différents de ceux de l'interface réseau principale du nœud, ces sous-réseaux et groupes de sécurité doivent se trouver dans le même VPC que le nœud.
- Pour Fargate, les sous-réseaux sont contrôlés via le profil Fargate. Pour de plus amples informations, consultez the section called "Définir des profils".

Personnalisation de l'interface réseau secondaire sur les nœuds Amazon EKS

Avant de commencer le tutoriel, veuillez effectuer les opérations suivantes :

- Passer en revue les considérations
- Avoir une bonne compréhension du fonctionnement du plug-in Amazon VPC CNI pour Kubernetes concernant la création d'interfaces réseau secondaires et l'attribution d'adresses IP aux pods. Pour de plus amples informations, veuillez consulter Allocation ENI sur GitHub.
- Version `2.12.3` ou version ultérieure `1.27.160` ou version ultérieure de l'interface de ligne de AWS commande (AWS CLI) installée et configurée sur votre appareil ou AWS CloudShell. Pour vérifier votre version actuelle, utilisez `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Les gestionnaires de packages tels que `yum` Homebrew pour macOS ont souvent plusieurs versions de retard sur la dernière version de la AWS CLI. `apt-get` Pour installer la dernière version, consultez la section Installation et configuration rapide avec aws configure dans le Guide de l'utilisateur de l'interface de ligne de AWS commande. La version de la AWS CLI installée AWS CloudShell peut également avoir plusieurs versions de retard par rapport à la dernière version. Pour le mettre à jour, consultez la section Installation de la AWS CLI dans votre répertoire de base dans le guide de AWS CloudShell l'utilisateur.
- L'outil de ligne de commande `kubectl` est installé sur votre appareil ou AWS CloudShell. Pour installer ou mettre à niveau `kubectl`, veuillez consulter the section called "Configurer kubectl et eksctl".
- Nous vous recommandons de terminer les étapes de cette rubrique dans un shell Bash. Si vous n'utilisez pas de shell Bash, certaines commandes de script telles que les caractères de continuation de ligne et la façon dont les variables sont définies et utilisées nécessitent un ajustement pour votre shell. En outre, les règles de votre shell en matière de guillemets peuvent

être différentes. Pour plus d'informations, consultez la section Utilisation de guillemets avec des chaînes dans la AWS CLI du Guide de l'utilisateur de l'interface de ligne de AWS commande.

Pour ce tutoriel, nous vous recommandons d'utiliser les exemples de valeurs, sauf dans les cas où il est noté de les remplacer. Vous pouvez remplacer n'importe quel exemple de valeur lorsque vous terminez les étapes d'un cluster de production. Nous vous recommandons de terminer toutes les étapes dans le même terminal. En effet, les variables sont définies et utilisées tout au long des étapes et n'existent pas dans différents terminaux.

Les commandes de cette rubrique sont mises en forme selon les conventions répertoriées dans la section Exemples d'utilisation de la AWS CLI. Si vous exécutez des commandes depuis la ligne de commande sur des ressources situées dans une AWS région différente de la AWS région par défaut définie dans le profil AWS CLI que vous utilisez, vous `--region us-west-2` devez ajouter des commandes en les `us-west-2` remplaçant par votre AWS région.

Lorsque vous souhaitez déployer une mise en réseau personnalisée sur votre cluster de production, passez à the section called "Étape 2 : configuration de votre VPC".

Étape 1 : créer un VPC test et un cluster

Les procédures suivantes vous aident à créer un VPC test et un cluster et à configurer une mise en réseau personnalisée pour ce cluster. Nous ne recommandons pas d'utiliser le cluster test pour les charges de travail de production, car plusieurs fonctionnalités non liées que vous pouvez utiliser sur votre cluster de production ne sont pas couvertes dans cette rubrique. Pour de plus amples informations, veuillez consulter the section called "Créer un cluster".

1. Exécutez la commande suivante afin de définir la variable `account_id`.

```
account_id=$(aws sts get-caller-identity --query Account --output text)
```

2. Créez un VPC.
   a. Si vous effectuez un déploiement sur un système de test, créez un VPC à l'aide d'un modèle Amazon EKS AWS CloudFormation .

```
aws cloudformation create-stack --stack-name my-eks-custom-networking-vpc \
  --template-url https://s3.us-west-2.amazonaws.com/amazon-eks/
cloudformation/2020-10-29/amazon-eks-vpc-private-subnets.yaml \
  --parameters ParameterKey=VpcBlock,ParameterValue=192.168.0.0/24 \
  ParameterKey=PrivateSubnet01Block,ParameterValue=192.168.0.64/27 \
```

```
  ParameterKey=PrivateSubnet02Block,ParameterValue=192.168.0.96/27 \
  ParameterKey=PublicSubnet01Block,ParameterValue=192.168.0.0/27 \
  ParameterKey=PublicSubnet02Block,ParameterValue=192.168.0.32/27
```

b. La création de la AWS CloudFormation pile prend quelques minutes. Utilisez la commande suivante pour vérifier l'état du déploiement de la pile.

```
aws cloudformation describe-stacks --stack-name my-eks-custom-networking-vpc --
query Stacks\[\].StackStatus  --output text
```

Ne passez pas à l'étape suivante tant que le résultat de la commande n'est pas `CREATE_COMPLETE`.

c. Définissez les variables avec les valeurs du sous-réseau privé IDs créé par le modèle.

```
subnet_id_1=$(aws cloudformation describe-stack-resources --stack-name my-eks-
custom-networking-vpc \
    --query "StackResources[?
LogicalResourceId=='PrivateSubnet01'].PhysicalResourceId" --output text)
subnet_id_2=$(aws cloudformation describe-stack-resources --stack-name my-eks-
custom-networking-vpc \
    --query "StackResources[?
LogicalResourceId=='PrivateSubnet02'].PhysicalResourceId" --output text)
```

d. Définissez des variables avec les zones de disponibilité des sous-réseaux récupérés lors de l'étape précédente.

```
az_1=$(aws ec2 describe-subnets --subnet-ids $subnet_id_1 --query
 'Subnets[*].AvailabilityZone' --output text)
az_2=$(aws ec2 describe-subnets --subnet-ids $subnet_id_2 --query
 'Subnets[*].AvailabilityZone' --output text)
```

3. Créez un rôle IAM de cluster.

a. Exécutez la commande suivante pour créer un fichier JSON de politique d'approbation IAM.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
```

```
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

b. Créez le rôle IAM de cluster Amazon EKS. Si nécessaire, faites précéder `eks-cluster-role-trust-policy.json` par le chemin d'accès sur votre ordinateur sur lequel vous avez enregistré le fichier lors de l'étape précédente. La commande associe la politique d'approbation que vous avez créée lors de l'étape précédente à ce rôle. Pour créer un rôle IAM, le principal IAM qui crée le rôle doit se voir attribuer l'action (autorisation) IAM `iam:CreateRole`.

```
aws iam create-role --role-name myCustomNetworkingAmazonEKSClusterRole --assume-role-policy-document file://"eks-cluster-role-trust-policy.json"
```

c. Associez la politique gérée par Amazon EKS nommée Amazon EKSCluster Policy au rôle. Pour attacher une politique IAM à un principal IAM, le principal qui attache la politique doit se voir attribuer l'une des actions (autorisations) IAM `iam:AttachUserPolicy` ou `iam:AttachRolePolicy`.

```
aws iam attach-role-policy --policy-arn arn:aws: iam::aws:policy/AmazonEKSClusterPolicy --role-name myCustomNetworkingAmazonEKSClusterRole
```

4. Créez un cluster Amazon EKS et configurez votre appareil pour qu'il communique avec lui.

a. Créer un cluster.

```
aws eks create-cluster --name my-custom-networking-cluster \
   --role-arn arn:aws: iam::$account_id:role/myCustomNetworkingAmazonEKSClusterRole \
   --resources-vpc-config subnetIds="$subnet_id_1","$subnet_id_2"
```

> **Note**
>
> Il est possible que vous receviez un message d'erreur indiquant que l'une des zones de disponibilité de votre demande ne dispose pas d'une capacité suffisante pour créer un cluster Amazon EKS. Si cela se produit, la sortie de l'erreur contient les zones de disponibilité qui peuvent prendre en charge un nouveau cluster. Essayez à nouveau de créer votre cluster avec au moins deux sous-réseaux situés dans les zones de

disponibilité prises en charge pour votre compte. Pour de plus amples informations, veuillez consulter the section called “Capacité insuffisante”.

b. La création du cluster prend quelques minutes. Utilisez la commande suivante pour vérifier l'état du déploiement du cluster.

```
aws eks describe-cluster --name my-custom-networking-cluster --query
 cluster.status
```

Ne passez pas à l'étape suivante tant que le résultat de la commande n'est pas `"ACTIVE"`.

c. Configurez `kubectl` pour communiquer avec votre cluster.

```
aws eks update-kubeconfig --name my-custom-networking-cluster
```

Étape 2 : configuration de votre VPC

Dans ce didacticiel, vous devez disposer du VPC créé dans the section called “Étape 1 : créer un VPC test et un cluster”. Pour un cluster de production, ajustez les étapes en conséquence pour votre VPC en remplaçant tous les exemples de valeur par vos propres valeurs.

1. Confirmez que la version installée du plug-in Amazon VPC CNI pour Kubernetes est la plus récente. Pour déterminer la version la plus récente du type de module complémentaire Amazon EKS et mettre à jour votre version en conséquence, consultez the section called “Mettre à jour un module complémentaire”. Pour déterminer la version la plus récente du type de module complémentaire autogéré et mettre à jour votre version en conséquence, consultez the section called “CNI Amazon VPC”.
2. Récupérez l'ID du VPC de votre cluster et stockez-le dans une variable pour une utilisation ultérieure.

```
vpc_id=$(aws eks describe-cluster --name my-custom-networking-cluster --query
 "cluster.resourcesVpcConfig.vpcId" --output text)
```

3. Associez un bloc de routage inter-domaines sans classe (CIDR) supplémentaire au VPC de votre cluster. Le bloc CIDR ne peut pas se chevaucher avec des blocs CIDR associés existants.

a. Affichez les blocs CIDR actuels associés à votre VPC.

```
aws ec2 describe-vpcs --vpc-ids $vpc_id \
```

```
    --query 'Vpcs[*].CidrBlockAssociationSet[*].{CIDRBlock: CidrBlock, State:
 CidrBlockState.State}' --out table
```

L'exemple qui suit illustre un résultat.

```
---------------------------------
|          DescribeVpcs         |
+-----------------+-------------+
|    CIDRBlock    |    State    |
+-----------------+-------------+
|  192.168.0.0/24 |  associated |
+-----------------+-------------+
```

b. Associez un bloc CIDR supplémentaire à votre VPC. Remplacez la valeur du bloc CIDR dans la commande suivante. Pour plus d'informations, consultez Associer des blocs IPv4 CIDR supplémentaires à votre VPC dans le guide de l'utilisateur Amazon VPC.

```
aws ec2 associate-vpc-cidr-block --vpc-id $vpc_id --cidr-block 192.168.1.0/24
```

c. Vérifiez que le nouveau bloc est associé.

```
aws ec2 describe-vpcs --vpc-ids $vpc_id --query
 'Vpcs[*].CidrBlockAssociationSet[*].{CIDRBlock: CidrBlock, State:
 CidrBlockState.State}' --out table
```

L'exemple qui suit illustre un résultat.

```
---------------------------------
|          DescribeVpcs         |
+-----------------+-------------+
|    CIDRBlock    |    State    |
+-----------------+-------------+
|  192.168.0.0/24 |  associated |
|  192.168.1.0/24 |  associated |
+-----------------+-------------+
```

Ne passez pas à l'étape suivante tant que le `State` de votre nouveau bloc CIDR n'est pas `associated`.

4. Créez autant de sous-réseaux que vous souhaitez utiliser dans chaque zone de disponibilité dans laquelle se trouvent vos sous-réseaux existants. Spécifiez un bloc CIDR qui se trouve dans le bloc CIDR que vous avez associé à votre VPC lors d'une étape précédente.

   a. Créez de nouveaux sous-réseaux. Remplacez les valeurs du bloc CIDR dans la commande suivante. Les sous-réseaux doivent être créés dans un bloc CIDR VPC différent de celui de vos sous-réseaux existants, mais dans les mêmes zones de disponibilité que vos sous-réseaux existants. Dans cet exemple, un sous-réseau est créé dans le nouveau bloc CIDR de chaque zone de disponibilité dans laquelle les sous-réseaux privés actuels existent. Les IDs sous-réseaux créés sont stockés dans des variables à utiliser lors des étapes ultérieures. Les valeurs `Name` correspondent aux valeurs affectées aux sous-réseaux créés à l'aide du modèle Amazon EKS VPC lors d'une étape précédente. Les noms ne sont pas obligatoires. Vous pouvez utiliser des noms différents.

```
new_subnet_id_1=$(aws ec2 create-subnet --vpc-id $vpc_id --availability-zone $az_1
 --cidr-block 192.168.1.0/27 \
    --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=my-eks-custom-
networking-vpc-PrivateSubnet01},{Key=kubernetes.io/role/internal-elb,Value=1}]' \
    --query Subnet.SubnetId --output text)
new_subnet_id_2=$(aws ec2 create-subnet --vpc-id $vpc_id --availability-zone $az_2
 --cidr-block 192.168.1.32/27 \
    --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=my-eks-custom-
networking-vpc-PrivateSubnet02},{Key=kubernetes.io/role/internal-elb,Value=1}]' \
    --query Subnet.SubnetId --output text)
```

> **Important**
>
> Par défaut, vos nouveaux sous-réseaux sont implicitement associés à la table de routage principale de votre VPC. Cette table de routage permet la communication entre toutes les ressources déployées dans le VPC. Toutefois, elle n'autorise pas la communication avec des ressources dont les adresses IP se trouvent en dehors des blocs CIDR associés à votre VPC. Vous pouvez associer votre propre table de routage à vos sous-réseaux pour modifier ce comportement. Pour plus d'informations, consultez Tables de routage de sous-réseau dans le Guide de l'utilisateur Amazon VPC.

   b. Affichez les sous-réseaux actuels dans votre VPC.

```
aws ec2 describe-subnets --filters "Name=vpc-id,Values=$vpc_id" \
```

```
    --query 'Subnets[*].{SubnetId: SubnetId,AvailabilityZone:
 AvailabilityZone,CidrBlock: CidrBlock}' \
    --output table
```

L'exemple qui suit illustre un résultat.

```
-------------------------------------------------------------------
|                          DescribeSubnets                        |
+------------------+--------------------+---------------------------+
| AvailabilityZone |     CidrBlock      |         SubnetId          |
+------------------+--------------------+---------------------------+
|  us-west-2d      |  192.168.0.0/27    |     subnet-example1       |
|  us-west-2a      |  192.168.0.32/27   |     subnet-example2       |
|  us-west-2a      |  192.168.0.64/27   |     subnet-example3       |
|  us-west-2d      |  192.168.0.96/27   |     subnet-example4       |
|  us-west-2a      |  192.168.1.0/27    |     subnet-example5       |
|  us-west-2d      |  192.168.1.32/27   |     subnet-example6       |
+------------------+--------------------+---------------------------+
```

Vous pouvez voir que les sous-réseaux se trouvant dans le bloc CIDR `192.168.1.0` que vous avez créé se trouvent dans les mêmes zones de disponibilité que les sous-réseaux dans le bloc d'adresse CIDR `192.168.0.0`.

Étape 3 : configuration des ressources Kubernetes

1. Définissez la variable d'`AWS_VPC_K8S_CNI_CUSTOM_NETWORK_CFG`environnement sur `true` dans le `aws-node` DaemonSet.

   ```
   kubectl set env daemonset aws-node -n kube-system
    AWS_VPC_K8S_CNI_CUSTOM_NETWORK_CFG=true
   ```

2. Récupérez l'ID de votre groupe de sécurité de cluster et stockez-le dans une variable pour une utilisation ultérieure. Amazon EKS crée automatiquement ce groupe de sécurité lorsque vous créez votre cluster.

   ```
   cluster_security_group_id=$(aws eks describe-cluster --name my-custom-networking-
   cluster --query cluster.resourcesVpcConfig.clusterSecurityGroupId --output text)
   ```

3. Créez une ressource personnalisée `ENIConfig` pour chaque sous-réseau dans lequel vous souhaitez déployer les pods.

a. Créez un fichier unique pour chaque configuration d'interface réseau.

Les commandes suivantes créent des fichiers `ENIConfig` distincts pour les deux sous-réseaux créés à l'étape précédente. La valeur pour `name` doit être unique. Le nom est le même que la zone de disponibilité dans laquelle se trouve le sous-réseau. Le groupe de sécurité de cluster est affecté au `ENIConfig`.

```
cat >$az_1.yaml <<EOF
apiVersion: crd.k8s.amazonaws.com/v1alpha1
kind: ENIConfig
metadata:
  name: $az_1
spec:
  securityGroups:
    - $cluster_security_group_id
  subnet: $new_subnet_id_1
EOF
```

```
cat >$az_2.yaml <<EOF
apiVersion: crd.k8s.amazonaws.com/v1alpha1
kind: ENIConfig
metadata:
  name: $az_2
spec:
  securityGroups:
    - $cluster_security_group_id
  subnet: $new_subnet_id_2
EOF
```

Pour un cluster de production, vous pouvez apporter les modifications suivantes aux commandes précédentes :

- Remplacez $cluster_security_group_id par l'ID d'un groupe de sécurité existant que vous souhaitez utiliser pour chaque `ENIConfig`.
- Nous vous recommandons de nommer vos `ENIConfigs` de la même façon que la zone de disponibilité pour laquelle vous allez utiliser la `ENIConfig`, chaque fois que c'est possible. Vous devrez peut-être utiliser pour vos `ENIConfigs` des noms différents de ceux des zones de disponibilité pour diverses raisons. Par exemple, si vous avez plus de deux sous-réseaux dans la même zone de disponibilité et que vous souhaitez les utiliser tous les deux avec une mise en réseau personnalisée, vous avez besoin de plusieurs `ENIConfigs` pour la même

zone de disponibilité. Puisque chaque `ENIConfig` nécessite un nom unique, vous ne pouvez nommer plus d'un de vos `ENIConfigs` en utilisant le nom de la zone de disponibilité.

Si vos `ENIConfig` noms ne sont pas tous identiques à ceux des zones de disponibilité, remplacez $az_1 et $az_2 par vos propres noms dans les commandes précédentes et annotez vos nœuds avec les derniers noms de ce didacticiel. ENIConfig

> **Note**
>
> Si vous ne spécifiez pas de groupe de sécurité valide pour un cluster de production et que vous utilisez :

- la version `1.8.0` ou ultérieure du plug-in Amazon VPC CNI pour Kubernetes, alors les groupes de sécurité associés à l'interface réseau Elastic principale du nœud sont utilisés.
- une version antérieure à `1.8.0` du plug-in Amazon VPC CNI pour Kubernetes, alors le groupe de sécurité par défaut du VPC est attribué aux interfaces réseau secondaires.

> **Important**
>
> - `AWS_VPC_K8S_CNI_EXTERNALSNAT=false` est un paramètre par défaut dans la configuration du plugin CNI Amazon VPC pour Kubernetes. Si vous utilisez les paramètres par défaut, le trafic destiné à des adresses IP situées en dehors des blocs CIDR associés à votre VPC utilisera les groupes de sécurité et les sous-réseaux de l'interface réseau primaire du nœud. Les sous-réseaux et groupes de sécurité définis dans votre `ENIConfigs`, qui servent à créer les interfaces réseau secondaires, ne sont pas utilisés pour ce trafic. Pour en savoir plus sur ce paramètre, consultez the section called "Trafic sortant".
> - Si vous utilisez également des groupes de sécurité pour les pods, le groupe de sécurité spécifié dans un `SecurityGroupPolicy` est utilisé à la place du groupe de sécurité défini dans `ENIConfigs`. Pour de plus amples informations, veuillez consulter the section called "Groupes de sécurité pour les pods".

b. Appliquez chaque fichier de ressource personnalisé que vous avez créé à votre cluster à l'aide des commandes suivantes.

```
kubectl apply -f $az_1.yaml
kubectl apply -f $az_2.yaml
```

4. Confirmez que vos `ENIConfigs` ont été créés.

```
kubectl get ENIConfigs
```

L'exemple qui suit illustre un résultat.

```
NAME         AGE
us-west-2a   117s
us-west-2d   105s
```

5. Si vous activez une mise en réseau personnalisée sur un cluster de production et que vous `ENIConfigs` lui avez donné un nom autre que la zone de disponibilité pour laquelle vous les utilisez, passez à l'étape suivante pour déployer des EC2 nœuds Amazon.

   Permettez à Kubernetes d'appliquer automatiquement la zone `ENIConfig` de disponibilité à tous les nouveaux EC2 nœuds Amazon créés dans votre cluster.

   a. Pour le cluster test de ce didacticiel, passez à l'étape suivante.

      Pour un cluster de production, vérifiez si une annotation contenant la clé de la variable `k8s.amazonaws.com/eniConfig` d' `ENI_CONFIG_ANNOTATION_DEF` environnement existe dans la spécification du conteneur pour le `aws-node` DaemonSet.

```
kubectl describe daemonset aws-node -n kube-system | grep
 ENI_CONFIG_ANNOTATION_DEF
```

      Si la sortie est renvoyée, l'annotation existe. Si aucune sortie n'est renvoyée, la variable n'est pas définie. Pour un cluster de production, vous pouvez utiliser ce paramètre ou le paramètre de l'étape suivante. Si vous utilisez ce paramètre, il remplace le paramètre de l'étape suivante. Dans ce didacticiel, le paramètre de l'étape suivante est utilisé.

   b. Mettez `aws-node` DaemonSet à jour votre `ENIConfig` pour appliquer automatiquement la zone de disponibilité à tous les nouveaux EC2 nœuds Amazon créés dans votre cluster.

```
kubectl set env daemonset aws-node -n kube-system
 ENI_CONFIG_LABEL_DEF=topology.kubernetes.io/zone
```

Étape 4 : Déployer des EC2 nœuds Amazon

1. Créez un rôle IAM de nœud.

a. Exécutez la commande suivante pour créer un fichier JSON de politique d'approbation IAM.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

b. Créez un rôle IAM et stockez son Amazon Resource Name (ARN) dans une variable pour une utilisation ultérieure.

```
node_role_arn=$(aws iam create-role --role-name myCustomNetworkingNodeRole --
assume-role-policy-document file://"node-role-trust-relationship.json" \
    --query Role.Arn --output text)
```

c. Attachez trois politiques gérées IAM requises au rôle IAM.

```
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::aws:policy/AmazonEKSWorkerNodePolicy \
  --role-name myCustomNetworkingNodeRole
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::aws:policy/AmazonEC2ContainerRegistryReadOnly \
  --role-name myCustomNetworkingNodeRole
aws iam attach-role-policy \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKS_CNI_Policy \
    --role-name myCustomNetworkingNodeRole
```

> **Important**
>
> Pour simplifier ce tutoriel, la politique AmazonEKS_CNI_Policy est attachée au nœud du rôle IAM. Pour un cluster de production, il est recommandé d'attacher cette politique à un rôle IAM distinct, utilisé uniquement par le plug-in Amazon VPC CNI pour

Kubernetes. Pour de plus amples informations, veuillez consulter the section called "Configuration pour IRSA".

2. Créez l'un des types de groupes de nœuds suivants. Pour déterminer le type d'instance que vous souhaitez déployer, consultez the section called "Types d'instances Amazon EC2". Pour ce didacticiel, terminez l'option Gérée, Sans modèle de lancement ou avec un modèle de lancement sans ID d'AMI spécifié. Si vous prévoyez d'utiliser le groupe de nœuds pour des charges de travail de production, nous vous recommandons de vous familiariser avec toutes les options relatives aux groupes de nœuds gérés et aux groupes de nœuds autogérés avant de déployer le groupe de nœuds.
   - Géré : déployez votre groupe de nœuds à l'aide de l'une des options suivantes :
     - Sans modèle de lancement ou avec un modèle de lancement sans ID d'AMI spécifié : exécutez la commande suivante. Dans le cadre de ce didacticiel, utilisez les exemples de valeurs. Pour un groupe de nœuds de production, remplacez toutes les valeurs d'exemple par vos propres valeurs. Le nom du groupe de nœuds ne peut pas dépasser 63 caractères. Il doit commencer par une lettre ou un chiffre, mais peut également inclure des tirets et des traits de soulignement pour les autres caractères.

       ```
       aws eks create-nodegroup --cluster-name my-custom-networking-cluster --nodegroup-
       name my-nodegroup \
           --subnets $subnet_id_1 $subnet_id_2 --instance-types t3.medium --node-role
        $node_role_arn
       ```

     - Avec un modèle de lancement avec un ID AMI spécifié

       A. Déterminez le nombre maximal de pods recommandé par Amazon EKS pour vos nœuds. Suivez les instructions de la section Nombre maximal de pods recommandé EC2 par Amazon EKS pour chaque type d'instance Amazon, `--cni-custom-networking-enabled` en complément de l'étape 3 de cette rubrique. Notez la sortie pour l'utiliser lors de l'étape suivante.

       B. Dans votre modèle de lancement, spécifiez un ID d'AMI optimisé pour Amazon EKS ou une AMI personnalisée créée à partir de l'AMI optimisée pour Amazon EKS, puis déployez le groupe de nœuds avec un modèle de lancement et fournissez les données utilisateur suivantes dans le modèle de lancement. Ces données utilisateur transmettent des arguments à la `NodeConfig` spécification. Pour plus d'informations à ce sujet NodeConfig, consultez la référence de NodeConfig l'API. Vous pouvez remplacer `20` par la valeur de l'étape précédente (recommandé) ou par votre propre valeur.

```
---
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="BOUNDARY"
--BOUNDARY
Content-Type: application/node.eks.aws

---
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
    name: my-cluster
    ...
    kubelet:
      config:
        maxPods: 20
```

Si vous avez créé une AMI personnalisée qui n'est pas basée sur l'AMI optimisée Amazon EKS, vous devez alors créer la configuration manuellement.

- Autogéré

  i. Déterminez le nombre maximal de pods recommandé par Amazon EKS pour vos nœuds. Suivez les instructions de la section the section called "Nombre maximal de pods recommandé par Amazon EKS pour chaque type d'instance Amazon EC2", en ajoutant `--cni-custom-networking-enabled` à l'étape 3 de cette rubrique. Notez la sortie pour l'utiliser lors de l'étape suivante.

  ii. Déployez le groupe de nœuds à l'aide des instructions contenues dans the section called "Amazon Linux".

> **Note**
>
> Si vous souhaitez que les nœuds d'un cluster de production prennent en charge un nombre de pods nettement plus élevé, exécutez de nouveau le script mentionné dans the section called "Nombre maximal de pods recommandé par Amazon EKS pour chaque type d'instance Amazon EC2". Ajoutez également l'option `--cni-prefix-delegation-enabled` à la commande. Par exemple, `110` est renvoyé pour un type d'instance `m5.large`. Pour obtenir des instructions sur la façon d'activer cette capacité,

consultez the section called "Augmentation du nombre d'adresses IP". Vous pouvez utiliser cette capacité avec une mise en réseau personnalisée.

3. La création d'un groupe de nœuds dure plusieurs minutes. Vous pouvez vérifier l'état de la création d'un groupe de nœuds géré à l'aide de la commande suivante.

```
aws eks describe-nodegroup --cluster-name my-custom-networking-cluster --nodegroup-
name my-nodegroup --query nodegroup.status --output text
```

Ne passez pas à l'étape suivante tant que le résultat renvoyé n'est pas `ACTIVE`.

4. Pour le didacticiel, vous pouvez ignorer cette étape.

Pour un cluster de production, si vous n'avez pas donné à vos `ENIConfigs` le même nom que la zone de disponibilité dans laquelle elles sont utilisées, vous devez annoter les nœuds avec le nom de l'`ENIConfig` devant leur être associé. Cette étape n'est pas nécessaire si vous avez un seul sous-réseau par zone de disponibilité et que vous avez nommé vos `ENIConfigs` avec les mêmes noms que vos zones de disponibilité. En effet, dans ce cas, le plug-in Amazon VPC CNI pour Kubernetes associe automatiquement la bonne `ENIConfig` au nœud, comme configuré lors d'une étape précédente.

a. Obtenez la liste des nœuds de votre cluster.

```
kubectl get nodes
```

L'exemple qui suit illustre un résultat.

```
NAME                                           STATUS   ROLES    AGE     VERSION
ip-192-168-0-126.us-west-2.compute.internal    Ready    <none>   8m49s   v1.22.9-
eks-810597c
ip-192-168-0-92.us-west-2.compute.internal     Ready    <none>   8m34s   v1.22.9-
eks-810597c
```

b. Déterminez dans quelle zone de disponibilité se trouve chaque nœud. Exécutez la commande suivante pour chaque nœud renvoyé précédemment, en remplaçant les adresses IP selon le résultat précédent.

```
aws ec2 describe-instances --filters Name=network-interface.private-dns-
name,Values=ip-192-168-0-126.us-west-2.compute.internal \
```

```
--query 'Reservations[].Instances[].{AvailabilityZone: Placement.AvailabilityZone,
 SubnetId: SubnetId}'
```

L'exemple qui suit illustre un résultat.

```
[
    {
        "AvailabilityZone": "us-west-2d",
        "SubnetId": "subnet-Example5"
    }
]
```

c. Annotez chaque nœud avec le `ENIConfig` que vous avez créé pour l'ID de sous-réseau et la zone de disponibilité. Vous ne pouvez annoter un nœud qu'avec un seul `ENIConfig`, bien que plusieurs nœuds puissent être annotés avec le même `ENIConfig`. Remplacez les exemples de valeurs par les vôtres.

```
kubectl annotate node ip-192-168-0-126.us-west-2.compute.internal
 k8s.amazonaws.com/eniConfig=EniConfigName1
kubectl annotate node ip-192-168-0-92.us-west-2.compute.internal
 k8s.amazonaws.com/eniConfig=EniConfigName2
```

5. Si vous aviez déjà des nœuds exécutant des pods dans un cluster de production avant la bascule vers la fonctionnalité de réseau personnalisé, effectuez les actions suivantes :

   a. Assurez-vous que vous disposez de nœuds disponibles utilisant la fonctionnalité de mise en réseau personnalisée.

   b. Cordonnez et videz les nœuds pour arrêter les pods de manière contrôlée. Pour plus d'informations, consultez Drainer un nœud en toute sécurité dans la documentation Kubernetes.

   c. Mettez fin aux nœuds. Si les nœuds se trouvent dans un groupe de nœuds géré existant, vous pouvez supprimer le groupe de nœuds. Exécutez la commande suivante.

```
aws eks delete-nodegroup --cluster-name my-custom-networking-cluster --nodegroup-
name my-nodegroup
```

   Seuls les nouveaux nœuds enregistrés auprès du label `k8s.amazonaws.com/eniConfig` utilisent la fonctionnalité de réseaux personnalisés.

6. Confirmez que les pods reçoivent bien une adresse IP appartenant à l'un des blocs CIDR associés aux sous-réseaux créés précédemment.

```
kubectl get pods -A -o wide
```

L'exemple qui suit illustre un résultat.

```
NAMESPACE     NAME                       READY   STATUS    RESTARTS   AGE     IP
           NODE                                            NOMINATED NODE   READINESS
 GATES
kube-system   aws-node-2rkn4             1/1     Running   0          7m19s
 192.168.0.92    ip-192-168-0-92.us-west-2.compute.internal    <none>
 <none>
kube-system   aws-node-k96wp             1/1     Running   0          7m15s
 192.168.0.126   ip-192-168-0-126.us-west-2.compute.internal   <none>
 <none>
kube-system   coredns-657694c6f4-smcgr   1/1     Running   0          56m
 192.168.1.23    ip-192-168-0-92.us-west-2.compute.internal    <none>
 <none>
kube-system   coredns-657694c6f4-stwv9   1/1     Running   0          56m
 192.168.1.28    ip-192-168-0-92.us-west-2.compute.internal    <none>
 <none>
kube-system   kube-proxy-jgshq           1/1     Running   0          7m19s
 192.168.0.92    ip-192-168-0-92.us-west-2.compute.internal    <none>
 <none>
kube-system   kube-proxy-wx9vk           1/1     Running   0          7m15s
 192.168.0.126   ip-192-168-0-126.us-west-2.compute.internal   <none>
 <none>
```

Vous pouvez constater que les pods coredns reçoivent des adresses IP issues du bloc CIDR `192.168.1.0` que vous avez ajouté à votre VPC. Sans mise en réseau personnalisée, des adresses auraient été attribuées à partir du bloc CIDR `192.168.0.0`, car il s'agissait du seul bloc CIDR initialement associé au VPC.

Si le champ `spec` d'un pod contient `hostNetwork=true`, cela signifie que le pod utilise l'adresse IP principale du nœud. Il ne reçoit pas d'adresse IP provenant des nouveaux sous-réseaux ajoutés. Par défaut, cette valeur indique `false`. Cette valeur est définie sur `true` pour le `kube-proxy` et pour les pods du plug-in Amazon VPC CNI pour Kubernetes (`aws-node`) exécutés sur votre cluster. C'est la raison pour laquelle le `kube-proxy` et les pods `aws-node` du plug-in ne reçoivent pas d'adresses de type 192.168.1.x dans le résultat affiché précédemment. Pour plus d'informations sur les `hostNetwork` paramètres d'un pod, consultez la section PodSpec v1 core dans la référence de l'API Kubernetes.

Étape 5 : suppression des ressources du didacticiel

Une fois le didacticiel terminé, nous vous recommandons de supprimer les ressources que vous avez créées. Vous pouvez ensuite ajuster les étapes pour activer la mise en réseau personnalisée pour un cluster de production.

1. Si le groupe de nœuds que vous avez créé n'était qu'à des fins de test, supprimez-le.

```
aws eks delete-nodegroup --cluster-name my-custom-networking-cluster --nodegroup-name
 my-nodegroup
```

2. Même une fois que la sortie de la AWS CLI indique que le cluster est supprimé, le processus de suppression peut ne pas être terminé. Le processus de suppression prend quelques minutes. Vérifiez qu'il est bien finalisé en exécutant la commande suivante.

```
aws eks describe-nodegroup --cluster-name my-custom-networking-cluster --nodegroup-
name my-nodegroup --query nodegroup.status --output text
```

   Ne poursuivez pas tant que le résultat renvoyé n'est pas similaire à l'exemple ci-dessous.

```
An error occurred (ResourceNotFoundException) when calling the DescribeNodegroup
 operation: No node group found for name: my-nodegroup.
```

3. Si le groupe de nœuds que vous avez créé n'était qu'à des fins de test, supprimez le rôle IAM du nœud.

   a. Détachez les politiques du rôle.

```
aws iam detach-role-policy --role-name myCustomNetworkingNodeRole --policy-arn
 arn:aws: iam::aws:policy/AmazonEKSWorkerNodePolicy
aws iam detach-role-policy --role-name myCustomNetworkingNodeRole --policy-arn
 arn:aws: iam::aws:policy/AmazonEC2ContainerRegistryReadOnly
aws iam detach-role-policy --role-name myCustomNetworkingNodeRole --policy-arn
 arn:aws: iam::aws:policy/AmazonEKS_CNI_Policy
```

   b. Supprimez le rôle.

```
aws iam delete-role --role-name myCustomNetworkingNodeRole
```

4. Supprimez le cluster.

```
aws eks delete-cluster --name my-custom-networking-cluster
```

Vérifiez que le cluster est supprimé avec la commande suivante.

```
aws eks describe-cluster --name my-custom-networking-cluster --query cluster.status
 --output text
```

Lorsqu'une sortie similaire à la suivante est renvoyée, le cluster est correctement supprimé.

```
An error occurred (ResourceNotFoundException) when calling the DescribeCluster
 operation: No cluster found for name: my-custom-networking-cluster.
```

5. Supprimez le rôle IAM de cluster.
   a. Détachez les politiques du rôle.

      ```
      aws iam detach-role-policy --role-name myCustomNetworkingAmazonEKSClusterRole --
      policy-arn arn:aws: iam::aws:policy/AmazonEKSClusterPolicy
      ```

   b. Supprimez le rôle.

      ```
      aws iam delete-role --role-name myCustomNetworkingAmazonEKSClusterRole
      ```

6. Supprimez les sous-réseaux que vous avez créés à l'étape précédente.

   ```
   aws ec2 delete-subnet --subnet-id $new_subnet_id_1
   aws ec2 delete-subnet --subnet-id $new_subnet_id_2
   ```

7. Supprimez le VPC que vous avez créé.

   ```
   aws cloudformation delete-stack --stack-name my-eks-custom-networking-vpc
   ```

Attribution de davantage d'adresses IP aux nœuds Amazon EKS avec des préfixes

S'applique à : nœuds Linux et Windows avec instances Amazon EC2

S'applique à : sous-réseaux publics et privés

Chaque instance Amazon EC2 prend en charge un nombre maximum d'interfaces réseau élastiques et un nombre maximum d'adresses IP pouvant être attribuées à chaque interface réseau. Chaque

nœud a besoin d'une adresse IP pour chaque interface réseau. Toutes les autres adresses IP disponibles peuvent être attribuées aux `Pods`. Chaque `Pod` a besoin de sa propre adresse IP. Il peut arriver que des nœuds disposent encore de ressources de calcul et de mémoire, mais ne puissent plus accueillir de nouveaux `Pods`, car ils n'ont plus d'adresses IP disponibles à attribuer aux `Pods`.

Vous pouvez augmenter le nombre d'adresses IP que les nœuds peuvent attribuer aux `Pods` en utilisant des préfixes IP, plutôt que des adresses IP secondaires individuelles. Chaque préfixe inclut plusieurs adresses IP. Si vous ne configurez pas votre cluster pour l'attribution de préfixes IP, le cluster doit effectuer un plus grand nombre d'appels d'interface de programmation d'application (API) Amazon EC2 pour configurer les interfaces réseau et les adresses IP nécessaires à la connectivité des pods. À mesure que les clusters augmentent en taille, la fréquence de ces appels d'API peut entraîner des délais plus longs pour le lancement des pods et des instances. Cela entraîne des retards de mise à l'échelle pour répondre à la demande d'applications volumineuses et épineuses, et ajoute des coûts et des frais généraux de gestion, car vous devez allouer des clusters et des VPC supplémentaires pour répondre aux exigences de mise à l'échelle. Pour plus d'informations, consultez les Seuils de capacité de mise à l'échelle de Kubernetes sur GitHub.

Compatibilité avec les fonctionnalités du module complémentaire plug-in Amazon VPC CNI pour Kubernetes

Vous pouvez utiliser les préfixes IP avec les fonctionnalités suivantes :

- Traduction d'adresses réseau source IPv4 ; pour plus d'informations, consultez the section called "Trafic sortant".
- Adresses IPv6 pour les clusters, pods et services ; pour plus d'informations, consultez the section called "IPv6".
- Restriction du trafic à l'aide des politiques réseau Kubernetes ; pour plus d'informations, consultez the section called "Politiques Kubernetes".

La liste suivante présente les paramètres pertinents du module complémentaire plug-in Amazon VPC CNI pour Kubernetes. Pour plus d'informations sur chaque paramètre, consultez amazon-vpc-cni-k8s sur GitHub.

- `WARM_IP_TARGET`
- `MINIMUM_IP_TARGET`
- `WARM_PREFIX_TARGET`

Considérations

Considérations liées à l'utilisation de cette fonctionnalité :

- Chaque type d'instance Amazon EC2 prend en charge un nombre maximal de pods. Si votre groupe de nœuds géré comporte plusieurs types d'instances, le plus petit nombre maximal de pods parmi ces types d'instances est appliqué à tous les nœuds du cluster.
- Par défaut, le nombre maximum de `Pods` que vous pouvez exécuter sur un nœud est de 110, mais vous pouvez modifier ce nombre. Si vous modifiez le numéro et que vous disposez d'un groupe de nœuds géré existant, la prochaine mise à jour d'AMI ou de modèle de lancement de votre groupe de nœuds entraînera la création de nouveaux nœuds avec la valeur modifiée.
- Lorsque vous passez de l'attribution d'adresses IP à l'attribution de préfixes IP, nous vous recommandons de créer de nouveaux groupes de nœuds afin d'augmenter le nombre d'adresses IP disponibles, plutôt que de remplacer progressivement les nœuds existants. Exécuter des pods sur un nœud ayant à la fois des adresses IP classiques et des préfixes IP peut entraîner une incohérence dans la capacité IP annoncée, ce qui peut affecter les charges de travail futures sur ce nœud. Pour connaître la méthode recommandée pour effectuer la transition, consultez Mode de délégation de préfixes pour Linux dans le Guide des bonnes pratiques Amazon EKS.
- La portée des groupes de sécurité s'applique au niveau du nœud ; pour plus d'informations, consultez Groupe de sécurité.
- Les préfixes IP attribués à une interface réseau permettent une forte densité de pods par nœud et offrent les meilleurs temps de lancement.
- Les préfixes IP et les adresses IP sont associés aux interfaces réseau Elastic Amazon EC2 standards. Les pods nécessitant des groupes de sécurité spécifiques se voient attribuer l'adresse IP principale d'une interface réseau de branche. Vous pouvez mélanger, sur un même nœud, des pods recevant des adresses IP individuelles ou des adresses issues de préfixes IP, ainsi que des pods recevant des interfaces réseau dérivées.
- Pour les clusters dotés de nœuds Linux uniquement.
  - Après avoir configuré le module complémentaire plug-in Amazon VPC CNI pour Kubernetes pour attribuer des préfixes aux interfaces réseau, vous ne pouvez pas rétrograder ce module à une version antérieure à la version `1.9.0` (ou `1.10.1`), sans supprimer tous les nœuds de tous les groupes de nœuds de votre cluster.
  - Si vous utilisez également des groupes de sécurité pour les pods, avec les paramètres `POD_SECURITY_GROUP_ENFORCING_MODE=standard` et `AWS_VPC_K8S_CNI_EXTERNALSNAT=false`, alors lorsque vos pods communiquent avec des

points de terminaison situés hors de votre VPC, ce sont les groupes de sécurité du nœud qui sont appliqués, et non les groupes de sécurité attribués aux pods.

Si vous utilisez également des groupes de sécurité pour les pods, avec `POD_SECURITY_GROUP_ENFORCING_MODE=strict`, lorsque vos `Pods` communiquent avec des points de terminaison hors de votre VPC, ce sont les groupes de sécurité `Pod's` qui sont utilisés.

Augmentation du nombre d'adresses IP disponibles pour un nœud Amazon EKS

Vous pouvez augmenter le nombre d'adresses IP que les nœuds peuvent attribuer aux pods en attribuant des préfixes IP, plutôt qu'en attribuant des adresses IP secondaires individuelles aux nœuds.

Conditions préalables

- Vous devez disposer d'un cluster existant. Pour en déployer un, consultez the section called "Créer un cluster".
- Les sous-réseaux dans lesquels se trouvent vos nœuds Amazon EKS doivent comporter suffisamment de blocs de routage inter-domaines sans classe (CIDR) contigus `/28` (pour les clusters `IPv4`) ou `/80` (pour les clusters `IPv6`). Vous ne pouvez avoir que des nœuds Linux dans un cluster `IPv6`. L'utilisation de préfixes IP peut échouer si les adresses IP sont dispersées dans le CIDR du sous-réseau. Nous vous recommandons la procédure suivante :
  - L'utilisation d'une réservation CIDR de sous-réseau garantit que, même si certaines adresses IP de la plage réservée sont encore utilisées, elles ne seront pas réattribuées après leur libération. Cela permet de s'assurer que les préfixes sont disponibles pour l'allocation sans segmentation.
  - Utilisez de nouveaux sous-réseaux spécifiquement utilisés pour exécuter les charges de travail auxquelles les préfixes IP sont attribués. Les charges de travail Windows et Linux peuvent fonctionner dans le même sous-réseau lors de l'attribution de préfixes IP.
- Pour attribuer des préfixes IP à vos nœuds, ceux-ci doivent être basés sur AWS Nitro. Les instances qui ne sont pas basées sur Nitro continuent d'allouer des adresses IP secondaires individuelles, mais disposent d'un nombre beaucoup plus faible d'adresses à attribuer aux pods que les instances basées sur Nitro.
- Pour les clusters avec des nœuds Linux uniquement : si votre cluster est configuré pour la famille `IPv4`, la version `1.9.0` ou une version ultérieure du module complémentaire plug-in Amazon VPC CNI pour Kubernetes doit être installée. Vous pouvez vérifier votre version actuelle à l'aide de la commande suivante.

```
kubectl describe daemonset aws-node --namespace kube-system | grep Image | cut -d "/"
 -f 2
```

Si votre cluster est configuré pour la famille `IPv6`, la version `1.10.1` du module complémentaire doit être installée. Si la version de votre plugin est antérieure aux versions requises, vous devez la mettre à jour. Pour plus d'informations, consultez les sections de mise à jour d'Assign IPs to Pods with the Amazon VPC CNI.

- Pour les clusters avec des nœuds Windows uniquement
  - Vous devez avoir activé la prise en charge de Windows pour votre cluster. Pour de plus amples informations, veuillez consulter the section called “Activer la prise en charge de Windows”.

Attribution de préfixes d'adresses IP aux nœuds

Configurez votre cluster pour attribuer des préfixes d'adresses IP aux nœuds. Suivez la procédure correspondant au système d'exploitation de vos nœuds.

Linux

1. Activez le paramètre pour attribuer des préfixes aux interfaces réseau pour le CNI Amazon DaemonSet VPC. Lors du déploiement d'un cluster, la version `1.10.1` ou ultérieure du module complémentaire plug-in Amazon VPC CNI est déployée avec celui-ci. Si vous avez créé le cluster avec la famille `IPv6`, ce paramètre a été réglé sur `true` par défaut. Si vous avez créé le cluster avec la famille `IPv4`, ce paramètre a été réglé sur `false` par défaut.

   ```
   kubectl set env daemonset aws-node -n kube-system ENABLE_PREFIX_DELEGATION=true
   ```

   Important

   Même si votre sous-réseau dispose d'adresses IP disponibles, s'il n'a pas de blocs `/28` contigus disponibles, vous verrez l'erreur suivante dans les journaux du plug-in Amazon VPC CNI pour Kubernetes.

   ```
   InsufficientCidrBlocks: The specified subnet does not have enough free cidr
    blocks to satisfy the request
   ```

   Cela peut se produire en raison de la fragmentation des adresses IP secondaires existantes réparties sur un sous-réseau. Pour résoudre cette erreur, créez un nouveau

sous-réseau et lancez-y des pods, ou utilisez une réservation CIDR du EC2 sous-réseau Amazon pour réserver de l'espace au sein d'un sous-réseau à utiliser avec l'attribution de préfixes. Pour plus d'informations, consultez la section Réservations CIDR de sous-réseau dans le Guide de l'utilisateur Amazon VPC.

2. Si vous prévoyez de déployer un groupe de nœuds gérés sans modèle de lancement ou avec un modèle de lancement sans AMI spécifié, et que vous utilisez une version du module complémentaire plug-in Amazon VPC CNI égale ou supérieure aux versions répertoriées dans les prérequis, passez à l'étape suivante. Les groupes de nœuds gérés calculent automatiquement le nombre maximal de pods.

   Si vous déployez un groupe de nœuds autogéré ou un groupe de nœuds gérés avec un modèle de lancement dans lequel vous avez spécifié une AMI, vous devez déterminer le nombre maximal de pods recommandé par Amazon EKS pour vos nœuds. Suivez les instructions de la section Nombre maximal de pods recommandé par Amazon EKS pour chaque type d' EC2 instance Amazon, en ajoutant `--cni-prefix-delegation-enabled` à l'étape 3. Notez la sortie pour une utilisation ultérieure.

   > **Important**
   >
   > Les groupes de nœuds gérés appliquent un nombre maximal sur la valeur de `maxPods`. Pour les instances avec moins de 30 v, CPUs le nombre maximum est de 110 et pour toutes les autres instances, le nombre maximum est de 250. Ce nombre maximal est appliqué que la délégation du préfixe soit activée ou non.

3. Si vous utilisez un cluster configuré pour `IPv6`, passez à l'étape suivante.

   Spécifiez les paramètres dans l'une des options suivantes. Pour déterminer quelle option vous convient le mieux et quelle valeur lui fournir, consultez WARM_PREFIX_TARGET, WARM_IP_TARGET et MINIMUM_IP_TARGET on. GitHub

   Vous pouvez remplacer la commande example values par une valeur supérieure à zéro.

   - `WARM_PREFIX_TARGET`

     ```
     kubectl set env ds aws-node -n kube-system WARM_PREFIX_TARGET=1
     ```

   - `WARM_IP_TARGET` ou `MINIMUM_IP_TARGET` : si l'une ou l'autre des valeurs est définie, elle remplace toute valeur définie pour `WARM_PREFIX_TARGET`.

```
kubectl set env ds aws-node -n kube-system WARM_IP_TARGET=5
```

```
kubectl set env ds aws-node -n kube-system MINIMUM_IP_TARGET=2
```

4. Créez l'un des types de groupes de nœuds suivants avec au moins un type d'instance Amazon EC2 Nitro Amazon Linux 2023. Pour obtenir la liste des types d'instances Nitro, consultez la section Instances créées sur le système Nitro dans le guide de EC2 l'utilisateur Amazon. Cette fonctionnalité n'est pas prise en charge sur Windows. Pour les options qui incluent *110*, remplacez-le par la valeur de l'étape 3 (recommandée) ou par votre propre valeur.

   - Autogéré : déployez le groupe de nœuds en suivant les instructions de la section Création de nœuds Amazon Linux autogérés. Avant de créer la CloudFormation pile, ouvrez le fichier modèle et ajustez `NodeLaunchTemplate` le `UserData` dans le comme suit

     ```
     ...
                 apiVersion: node.eks.aws/v1alpha1
                 kind: NodeConfig
                 spec:
                   cluster:
                     name: ${ClusterName}
                     apiServerEndpoint: ${ApiServerEndpoint}
                     certificateAuthority: ${CertificateAuthorityData}
                     cidr: ${ServiceCidr}
                   kubelet:
                     config:
                       maxPods: 110
     ...
     ```

     Si vous utilisez `eksctl` pour créer le groupe de nœuds, vous pouvez utiliser la commande suivante.

     ```
     eksctl create nodegroup --cluster my-cluster --managed=false --max-pods-per-node
      110
     ```

   - Géré : déployez votre groupe de nœuds à l'aide de l'une des options suivantes :

     - Sans modèle de lancement ou avec un modèle de lancement sans ID d'AMI spécifié : suivez la procédure décrite dans Création d'un groupe de nœuds géré pour votre cluster. Les groupes de nœuds gérés calculent automatiquement pour vous la valeur `max-pods` recommandée par Amazon EKS.

- Avec un modèle de lancement avec un ID d'AMI spécifié : dans votre modèle de lancement, spécifiez un ID d'AMI optimisé pour Amazon EKS ou une AMI personnalisée créée à partir de l'AMI optimisée pour Amazon EKS, puis Déployez le groupe de nœuds avec un modèle de lancement et fournissez les données utilisateur suivantes dans le modèle de lancement. Ces données utilisateur transmettent un `NodeConfig` objet à lire par l'`nodeadm`outil sur le nœud. Pour plus d'informations`nodeadm`, consultez la documentation de nodeadm.

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="//"

--//
Content-Type: application/node.eks.aws

---
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
 cluster:
   apiServerEndpoint: [.replaceable]`my-cluster`
   certificateAuthority: [.replaceable]`LS0t...`
   cidr: [.replaceable]`10.100.0.0/16`
   name: [.replaceable]`my-cluster
 kubelet:
   config:
     maxPods: [.replaceable]`110`
--//--
```

Si vous utilisez `eksctl` pour créer le groupe de nœuds, vous pouvez utiliser la commande suivante.

```
eksctl create nodegroup --cluster my-cluster --max-pods-per-node 110
```

Si vous avez créé une AMI personnalisée qui n'est pas basée sur l'AMI optimisée Amazon EKS, vous devez alors créer la configuration manuellement.

> **Note**
>
> Si vous souhaitez également attribuer des adresses IP aux pods à partir d'un sous-réseau différent de celui de l'instance, vous devez activer cette fonctionnalité à cette étape.

Pour de plus amples informations, veuillez consulter the section called "Mise en réseau personnalisée".

Windows

1. Activez l'attribution de préfixes IP.
   a. Ouvrez le `amazon-vpc-cni` `ConfigMap` pour le modifier.

   ```
   kubectl edit configmap -n kube-system amazon-vpc-cni -o yaml
   ```

   b. Ajoutez les lignes suivantes à la section `data`.

   ```
     enable-windows-prefix-delegation: "true"
   ```

   c. Enregistrez le fichier et fermez l'éditeur.
   d. Confirmez que la ligne a été ajoutée à la `ConfigMap`.

   ```
   kubectl get configmap -n kube-system amazon-vpc-cni -o "jsonpath={.data.enable-
   windows-prefix-delegation}"
   ```

   Si le résultat renvoyé n'est pas `true`, il se peut qu'il y ait eu une erreur. Essayez à nouveau d'effectuer l'étape.

   > **Important**
   >
   > Même si votre sous-réseau dispose d'adresses IP disponibles, s'il n'a pas de blocs `/28` contigus disponibles, vous verrez l'erreur suivante dans les journaux du plug-in Amazon VPC CNI pour Kubernetes.
   >
   > ```
   > InsufficientCidrBlocks: The specified subnet does not have enough free cidr
   >  blocks to satisfy the request
   > ```
   >
   > Cela peut se produire en raison de la fragmentation des adresses IP secondaires existantes réparties sur un sous-réseau. Pour résoudre cette erreur, créez un nouveau sous-réseau et lancez-y des pods, ou utilisez une réservation CIDR du EC2 sous-réseau Amazon pour réserver de l'espace au sein d'un sous-réseau à utiliser avec

l'attribution de préfixes. Pour plus d'informations, consultez la section Réservations CIDR de sous-réseau dans le Guide de l'utilisateur Amazon VPC.

2. (Facultatif) Spécifiez une configuration supplémentaire pour contrôler le comportement de pré-échelonnement et d'échelonnement dynamique de votre cluster. Pour plus d'informations, consultez la section Options de configuration avec le mode de délégation de préfixes sous Windows activé GitHub.

   a. Ouvrez le `amazon-vpc-cni ConfigMap` pour le modifier.

   ```
   kubectl edit configmap -n kube-system amazon-vpc-cni -o yaml
   ```

   b. Remplacez les valeurs d'exemple par une valeur supérieure à zéro et ajoutez les entrées dont vous avez besoin dans la `data` section du`ConfigMap`. Si vous définissez une valeur pour `warm-ip-target` ou `minimum-ip-target`, la valeur remplace toute valeur définie pour `warm-prefix-target`.

   ```
   warm-prefix-target: "1"
   warm-ip-target: "5"
   minimum-ip-target: "2"
   ```

   c. Enregistrez le fichier et fermez l'éditeur.

3. Créez des groupes de nœuds Windows avec au moins un type d'instance Amazon EC2 Nitro. Pour obtenir la liste des types d'instances Nitro, consultez la section Instances créées sur le système Nitro dans le guide de EC2 l'utilisateur Amazon. Par défaut, le nombre maximum de pods que vous pouvez déployer sur un nœud est de 110. Si vous souhaitez augmenter ou diminuer ce nombre, indiquez ce qui suit dans les données utilisateur de la configuration d'amorçage. Remplacez *`max-pods-quantity`* par votre valeur maximale de pods.

   ```
   -KubeletExtraArgs '--max-pods=max-pods-quantity'
   ```

   Si vous déployez des groupes de nœuds gérés, cette configuration doit être ajoutée dans le modèle de lancement. Pour de plus amples informations, veuillez consulter the section called “Modèles de lancement”. Pour plus d’informations sur les paramètres du script d’amorçage Windows, consultez the section called “Paramètres de configuration du script d'amorçage”.

Détermination du nombre maximal de pods et du nombre d'adresses IP disponibles

1. Une fois que vos nœuds sont déployés, affichez les nœuds de votre cluster.

   ```
   kubectl get nodes
   ```

   L'exemple qui suit illustre un résultat.

   ```
   NAME                                           STATUS     ROLES    AGE   VERSION
   ip-192-168-22-103.region-code.compute.internal   Ready      <none>   19m   v1.XX.X-
   eks-6b7464
   ip-192-168-97-94.region-code.compute.internal    Ready      <none>   19m   v1.XX.X-
   eks-6b7464
   ```

2. Décrivez l'un des nœuds pour déterminer la valeur de `max-pods` pour le nœud et le nombre d'adresses IP disponibles. Remplacez *`192.168.30.193`* par l'adresse `IPv4` dans le nom de l'un de vos nœuds renvoyés dans la sortie précédente.

   ```
   kubectl describe node ip-192-168-30-193.region-code.compute.internal | grep 'pods\|
   PrivateIPv4Address'
   ```

   L'exemple qui suit illustre un résultat.

   ```
   pods:                                      110
   vpc.amazonaws.com/PrivateIPv4Address:  144
   ```

   Dans la sortie précédente, `110` c'est le nombre maximum de pods que Kubernetes déploiera sur le nœud, même si des adresses *`144`* IP sont disponibles.

Attribution de groupes de sécurité à des pods individuels

S'applique à : nœuds Linux dotés d' EC2 instances Amazon

S'applique à : sous-réseaux privés

Les groupes de sécurité pour Pods intègrent les groupes EC2 de sécurité Amazon aux Pods Kubernetes. Vous pouvez utiliser les groupes EC2 de sécurité Amazon pour définir des règles qui autorisent le trafic réseau entrant et sortant vers et depuis les pods que vous déployez vers des nœuds exécutés sur de nombreux types d' EC2 instances Amazon et sur Fargate. Pour une

explication détaillée de cette fonctionnalité, consultez l'article de blog Introducing security groups for Pods.

Compatibilité avec les fonctionnalités du module complémentaire plug-in Amazon VPC CNI pour Kubernetes

Vous pouvez utiliser les groupes de sécurité pour les pods avec les fonctionnalités suivantes :

- IPv4 Traduction de l'adresse réseau source - Pour plus d'informations, consultezthe section called "Trafic sortant".
- IPv6 adresses vers les clusters, les pods et les services - Pour plus d'informations, consultezthe section called "IPv6".
- Restriction du trafic à l'aide des politiques réseau Kubernetes ; pour plus d'informations, consultez the section called "Politiques Kubernetes".

Considérations

Avant de déployer des groupes de sécurité pour les pods, veuillez tenir compte des limitations et conditions suivantes :

- Les groupes de sécurité pour les pods ne peuvent pas être utilisés avec les nœuds Windows ou le mode automatique EKS.
- Les groupes de sécurité pour Pods peuvent être utilisés avec des clusters configurés pour la `IPv6` famille contenant des EC2 nœuds Amazon à l'aide de la version 1.16.0 ou ultérieure du plugin Amazon VPC CNI. Vous pouvez utiliser des groupes de sécurité pour les pods avec des clusters configurés pour la famille `IPv6` qui contiennent uniquement des nœuds Fargate à l'aide de la version 1.7.7 ou ultérieure du plug-in CNI Amazon VPC. Pour de plus amples informations, consultez the section called "IPv6".
- Les groupes de sécurité pour Pods sont pris en charge par la plupart des familles d' EC2 instances Amazon basées sur Nitro, mais pas par toutes les générations d'une famille. Par exemple, les familles et générations d'instances `m5`, `c5`, `r5`, `m6g`, `c6g` et `r6g` sont prises en charge. Aucun type d'instance de la famille `t` n'est pris en charge. Pour obtenir la liste complète des types d'instances pris en charge, consultez le fichier limits.go sur. GitHub Vos nœuds doivent être l'un des types d'instance répertoriés qui ont `IsTrunkingCompatible: true` dans ce fichier.
- Si vous utilisez conjointement un réseau personnalisé et des groupes de sécurité pour les pods, le groupe de sécurité spécifié par les groupes de sécurité pour les pods est utilisé à la place du groupe de sécurité spécifié dans `ENIConfig`.

- Si vous utilisez la version `1.10.2` ou antérieure du plug-in CNI Amazon VPC et que vous incluez le paramètre `terminationGracePeriodSeconds` dans votre spécification de pod, la valeur de ce paramètre ne peut pas être zéro.
- Si vous utilisez la version `1.10` ou antérieure du plug-in CNI Amazon VPC, ou la version `1.11` avec `POD_SECURITY_GROUP_ENFORCING_MODE=strict`, qui est le paramètre par défaut, les services Kubernetes de type `NodePort` et `LoadBalancer` utilisant des cibles d'instance avec une `externalTrafficPolicy` définie sur `Local` ne sont pas pris en charge avec les pods auxquels vous attribuez des groupes de sécurité. Pour plus d'informations sur l'utilisation d'un équilibreur de charge avec des cibles d'instance, consultez the section called "Répartition de charge réseau".
- Si vous utilisez la version `1.10` ou antérieure du plug-in CNI Amazon VPC ou la version `1.11` avec `POD_SECURITY_GROUP_ENFORCING_MODE=strict`, qui est le paramètre par défaut, le NAT source est désactivé pour le trafic sortant des pods auxquels des groupes de sécurité sont attribués, afin que les règles de groupe de sécurité sortantes soient appliquées. Pour accéder à Internet, les pods auxquels des groupes de sécurité sont attribués doivent être lancés sur des nœuds déployés dans un sous-réseau privé configuré avec une passerelle ou une instance NAT. Les pods avec des groupes de sécurité affectés déployés sur des sous-réseaux publics ne sont pas en mesure d'accéder à Internet.

  Si vous utilisez la version `1.11` ou ultérieure du plug-in avec `POD_SECURITY_GROUP_ENFORCING_MODE=standard`, le trafic des pods destiné à l'extérieur du VPC est traduit en adresse IP de l'interface réseau principale de l'instance. Pour ce trafic, les règles des groupes de sécurité de l'interface réseau principale sont utilisées, plutôt que celles des groupes de sécurité des pods.
- Pour utiliser la politique réseau Calico avec des pods associés à des groupes de sécurité, vous devez utiliser la version `1.11.0` ou ultérieure du plug-in CNI Amazon VPC et définir `POD_SECURITY_GROUP_ENFORCING_MODE=standard`. Dans le cas contraire, le flux de trafic à destination et en provenance des pods associés aux groupes de sécurité n'est pas soumis à l'application de la politique réseau de Calico et est limité à l'application des groupes EC2 de sécurité Amazon uniquement. Pour mettre à jour la version de votre plug-in CNI Amazon VPC, consultez the section called "CNI Amazon VPC"
- Les pods exécutés sur des EC2 nœuds Amazon qui utilisent des groupes de sécurité dans des clusters qui NodeLocal DNSCacheles utilisent ne sont pris en charge qu'avec la version `1.11.0` ou une version ultérieure du plugin Amazon VPC CNI et avec =. `POD_SECURITY_GROUP_ENFORCING_MODE standard` Pour mettre à jour la version de votre plug-in CNI Amazon VPC, consultez the section called "CNI Amazon VPC"

- Les groupes de sécurité pour les pods peuvent entraîner une latence de start-up plus élevée pour les pods à forte rotation. Cela est dû à la limitation du débit dans le contrôleur de ressources.
- L'étendue du groupe EC2 de sécurité est au niveau du POD. Pour plus d'informations, consultez la section Groupe de sécurité.

  Si vous définissez `POD_SECURITY_GROUP_ENFORCING_MODE=standard` et `AWS_VPC_K8S_CNI_EXTERNALSNAT=false`, le trafic destiné aux points de terminaison en dehors du VPC utilise les groupes de sécurité du nœud, et non ceux du pod.

Configurez le plug-in CNI Amazon VPC pour Kubernetes pour les groupes de sécurité des pods Amazon EKS

Si vous utilisez des pods avec des instances Amazon EC2, vous devez configurer le plug-in CNI Amazon VPC pour Kubernetes pour les groupes de sécurité

Si vous utilisez uniquement des pods Fargate et que vous n'avez aucun nœud Amazon EC2 dans votre cluster, consultez the section called "SecurityGroupPolicy".

1. Vérifiez la version actuelle de votre plug-in CNI Amazon VPC pour Kubernetes à l'aide de la commande suivante :

   ```
   kubectl describe daemonset aws-node --namespace kube-system | grep amazon-k8s-cni: |
    cut -d : -f 3
   ```

   L'exemple qui suit illustre un résultat.

   ```
   v1.7.6
   ```

   Si la version de votre plug-in CNI Amazon VPC pour Kubernetes est antérieure à `1.7.7`, veuillez mettre à jour le plug-in vers la version `1.7.7` ou une version ultérieure. Pour plus d'informations, consultez the section called "CNI Amazon VPC".
2. Ajoutez la politique IAM gérée AmazonEKSVPCResourceController au rôle de cluster associé à votre cluster Amazon EKS. La politique permet au rôle de gérer les interfaces réseau, leurs adresses IP privées, leur attachement et leur détachement vers et depuis les instances réseau.
   a. Récupérez le nom de votre rôle IAM de cluster et stockez-le dans une variable. Remplacez *my-cluster* par le nom de votre cluster.

```
cluster_role=$(aws eks describe-cluster --name my-cluster --query cluster.roleArn
 --output text | cut -d / -f 2)
```

b. Attachez la stratégie au rôle.

```
aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKSVPCResourceController --role-name $cluster_role
```

3. Activez le module complémentaire CNI Amazon VPC pour gérer les interfaces réseau des pods en définissant la variable `ENABLE_POD_ENI` sur `true` dans le DaemonSet `aws-node`. Une fois ce paramètre défini sur `true`, le module complémentaire crée une ressource personnalisée `cninode` pour chaque nœud du cluster. Le contrôleur de ressources VPC crée et attache une interface réseau spéciale appelée interface réseau de tronc avec la description `aws-k8s-trunk-eni`.

```
kubectl set env daemonset aws-node -n kube-system ENABLE_POD_ENI=true
```

> **Note**
>
> L'interface réseau de tronc est incluse dans le nombre maximal d'interfaces réseau prises en charge par le type d'instance. Pour obtenir la liste du nombre maximal d'interfaces réseau prises en charge par chaque type d'instance, consultez Adresses IP par interface réseau par type d'instance dans le Guide de l'utilisateur Amazon EC2. Si votre nœud a déjà le nombre maximal d'interfaces réseau standard qui lui sont attachées, le contrôleur de ressources VPC réservera un espace. Vous devrez réduire verticalement vos pods en cours d'exécution suffisamment pour que le contrôleur puisse détacher et supprimer une interface réseau standard, créer l'interface réseau trunk et la connecter à l'instance.

4. Vous pouvez voir quels nœuds ont une ressource personnalisée `CNINode` avec la commande suivante. Si le message `No resources found` est renvoyé, attendez plusieurs secondes et réessayez. L'étape précédente nécessite de redémarrer le plug-in CNI Amazon VPC pour les pods Kubernetes, ce qui prend plusieurs secondes.

```
kubectl get cninode -A
     NAME FEATURES
     ip-192-168-64-141.us-west-2.compute.internal [{"name":"SecurityGroupsForPods"}]
     ip-192-168-7-203.us-west-2.compute.internal [{"name":"SecurityGroupsForPods"}]
```

Si vous utilisez des versions CNI VPC antérieures à `1.15`, les étiquettes de nœud ont été utilisées à la place de la ressource personnalisée `CNINode`. Vous pouvez voir quels nœuds ont le label de nœud `aws-k8s-trunk-eni` défini sur `true` à l'aide de la commande suivante. Si le message `No resources found` est renvoyé, attendez plusieurs secondes et réessayez. L'étape précédente nécessite de redémarrer le plug-in CNI Amazon VPC pour les pods Kubernetes, ce qui prend plusieurs secondes.

```
kubectl get nodes -o wide -l vpc.amazonaws.com/has-trunk-attached=true
```

Une fois l'interface réseau trunk créée, les pods se voient attribuer des adresses IP secondaires à partir des interfaces réseau trunk ou standard. L'interface de tronc est automatiquement supprimée si le nœud est supprimé.

Lorsque vous déployez un groupe de sécurité pour un pod dans une étape ultérieure, le contrôleur de ressources VPC crée une interface réseau spéciale appelée interface réseau de branche avec une description `aws-k8s-branch-eni` et y associe les groupes de sécurité. Les interfaces réseau de branche sont créées en plus des interfaces réseau standard et de tronc attachées au nœud.

Si vous utilisez des sondes de vivacité ou de disponibilité, vous devez également désactiver TCP early demux, afin que le `kubelet` puisse se connecter aux pods sur les interfaces réseau de branche à l'aide de TCP. Pour désactiver le TCP early demux, exécutez la commande suivante :

```
kubectl patch daemonset aws-node -n kube-system \
  -p '{"spec": {"template": {"spec": {"initContainers": [{"env":
[{"name":"DISABLE_TCP_EARLY_DEMUX","value":"true"}],"name":"aws-vpc-cni-init"}]}}}}'
```

> **Note**
>
> Si vous utilisez la version `1.11.0` ou ultérieure du plug-in CNI Amazon VPC pour le module complémentaire Kubernetes et que vous définissez `POD_SECURITY_GROUP_ENFORCING_MODE=standard`, comme décrit à l'étape suivante, vous n'avez pas besoin d'exécuter la commande précédente.

5. Si votre cluster utilise `NodeLocal DNSCache`, ou si vous voulez utiliser la politique réseau Calico avec vos pods qui ont leurs propres groupes de sécurité, ou si vous disposez de services Kubernetes de type `NodePort` et `LoadBalancer` utilisant des cibles d'instance avec

une `externalTrafficPolicy` définie sur `Local` pour les pods que vous voulez attribuer des groupes de sécurité, vous devez utiliser la version `1.11.0` ou ultérieure du module complémentaire Plug-in CNI Amazon VPC pour Kubernetes et activer le paramètre suivant :

```
kubectl set env daemonset aws-node -n kube-system
 POD_SECURITY_GROUP_ENFORCING_MODE=standard
```

IMPORTANT : Les règles de groupe de sécurité des pods ne s'appliquent pas au trafic entre les pods ou entre les pods et les services, tels que **`kubelet`** ou **`nodeLocalDNS`**, qui se trouvent sur le même nœud. Les pods utilisant différents groupes de sécurité sur le même nœud ne peuvent pas communiquer car ils sont configurés dans des sous-réseaux différents et le routage est désactivé entre ces sous-réseaux. Le trafic sortant des pods vers des adresses en dehors du VPC est soumis à une traduction d'adresses réseau vers l'adresse IP de l'interface réseau principale de l'instance (sauf si vous avez également défini `AWS_VPC_K8S_CNI_EXTERNALSNAT=true`). Pour ce trafic, les règles des groupes de sécurité de l'interface réseau principale sont utilisées, plutôt que celles des groupes de sécurité des pods. ** Pour que ce paramètre s'applique aux pods existants, vous devez redémarrer les pods ou les nœuds sur lesquels les pods s'exécutent.

6. Pour savoir comment utiliser une stratégie de groupe de sécurité pour votre pod, consultez the section called "SecurityGroupPolicy".

Utiliser une politique de groupe de sécurité pour un pod Amazon EKS

Pour utiliser des groupes de sécurité pour les pods, vous devez disposer d'un groupe de sécurité existant. Les étapes suivantes vous montrent comment utiliser la politique de groupe de sécurité pour un pod. Sauf indication contraire, effectuez toutes les étapes à partir du même terminal, car les étapes suivantes utilisent des variables qui ne persistent pas d'un terminal à l'autre.

Si vous possédez un Pod avec EC2 des instances Amazon, vous devez configurer le plugin avant d'utiliser cette procédure. Pour de plus amples informations, veuillez consulter the section called "Configuration".

1. Créez un espace de noms Kubernetes vers lequel déployer les ressources . Vous pouvez le *`my-namespace`* remplacer par le nom de l'espace de noms que vous souhaitez utiliser.

```
kubectl create namespace my-namespace
```

2. Déployez une politique `SecurityGroupPolicy` Amazon EKS sur votre cluster.

a. Copiez les contenus suivants sur votre appareil. Vous pouvez les *podSelector* remplacer par `serviceAccountSelector` si vous préférez sélectionner des pods en fonction des étiquettes des comptes de service. Vous devez spécifier un sélecteur ou l'autre. Un `podSelector` vide (exemple : `podSelector: {}`) sélectionne tous les pods de l'espace de noms. Vous pouvez *my-role* modifier le nom de votre rôle. Un `serviceAccountSelector` vide sélectionne tous les comptes de service dans l'espace de noms. Vous pouvez le *my-security-group-policy* remplacer par un nom pour votre nom `SecurityGroupPolicy` et *my-namespace* par l'espace de noms `SecurityGroupPolicy` dans lequel vous souhaitez le créer.

   Vous devez le *my_pod_security_group_id* remplacer par l'ID d'un groupe de sécurité existant. Si vous ne disposez pas d'un groupe de sécurité existant, vous devez en créer un. Pour plus d'informations, consultez les groupes EC2 de sécurité Amazon pour les instances Linux dans le guide de EC2 l'utilisateur Amazon. Vous pouvez spécifier un groupe IDs de sécurité de 1 à 5. Si vous spécifiez plusieurs ID, la combinaison de toutes les règles de tous les groupes de sécurité s'applique aux pods sélectionnés.

```
cat >my-security-group-policy.yaml <<EOF
apiVersion: vpcresources.k8s.aws/v1beta1
kind: SecurityGroupPolicy
metadata:
  name: my-security-group-policy
  namespace: my-namespace
spec:
  podSelector:
    matchLabels:
      role: my-role
  securityGroups:
    groupIds:
      - my_pod_security_group_id
EOF
```

   > **Important**
   >
   > Le ou les groupes de sécurité que vous spécifiez pour vos pods doivent répondre aux critères suivants :
   >
   > - Ils doivent exister. S'ils n'existent pas, lorsque vous déployez un pod qui correspond au sélecteur, votre pod reste bloqué dans le processus de création. Si vous décrivez le pod, vous verrez un message d'erreur similaire à celui-ci : `An error occurred (InvalidSecurityGroupID.NotFound) when calling`

```
the CreateNetworkInterface operation: The securityGroup ID
'sg-05b1d815d1EXAMPLE' does not exist.
```

- Ils doivent autoriser les communications entrantes provenant du groupe de sécurité appliqué à vos nœuds (pour `kubelet`) sur tous les ports pour lesquels vous avez configuré des sondes.
- Ils doivent autoriser les communications sortantes sur les ports TCP et UDP 53 vers un groupe de sécurité attribué aux pods (ou aux nœuds sur lesquels les pods s'exécutent) exécutant CoreDNS. Le groupe de sécurité de vos pods CoreDNS doit autoriser le trafic entrant sur les ports TCP et UDP 53 provenant du groupe de sécurité que vous spécifiez.
- Ils doivent disposer des règles entrantes et sortantes nécessaires pour communiquer avec les autres pods avec lesquels ils doivent communiquer.
- Ils doivent disposer de règles autorisant les pods à communiquer avec le plan de contrôle Kubernetes si vous utilisez le groupe de sécurité avec Fargate. La façon la plus simple de le faire est de spécifier le groupe de sécurité de cluster comme l'un des groupes de sécurité.

Les politiques du groupe de sécurité s'appliquent uniquement aux pods nouvellement planifiés. Elles n'affectent pas les pods en cours d'exécution.

b. Déployez la politique.

```
kubectl apply -f my-security-group-policy.yaml
```

3. Déployez un exemple d'application avec une étiquette correspondant à la valeur *my-role* pour *podSelector* que vous avez spécifié à l'étape précédente.

a. Copiez les contenus suivants sur votre appareil. Remplacez les exemples de valeur par les vôtres, puis exécutez la commande modifiée. Si vous remplacez*my-role*, assurez-vous qu'elle est identique à la valeur que vous avez spécifiée pour le sélecteur à l'étape précédente.

```
cat >sample-application.yaml <<EOF
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-deployment
  namespace: my-namespace
  labels:
    app: my-app
```

```
spec:
  replicas: 4
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
        role: my-role
    spec:
      terminationGracePeriodSeconds: 120
      containers:
      - name: nginx
        image: public.ecr.aws/nginx/nginx:1.23
        ports:
        - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  name: my-app
  namespace: my-namespace
  labels:
    app: my-app
spec:
  selector:
    app: my-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
EOF
```

b. Pour déployer l'application, exécutez la commande suivante. Lorsque vous déployez l'application, le plug-in CNI Amazon VPC pour Kubernetes fait correspondre l'étiquette `role` et les groupes de sécurité que vous avez spécifiés à l'étape précédente sont appliqués au pod.

```
kubectl apply -f sample-application.yaml
```

4. Affichez les pods déployés avec l'application exemple. Pour le reste de ce sujet, ce terminal est appelé `TerminalA`.

```
kubectl get pods -n my-namespace -o wide
```

L'exemple qui suit illustre un résultat.

```
NAME                              READY   STATUS    RESTARTS   AGE     IP
  NODE                                                NOMINATED NODE   READINESS GATES
my-deployment-5df6f7687b-4fbjm   1/1     Running   0          7m51s   192.168.53.48
  ip-192-168-33-28.region-code.compute.internal   <none>           <none>
my-deployment-5df6f7687b-j9fl4   1/1     Running   0          7m51s   192.168.70.145
  ip-192-168-92-33.region-code.compute.internal   <none>           <none>
my-deployment-5df6f7687b-rjxcz   1/1     Running   0          7m51s   192.168.73.207
  ip-192-168-92-33.region-code.compute.internal   <none>           <none>
my-deployment-5df6f7687b-zmb42   1/1     Running   0          7m51s   192.168.63.27
  ip-192-168-33-28.region-code.compute.internal   <none>           <none>
```

> **Note**
>
> Suivez ces conseils si des pods sont bloqués.
>
> - Si des pods sont bloqués dans l'état `Waiting`, exécutez `kubectl describe pod` *`my-deployment-xxxxxxxxxx-xxxxx`* `-n` *`my-namespace`* . Si vous voyez `Insufficient permissions: Unable to create Elastic Network Interface.`, vérifiez que vous avez ajouté la politique IAM au rôle de cluster IAM dans une étape précédente.
> - Si des pods sont bloqués dans l'état `Pending`, vérifiez que votre type d'instance de nœud est répertorié dans limits.go et que le produit du nombre maximal d'interfaces réseau de branche prises en charge par le type d'instance multiplié par le nombre de nœuds dans votre groupe de nœuds n'a pas déjà été atteint. Par exemple, une instance `m5.large` prend en charge neuf interfaces réseau de branche. Si votre groupe de nœuds comporte cinq nœuds, un maximum de 45 interfaces réseau de branche peut être créé pour le groupe de nœuds. Le 46e pod que vous essayez de déployer restera dans l'état `Pending` jusqu'à ce qu'un autre pod associé à des groupes de sécurité soit supprimé.

Si vous exécutez `kubectl describe pod` *`my-deployment-xxxxxxxxxx-xxxxx`* `-n` *`my-namespace`* et recevez un message similaire au message suivant, il peut être ignoré en toute sécurité. Ce message peut apparaître lorsque le plug-in CNI Amazon VPC pour Kubernetes

tente de configurer le réseau hôte et échoue pendant la création de l'interface réseau. Le plugin journalise cet événement jusqu'à ce que l'interface réseau soit créée.

```
Failed to create Pod sandbox: rpc error: code = Unknown desc = failed to set up
 sandbox container "e24268322e55c8185721f52df6493684f6c2c3bf4fd59c9c121fd4cdc894579f"
 network for Pod "my-deployment-5df6f7687b-4fbjm": networkPlugin
cni failed to set up Pod "my-deployment-5df6f7687b-4fbjm-c89wx_my-namespace" network:
 add cmd: failed to assign an IP address to container
```

Vous ne pouvez pas dépasser le nombre maximal de pods pouvant être exécutés sur le type d'instance. Pour obtenir la liste du nombre maximal de pods que vous pouvez exécuter sur chaque type d'instance, consultez eni-max-pods.txt on GitHub. Lorsque vous supprimez un pod associé à des groupes de sécurité ou que vous supprimez le nœud sur lequel le pod est exécuté, le contrôleur de ressources VPC supprime l'interface réseau de la branche. Si vous supprimez un cluster avec des pods utilisant des pods pour les groupes de sécurité, le contrôleur ne supprime pas les interfaces réseau de la branche. Vous devrez donc les supprimer vous-même. Pour plus d'informations sur la suppression d'interfaces réseau, consultez Supprimer une interface réseau dans le guide de EC2 l'utilisateur Amazon.

5. Dans un terminal séparé, connectez-vous à l'un des pods. Pour le reste de ce sujet, ce terminal est appelé `TerminalB`. *`5df6f7687b-4fbjm`*Remplacez-le par l'ID de l'un des pods renvoyés dans votre sortie de l'étape précédente.

```
kubectl exec -it -n my-namespace my-deployment-5df6f7687b-4fbjm -- /bin/bash
```

6. À partir du shell dans `TerminalB`, confirmez que l'exemple d'application fonctionne.

```
curl my-app
```

L'exemple qui suit illustre un résultat.

```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
[...]
```

Vous avez reçu cette sortie car tous les pods exécutant l'application sont associés au groupe de sécurité que vous avez créé. Ce groupe contient une règle qui autorise tout le trafic entre tous

les pods associés au groupe de sécurité. Le trafic DNS est autorisé à partir de ce groupe de sécurité vers le groupe de sécurité du cluster associé à vos nœuds. Les nœuds exécutent les pods CoreDNS, vers lesquels vos pods ont effectué une recherche de nom.

7. À partir du `TerminalA`, supprimez les règles de groupe de sécurité qui autorisent la communication DNS au groupe de sécurité du cluster de votre groupe de sécurité. Si vous n'avez pas ajouté les règles DNS au groupe de sécurité du cluster lors d'une étape précédente, *`$my_cluster_security_group_id`* remplacez-les par l'ID du groupe de sécurité dans lequel vous avez créé les règles.

```
aws ec2 revoke-security-group-ingress --group-id $my_cluster_security_group_id --
security-group-rule-ids $my_tcp_rule_id
aws ec2 revoke-security-group-ingress --group-id $my_cluster_security_group_id --
security-group-rule-ids $my_udp_rule_id
```

8. À partir du `TerminalB`, essayez de nouveau d'accéder à l'application.

```
curl my-app
```

L'exemple qui suit illustre un résultat.

```
curl: (6) Could not resolve host: my-app
```

La tentative échoue car le pod ne peut plus accéder aux pods CoreDNS, qui sont associés au groupe de sécurité du cluster. Le groupe de sécurité du cluster ne dispose plus des règles de groupe de sécurité qui autorisent la communication DNS à partir du groupe de sécurité associé à votre pod.

Si vous essayez d'accéder à l'application à l'aide des adresses IP renvoyées pour l'un des pods lors d'une étape précédente, vous recevez toujours une réponse, car tous les ports sont autorisés entre les pods associés au groupe de sécurité et aucune recherche de nom n'est requise.

9. Une fois votre expérience terminée, vous pouvez supprimer l'exemple de politique de groupe de sécurité, l'application et le groupe de sécurité que vous avez créés. Exécutez les commandes suivantes à partir du `TerminalA`.

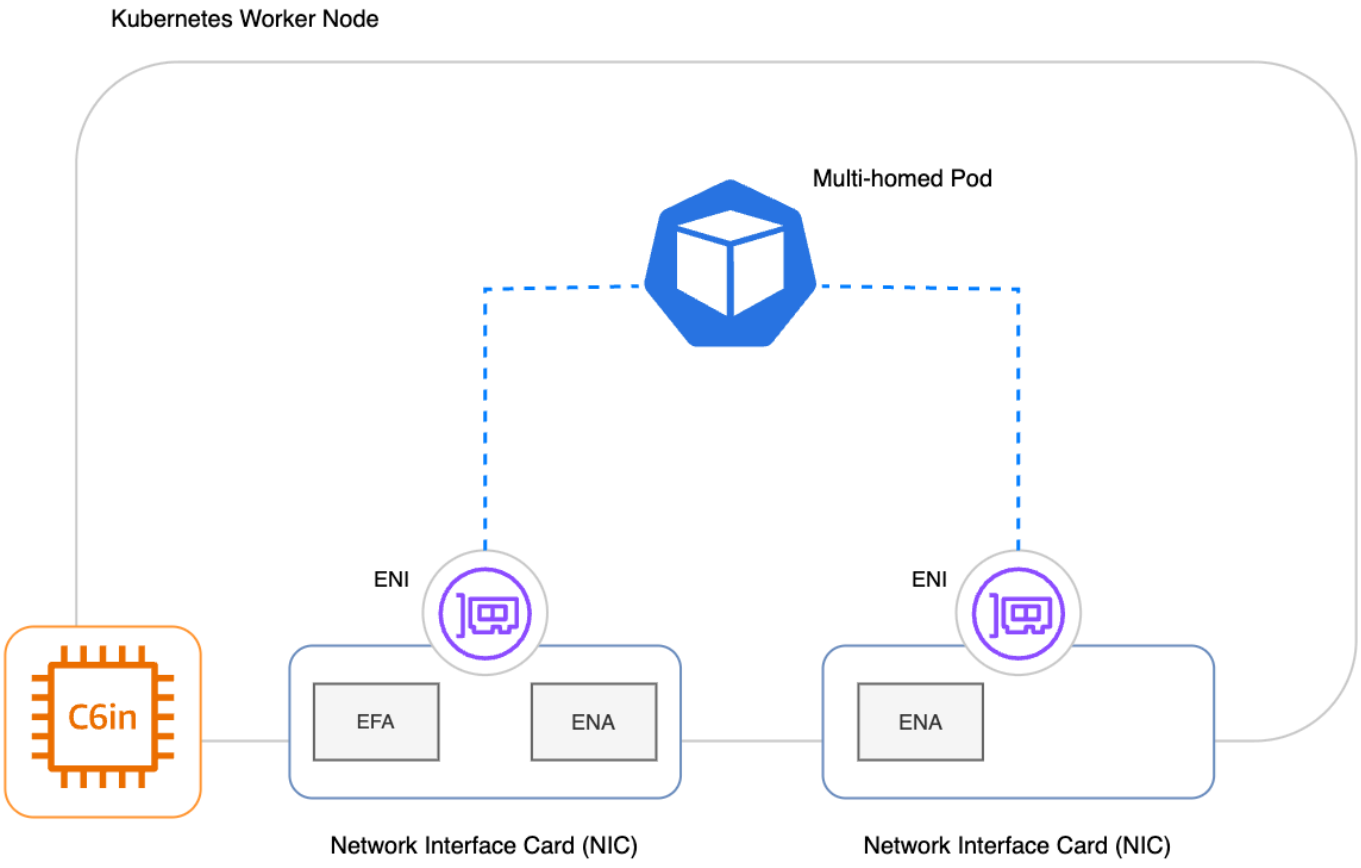
```
kubectl delete namespace my-namespace
aws ec2 revoke-security-group-ingress --group-id $my_pod_security_group_id --
security-group-rule-ids $my_inbound_self_rule_id
wait
```

```
sleep 45s
aws ec2 delete-security-group --group-id $my_pod_security_group_id
```

Connecter plusieurs interfaces réseau à des pods

Par défaut, le plug-in CNI Amazon VPC attribue une adresse IP à chaque pod. Cette adresse IP est associée à une interface réseau Elastic qui gère tout le trafic entrant et sortant du pod. Pour augmenter la bande passante et le débit de paquets par seconde, vous pouvez utiliser la fonctionnalité multi-NIC de CNI VPC pour configurer un pod multi-homed. Un pod multi-homed est un pod Kubernetes unique qui utilise plusieurs interfaces réseau (et plusieurs adresses IP). En exécutant un pod multi-homed, vous pouvez répartir le trafic de son application sur plusieurs interfaces réseau à l'aide de connexions simultanées. Cela est particulièrement utile pour les cas d'utilisation de l'intelligence artificielle (IA), de machine learning (ML) et du calcul haute performance (HPC).

Le schéma suivant montre un module multi-hébergement exécuté sur un nœud de travail avec plusieurs cartes d'interface réseau (NICs) utilisées.



Contexte

Sur Amazon EC2, une interface réseau élastique est un composant réseau logique d'un VPC qui représente une carte réseau virtuelle. Pour de nombreux types d' EC2 instances, les interfaces réseau partagent une seule carte d'interface réseau (NIC) matérielle. Cette carte réseau unique dispose d'une bande passante maximale et d'un débit maximal de paquets par seconde.

Si la fonctionnalité multi-NIC est activée, le CNI VPC n'attribue pas d'adresses IP en masse, comme il le fait par défaut. Au lieu de cela, le CNI VPC attribue une adresse IP à une interface réseau sur chaque carte réseau à la demande lorsqu'un nouveau pod démarre. Ce comportement réduit le taux d'épuisement des adresses IP, qui est augmenté par l'utilisation de pods multi-homed. Étant donné que le CNI VPC attribue des adresses IP à la demande, le démarrage des pods peut prendre plus de temps sur les instances où la fonctionnalité multi-NIC est activée.

Considérations

- Veuillez vous assurer que votre cluster Kubernetes exécute la version `1.20.0` ou ultérieure du CNI VPC. La fonctionnalité multi-NIC n'est disponible que dans la version `1.20.0` ou ultérieure du CNI VPC.
- Activez la variable d'environnement `ENABLE_MULTI_NIC` dans le plug-in CNI Amazon VPC. Vous pouvez exécuter la commande suivante pour définir la variable et démarrer le déploiement du DaemonSet.
  - `kubectl set env daemonset aws-node -n kube-system ENABLE_MULTI_NIC=true`
- Assurez-vous de créer des nœuds de travail dotés de plusieurs cartes d'interface réseau (NICs). Pour obtenir la liste des EC2 instances dotées de plusieurs cartes d'interface réseau, consultez la section Cartes réseau du guide de EC2 l'utilisateur Amazon.
- Si la fonctionnalité multi-NIC est activée, le CNI VPC n'attribue pas d'adresses IP en masse, comme il le fait par défaut. Étant donné que le CNI VPC attribue des adresses IP à la demande, le démarrage des pods peut prendre plus de temps sur les instances où la fonctionnalité multi-NIC est activée. Pour de plus amples informations, veuillez consulter la section précédente the section called "Contexte".
- Lorsque la fonctionnalité multi-NIC est activée, les pods ne disposent pas de plusieurs interfaces réseau par défaut. Vous devez configurer chaque charge de travail pour utiliser la fonctionnalité multi-NIC. Ajoutez l'annotation `k8s.amazonaws.com/nicConfig: multi-nic-attachment` aux charges de travail qui doivent disposer de plusieurs interfaces réseau.

Considérations relatives à **IPv6**

- Politique IAM personnalisée : pour les clusters IPv6, créez et utilisez la politique IAM personnalisée suivante pour le CNI VPC. Cette politique est spécifique à la fonctionnalité multi-NIC. Pour plus d'informations générales sur l'utilisation du CNI VPC avec les clusters IPv6, consultez the section called "IPv6".

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AmazonEKSCNIPolicyIPv6MultiNIC",
            "Effect": "Allow",
            "Action": [
                "ec2:CreateNetworkInterface",
                "ec2:DescribeInstances",
                "ec2:AssignIpv6Addresses",
                "ec2:DetachNetworkInterface",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeTags",
                "ec2:ModifyNetworkInterfaceAttribute",
                "ec2:DeleteNetworkInterface",
                "ec2:DescribeInstanceTypes",
                "ec2:UnassignIpv6Addresses",
                "ec2:AttachNetworkInterface",
                "ec2:DescribeSubnets"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AmazonEKSCNIPolicyENITagIPv6MultiNIC",
            "Effect": "Allow",
            "Action": "ec2:CreateTags",
            "Resource": "arn:aws:ec2:*:*:network-interface/*"
        }
    ]
}
```

- Mécanisme de transition **IPv6** non disponible : si vous utilisez la fonctionnalité multi-NIC, le CNI VPC n'attribue pas d'adresse IPv4 aux pods d'un cluster IPv6. Sinon, le CNI Amazon VPC attribue une adresse IPv4 locale à chaque pod afin que celui-ci puisse communiquer avec des ressources IPv4 externes dans un autre Amazon VPC ou sur Internet.

Usage

Une fois la fonctionnalité multi-NIC activée dans le CNI Amazon VPC et les pods `aws-node` redémarrés, vous pouvez configurer chaque charge de travail pour qu'elle soit multi-homed. Voici un exemple de configuration YAML avec l'annotation requise :

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: orders-deployment
  namespace: ecommerce
  labels:
    app: orders
spec:
  replicas: 3
  selector:
    matchLabels:
      app: orders
  template:
    metadata:
      annotations:
         k8s.amazonaws.com/nicConfig: multi-nic-attachment
      labels:
        app: orders
    spec:
...
```

Questions fréquentes (FAQ)

1. Qu'est-ce qu'une carte d'interface réseau (NIC) ?

Une carte d'interface réseau (NIC), également appelée simplement carte réseau, est un périphérique physique qui permet la connectivité réseau pour le matériel informatique cloud sous-jacent. Dans EC2 les serveurs modernes, cela fait référence à la carte réseau Nitro. Une interface réseau Elastic (ENI) est une représentation virtuelle de cette carte réseau sous-jacente.

Certains types d' EC2 instances en ont plusieurs NICs pour améliorer les performances en termes de bande passante et de débit de paquets. Dans de tels cas, vous pouvez attribuer des cartes réseau secondaires ENIs aux cartes réseau supplémentaires. Par exemple, l'ENI n° 1 peut fonctionner comme interface pour la carte NIC connectée à l'index de carte réseau 0, tandis que l'ENI n° 2 peut fonctionner comme interface pour la carte NIC connectée à un index de carte réseau distinct.

2. Qu'est-ce qu'un pod multi-homed ?

Un pod multi-homed est un pod Kubernetes unique doté de plusieurs interfaces réseau (et donc de plusieurs adresses IP). Chaque interface réseau du pod est associée à une interface réseau élastique (ENI), et il s' ENIs agit de représentations logiques distinctes NICs sur le nœud de travail sous-jacent. Grâce à ses multiples interfaces réseau, un pod multi-homed dispose d'une capacité de transfert de données supplémentaire, ce qui augmente également son débit de transfert de données.

> ⚠ Important
>
> Le VPC CNI peut uniquement configurer des pods multi-hôtes sur des types d'instances qui en ont plusieurs. NICs

3. Pourquoi utiliser cette fonctionnalité ?

Si vous devez adapter les performances réseau de vos charges de travail basées sur Kubernetes, vous pouvez utiliser la fonctionnalité multi-cartes réseau pour exécuter des pods multi-hôtes qui s'interfacent avec tous les composants sous-jacents NICs auxquels un périphérique ENA est connecté. L'utilisation de cartes réseau supplémentaires augmente la capacité de bande passante et les performances en termes de débit de paquets dans vos applications en répartissant le trafic des applications sur plusieurs connexions simultanées. Cela est particulièrement utile pour les cas d'utilisation de l'intelligence artificielle (IA), de machine learning (ML) et du calcul haute performance (HPC).

4. Comment utiliser cette fonctionnalité ?

1. Tout d'abord, vous devez vous assurer que votre cluster Kubernetes utilise la version 1.20 ou ultérieure de CNI VPC. Pour connaître les étapes de mise à jour de CNI VPC en tant que module complémentaire EKS, consultez the section called "Mise à jour (module complémentaire EKS)".
2. Ensuite, vous devez activer la prise en charge multi-NIC dans le CNI Amazon VPC à l'aide de la variable d'environnement `ENABLE_MULTI_NIC`.
3. Enfin, vous devez vous assurer de créer et de rejoindre des nœuds disposant de plusieurs cartes réseau. Pour obtenir la liste des types d' EC2 instances dotés de plusieurs cartes réseau, consultez la section Cartes réseau du guide de EC2 l'utilisateur Amazon.
4. Enfin, vous devez configurer chaque charge de travail pour qu'elle utilise soit plusieurs interfaces réseau (pods multi-homed), soit une seule interface réseau.

5. Comment configurer mes charges de travail pour en utiliser plusieurs NICs sur un nœud de travail pris en charge ?

Pour utiliser des pods multi-homed, vous devez ajouter l'annotation suivante : `k8s.amazonaws.com/nicConfig: multi-nic-attachment`. Cela associera un ENI de chaque carte réseau de l'instance sous-jacente au pod (un ou plusieurs mappages entre un pod et le NICs).

Si cette annotation est manquante, le CNI VPC suppose que votre pod ne nécessite qu'une seule interface réseau et lui attribue une adresse IP provenant d'une ENI sur n'importe quelle carte réseau disponible.

6. Quels adaptateurs d'interface réseau sont pris en charge avec cette fonctionnalité ?

Vous pouvez utiliser n'importe quel adaptateur d'interface réseau si vous disposez d'au moins un ENA connecté à la carte réseau sous-jacente pour le trafic IP. Pour plus d'informations sur l'ENA, consultez Elastic Network Adapter (ENA) dans le guide de EC2 l'utilisateur Amazon.

Configurations de périphériques réseau prises en charge :

- les interfaces ENA fournissent toutes les fonctionnalités de réseau et de routage IP traditionnelles requises pour prendre en charge le réseau IP d'un VPC. Pour plus d'informations, consultez Activer la mise en réseau améliorée avec ENA sur vos EC2 instances.
- Les interfaces EFA (EFA avec ENA) fournissent à la fois le périphérique ENA pour la mise en réseau IP et le périphérique EFA pour une communication à faible latence et haut débit.

> **Important**
>
> Si une carte réseau ne dispose que d'un adaptateur EFA uniquement attaché, le CNI VPC l'ignorera lors de la fourniture d'une connectivité réseau pour un pod multi-hébergé. Toutefois, si vous combinez un adaptateur EFA uniquement avec un adaptateur ENA sur une carte réseau, le VPC CNI s'en chargera également sur cet appareil. ENIs Pour utiliser des interfaces EFA uniquement avec des clusters EKS, consultez the section called "Configurer des clusters d'entraînement avec EFA".

7. Puis-je voir si un nœud de mon cluster prend en charge ENA ?

Oui, vous pouvez utiliser la AWS CLI ou EC2 l'API pour récupérer des informations réseau concernant une EC2 instance de votre cluster. Cela vous permettra de savoir si l'instance prend en

charge ENA. Dans l'exemple suivant, remplacez `<your-instance-id>` par l'ID d' EC2 instance d'un nœud.

AWS Exemple de CLI :

```
aws ec2 describe-instances --instance-ids <your-instance-id> --query
 "Reservations[].Instances[].EnaSupport"
```

Exemple de sortie :

```
[ true ]
```

8. Puis-je voir les différentes adresses IP associées à un pod ?

Non, pas facilement. Cependant, vous pouvez utiliser `nsenter` à partir du nœud pour exécuter des outils réseau courants tels que `ip route show` et voir les adresses IP et les interfaces supplémentaires.

9. Puis-je contrôler le nombre d'interfaces réseau pour mes pods ?

Non Lorsque votre charge de travail est configurée pour en utiliser plusieurs NICs sur une instance prise en charge, un seul pod possède automatiquement une adresse IP provenant de chaque carte réseau de l'instance. Sinon, les pods à connexion unique disposeront d'une interface réseau connectée à une carte réseau sur l'instance.

> **Important**
>
> Les cartes réseau auxquelles seul un périphérique EFA uniquement est connecté sont ignorées par le CNI VPC.

10. Puis-je configurer mes pods pour qu'ils utilisent une carte réseau spécifique ?

Non, cela n'est pas pris en charge. Si un pod dispose de l'annotation appropriée, le CNI VPC le configure automatiquement pour utiliser toutes les cartes réseau avec un adaptateur ENA sur le composant master.

11. Cette fonctionnalité est-elle compatible avec les autres fonctionnalités réseau du CNI VPC ?

Oui, la fonctionnalité multi-cartes réseau du CNI VPC est compatible avec le réseau personnalisé et la détection améliorée des sous-réseaux. Cependant, les pods multi-homed n'utilisent pas les sous-

réseaux personnalisés ni les groupes de sécurité. Au lieu de cela, le CNI VPC attribue des adresses IP et des interfaces réseau aux pods multi-homed avec la même configuration de sous-réseau et de groupe de sécurité que le nœud. Pour plus d'informations sur la mise en réseau personnalisée, consultez the section called "Mise en réseau personnalisée".

La fonctionnalité multi-NIC du CNI VPC ne fonctionne pas avec les groupes de sécurité pour les pods et ne peut pas être combinée avec ceux-ci.

12. Puis-je utiliser des politiques réseau avec cette fonctionnalité ?

Oui, vous pouvez utiliser les politiques réseau Kubernetes avec plusieurs cartes réseau.
Les politiques réseau Kubernetes limitent le trafic réseau vers et depuis vos pods. Pour plus d'informations sur l'application des politiques réseau avec le CNI VPC EKS, consultez the section called "Politiques Kubernetes".

13. La prise en charge de plusieurs cartes réseau est-elle activée dans le mode automatique EKS ?

Plusieurs cartes réseau ne sont pas prises en charge pour les clusters du mode automatique EKS.

Plug-ins CNI alternatifs pour les clusters Amazon EKS

Le plug-in Amazon VPC CNI pour Kubernetes est le seul plug-in CNI pris en charge par Amazon EKS avec des nœuds Amazon EC2. Amazon EKS prend en charge les fonctionnalités essentielles de Cilium et Calico pour les nœuds hybrides Amazon EKS. Amazon EKS exécute Kubernetes en amont, ce qui signifie que vous pouvez installer des plug-ins CNI alternatifs compatibles sur les nœuds Amazon EC2 de votre cluster. Si votre cluster utilise des nœuds Fargate, le plug-in Amazon VPC CNI pour Kubernetes est déjà présent sur les nœuds Fargate. C'est le seul plug-in CNI que vous pouvez utiliser avec les nœuds Fargate. Une tentative d'installation d'un autre plugin CNI sur les nœuds Fargate a échoué.

Si vous prévoyez d'utiliser un autre plugin CNI sur des nœuds Amazon EC2, nous vous recommandons d'obtenir un support commercial pour le plugin ou de disposer de l'expertise interne pour dépanner et contribuer aux corrections du projet de plugin CNI.

Amazon EKS entretient des relations avec un réseau de partenaires qui offrent un support pour d'autres plugins CNI compatibles. Pour plus d'informations sur les versions, les qualifications et les tests effectués, consultez la documentation des partenaires suivante.

Partenaire	Produit (langue française non garantie)	Documentation
Tigera	Calico	Instructions d'installation
Isovalent	Cilium	Instructions d'installation
Juniper	Réseau Contrail natif cloud (CN2)	Instructions d'installation
VMware	Antrea	Instructions d'installation

Amazon EKS vise à vous offrir un large choix d'options pour couvrir tous les cas d'utilisation.

Plug-ins alternatifs compatibles avec les politiques réseau

Calico est une solution largement adoptée pour la mise en réseau et la sécurité des conteneurs. L'utilisation de Calico avec EKS offre une application complète des politiques réseau pour vos clusters EKS. Vous pouvez également choisir d'utiliser le réseau de Calico, ce qui permet de conserver les adresses IP provenant de votre VPC sous-jacent. Calico Cloud enrichit les fonctionnalités de Calico Open Source en offrant des capacités avancées de sécurité et d'observabilité.

Le flux de trafic à destination et en provenance des pods avec des groupes de sécurité associés n'est pas soumis à l'application de la politique réseau Calico et est limité à l'application des groupes de sécurité Amazon VPC uniquement.

Si vous utilisez l'application des politiques réseau de Calico, il est recommandé de définir la variable d'environnement `ANNOTATE_POD_IP` sur `true` afin d'éviter un problème connu avec Kubernetes. Pour utiliser cette fonctionnalité, vous devez ajouter l'autorisation `patch` pour les pods au ClusterRole `aws-node`. Notez que l'ajout d'autorisations « patch » au DaemonSet `aws-node` augmente le périmètre de sécurité du plug-in. Pour plus d'informations, consultez ANNOTATE_POD_IP dans le référentiel CNI VPC sur GitHub.

Considérations relatives au mode automatique Amazon EKS

Le mode automatique Amazon EKS ne prend pas en charge les plug-ins CNI alternatifs ni les plug-ins de politique réseau. Pour de plus amples informations, consultez Mode automatique EKS.

Attacher plusieurs interfaces réseau à des pods avec Multus

Le CNI Multus est un plug-in d'interface réseau de conteneur (CNI) pour Amazon EKS qui permet d'attacher plusieurs interfaces réseau à un pod. Pour plus d'informations, consultez la documentation Multus CNI sur GitHub.

Dans Amazon EKS, chaque pod dispose d'une interface réseau attribuée par le plug-in CNI Amazon VPC. Avec Multus, vous pouvez créer un pod multi-homed disposant de plusieurs interfaces. Ceci est accompli par Multus agissant comme un « méta-plugin » ; un plugin CNI qui peut appeler plusieurs autres plugins CNI. La prise en charge de AWS pour Multus est configurée avec le plugin CNI Amazon VPC comme plugin délégué par défaut.

- Amazon EKS ne créera ni ne publiera de plug-ins CNI SR-IOV (Single Root I/O Virtualization) et DPDK (Data Plane Development Kit). Toutefois, vous pouvez accélérer les paquets en vous connectant directement à Amazon EC2 Elastic Network Adapters (ENA) via le périphérique hôte géré par Multus et les plugins `ipvlan`.
- Amazon EKS prend en charge Multus, qui fournit un processus générique qui permet un chaînage simple de plugins CNI supplémentaires. Multus et le processus de chaînage sont pris en charge, mais AWS ne fournira pas d'assistance pour tous les plugins CNI compatibles pouvant être chaînés, ni pour les problèmes pouvant survenir dans ces plugins CNI qui ne sont pas liés à la configuration du chaînage.
- Amazon EKS fournit une assistance et une gestion du cycle de vie pour le plug-in Multus, mais n'est pas responsable des adresses IP ou de la gestion supplémentaire associée aux interfaces réseau supplémentaires. L'adresse IP et la gestion de l'interface réseau par défaut utilisant le plugin CNI Amazon VPC restent inchangées.
- Seul le plugin CNI Amazon VPC est officiellement pris en charge en tant que plugin délégué par défaut. Vous devez modifier le manifeste d'installation de Multus publié pour reconfigurer le plugin délégué par défaut vers un autre CNI si vous choisissez de ne pas utiliser le plugin CNI Amazon VPC pour la mise en réseau principale.
- Multus n'est pris en charge que lorsque vous utilisez le CNI d'Amazon VPC comme CNI principal. Nous ne prenons pas en charge le CNI d'Amazon VPC lorsqu'il est utilisée pour des interfaces d'ordre supérieur, secondaires ou autres.
- Pour empêcher le plug-in CNI Amazon VPC de tenter de gérer les interfaces réseau supplémentaires attribuées aux pods, balisez l'interface réseau avec la balise suivante :

  key

  : `node.k8s.amazonaws.com/no_manage`

valeur

: `true`

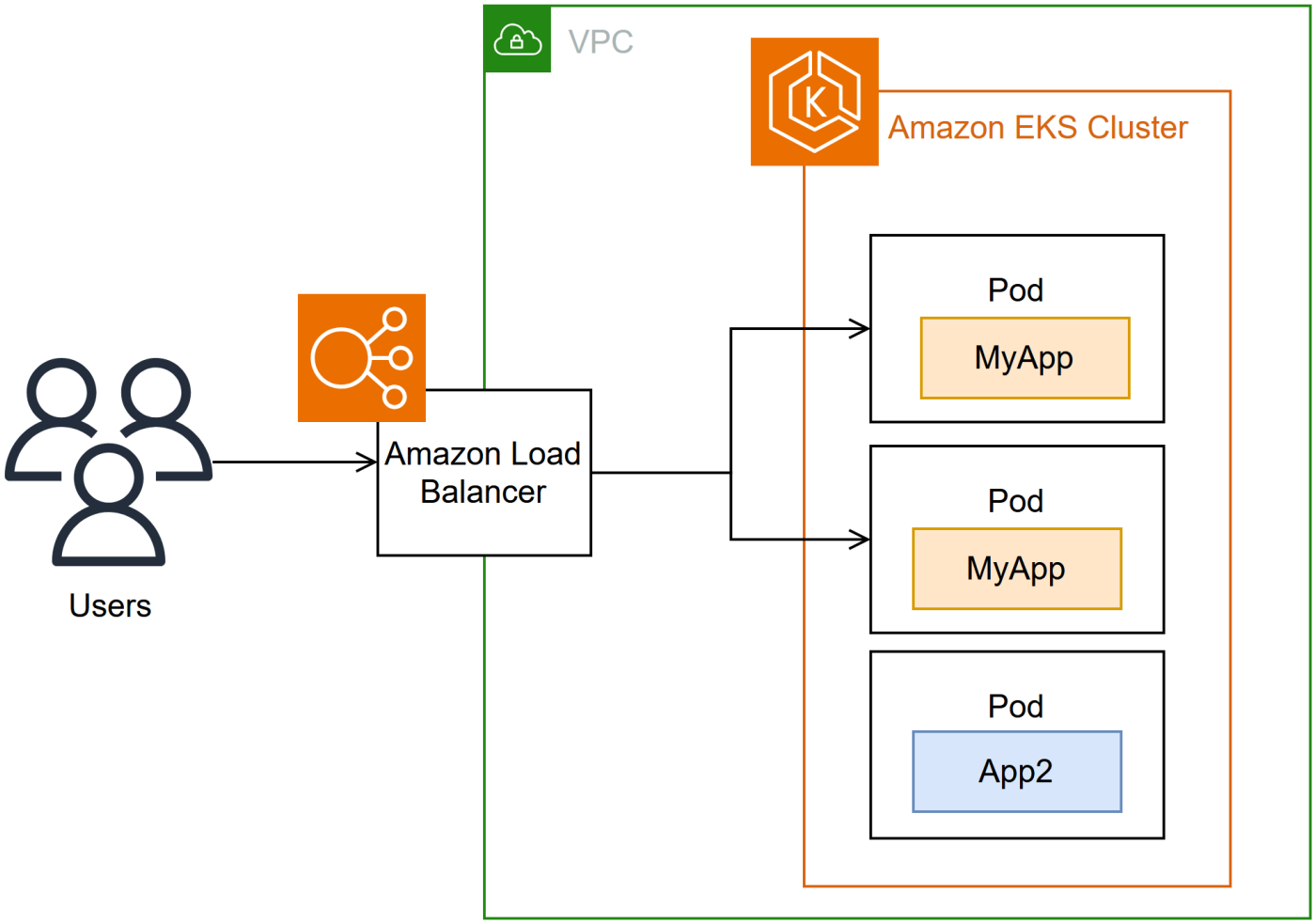
- Multus est compatible avec les politiques réseau, mais la politique doit être enrichie pour inclure les ports et les adresses IP qui peuvent faire partie des interfaces réseau supplémentaires attachées aux pods.

Pour une démonstration de la mise en œuvre, consultez le Guide d'installation de Multus sur GitHub.

Acheminez le trafic Internet avec le AWS Load Balancer Controller

> **Tip**
>
> Inscrivez-vous aux prochains ateliers Amazon EKS.

Le AWS Load Balancer Controller gère les AWS Elastic Load Balancers pour un cluster Kubernetes. Vous pouvez utiliser le contrôleur pour exposer les applications de votre cluster à Internet. Le contrôleur fournit des équilibreurs de AWS charge qui pointent vers les ressources de service ou d'entrée du cluster. En d'autres termes, le contrôleur crée une adresse IP ou un nom DNS unique qui achemine le trafic vers plusieurs pods dans votre cluster.



Le contrôleur surveille les ressources Service ou Ingress Kubernetes. En réponse, il crée les ressources AWS Elastic Load Balancing appropriées. Vous pouvez configurer le comportement spécifique des équilibreurs de charge en ajoutant des annotations aux ressources Kubernetes. Par exemple, vous pouvez associer des groupes AWS de sécurité aux équilibreurs de charge à l'aide d'annotations.

Le contrôleur fournit les ressources suivantes :

Kubernetes **`Ingress`**

Le LBC crée un équilibreur de charge AWS Application Load Balancer (ALB) lorsque vous créez une ressource `Ingress` Kubernetes. Consultez la liste des annotations applicables à une ressource Ingress.

Service Kubernetes de type **`LoadBalancer`**

Le LBC crée un équilibreur de charge AWS Network Load Balancer (NLB) lorsque vous créez un service Kubernetes de type `LoadBalancer`. Consultez la liste des annotations applicables à une ressource Service.

Historiquement, l'équilibreur de charge Network Load Balancer Kubernetes était utilisé pour des cibles de type instance, tandis que le LBC était utilisé pour des cibles de type IP. Avec la version AWS Load Balancer Controller `2.3.0` ou ultérieure, vous pouvez créer en NLBs utilisant l'un ou l'autre type de cible. Pour plus d'informations sur les types de cibles NLB, consultez Type de cible dans le guide de l'utilisateur de Network Load Balancer.

Le contrôleur est un projet open source géré sur GitHub.

Avant de déployer le contrôleur, nous vous recommandons de consulter les prérequis et points à considérer dans Routage du trafic des applications et du trafic HTTP à l'aide des équilibreurs de charge d'applications Application Load Balancer et the section called "Répartition de charge réseau". Dans ces rubriques, vous allez déployer un exemple d'application qui inclut un équilibreur de AWS charge.

API Kubernetes **`Gateway`**

Avec la version AWS Load Balancer Controller `2.14.0` ou ultérieure, le LBC crée un Application AWS Load Balancer (ALB) lorsque vous créez un Kubernetes. `Gateway` Kubernetes Gateway normalise davantage la configuration qu'Ingress, qui nécessitait des annotations personnalisées pour de nombreuses options courantes. Passez en revue la configuration que vous pouvez appliquer à une ressource de passerelle. Pour plus d'informations sur l'`Gateway`API, consultez la section API Gateway dans la documentation de Kubernetes.

Installation du contrôleur

Vous pouvez utiliser l'une des procédures suivantes pour installer le AWS Load Balancer Controller :

- Si vous utilisez Amazon EKS pour la première fois, nous vous recommandons d'utiliser Helm pour l'installation, car cela simplifie l'installation du AWS Load Balancer Controller. Pour de plus amples informations, veuillez consulter the section called "Installer avec Helm".

- Pour les configurations avancées, telles que les clusters avec un accès réseau restreint aux registres de conteneurs publics, utilisez les manifestes Kubernetes. Pour de plus amples informations, veuillez consulter the section called "Installer à l'aide de manifestes".

Migration à partir des versions obsolètes du contrôleur

- Si vous avez installé des versions obsolètes du AWS Load Balancer Controller, consultez. the section called "Migration à partir de la version obsolète"
- Les versions rendues obsolètes ne peuvent pas être mises à niveau. Ils doivent être supprimés et une version actuelle du AWS Load Balancer Controller doit être installée.
- Les versions obsolètes incluent :
  - AWS Contrôleur d'entrée ALB pour Kubernetes (« Ingress Controller »), prédécesseur du Load Balancer Controller. AWS
  - N'importe quelle `0.1.x` version du AWS Load Balancer Controller

Ancien fournisseur de cloud

Kubernetes inclut un ancien fournisseur de cloud pour AWS. L'ancien fournisseur de cloud est capable de fournir des équilibreurs de AWS charge, comme le AWS Load Balancer Controller. L'ancien fournisseur de cloud crée des équilibreurs de charge Classic Load Balancer. Si vous n'installez pas le AWS Load Balancer Controller, Kubernetes utilisera par défaut l'ancien fournisseur de cloud. Vous devez installer le AWS Load Balancer Controller et éviter d'utiliser l'ancien fournisseur de cloud.

> **Important**
>
> Dans les versions 2.5 et ultérieures, le AWS Load Balancer Controller devient le contrôleur par défaut pour les ressources de service Kubernetes avec le Network Load Balancer (NLB) `type: LoadBalancer` et crée un AWS Network Load Balancer (NLB) pour chaque service. Pour ce faire, il génère un webhook de mutation pour les services, qui définit le champ `spec.loadBalancerClass` sur `service.k8s.aws/nlb` pour les nouveaux services d'équilibreur `type: LoadBalancer`. Vous pouvez désactiver cette fonctionnalité et revenir à l'utilisation de l'ancien fournisseur de cloud en tant que contrôleur par défaut, en définissant la valeur `enableServiceMutatorWebhook` des charts de Helm sur `false`. À moins que vous ne désactiviez cette fonctionnalité, le cluster ne générera pas de

nouveaux équilibreurs de charge Classic Load Balancer pour vos services. Les équilibreurs de charge Classic Load Balancer existants continueront de fonctionner.

Installez le contrôleur AWS Load Balancer avec Helm

> **Tip**
>
> Inscrivez-vous aux prochains ateliers Amazon EKS.

> **Tip**
>
> Avec le mode automatique Amazon EKS, il n'est pas nécessaire d'installer ou de mettre à niveau de modules complémentaires de mise en réseau. Le mode automatique inclut des fonctionnalités de mise en réseau des pods et d'équilibrage de charge.
> Pour de plus amples informations, veuillez consulter Mode automatique EKS.

Cette rubrique décrit comment installer le AWS Load Balancer Controller à l'aide de Helm, un gestionnaire de packages pour Kubernetes, et. `eksctl` Le contrôleur est installé avec les options par défaut. Pour plus d'informations sur le contrôleur, notamment sur sa configuration à l'aide d'annotations, consultez la documentation du contrôleur AWS Load Balancer sur. GitHub

Dans les étapes suivantes, remplacez les valeurs d'exemple par vos propres valeurs.

Conditions préalables

Avant de commencer ce tutoriel, vous devez suivre les étapes suivantes :

- Créez un cluster Amazon EKS. Pour en créer un, consultez Démarrer.
- Installez Helm sur votre machine locale.
- Assurez-vous que votre plug-in Amazon VPC CNI pour Kubernetes, `kube-proxy`, et les modules complémentaires CoreDNS sont au minimum aux versions indiquées dans les jetons de compte de service.
- Découvrez les concepts d' AWS Elastic Load Balancing. Pour plus d'informations, consultez le Guide de l'utilisateur Elastic Load Balancing.

- Découvrez les ressources Kubernetes Service et Ingress.

Considérations

Avant de procéder aux étapes de configuration décrites sur cette page, tenez compte des points suivants :

- La politique et le rôle IAM (`AmazonEKSLoadBalancerControllerRole`) peuvent être réutilisés sur plusieurs clusters EKS du même AWS compte.
- Si vous installez le contrôleur sur le même cluster où le rôle (`AmazonEKSLoadBalancerControllerRole`) a été créé à l'origine, passez à l'étape 2 : installer le contrôleur d'équilibreur de charge après avoir vérifié que le rôle existe.
- Si vous utilisez les rôles IAM pour les comptes de service (IRSA), IRSA doit être configuré pour chaque cluster, et l'ARN du fournisseur OpenID Connect (OIDC) dans la politique de confiance du rôle est spécifique à chaque cluster EKS. De plus, si vous installez le contrôleur sur un nouveau cluster avec un `AmazonEKSLoadBalancerControllerRole` existant, mettez à jour la politique de confiance du rôle afin d'inclure le fournisseur OIDC du nouveau cluster et créez un nouveau compte de service avec l'annotation de rôle appropriée. Pour déterminer si vous disposez déjà d'un fournisseur OIDC ou pour en créer un, consultez the section called "Fournisseur IAM OIDC".

Étape 1 : créer un rôle IAM à l'aide d'**`eksctl`**

Les étapes suivantes font référence à la version 2.14.1 du AWS Load Balancer Controller. Pour plus d'informations sur toutes les versions, consultez la page de mise à jour du AWS Load Balancer Controller sur. GitHub

1. Téléchargez une politique IAM pour le AWS Load Balancer Controller qui lui permet de passer des appels en votre AWS APIs nom.

```
curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-
controller/v2.14.1/docs/install/iam_policy.json
```

   - Si vous êtes une AWS partition non standard, telle qu'une région gouvernementale ou chinoise, consultez les politiques GitHub et téléchargez la politique appropriée pour votre région.

2. Créez une politique IAM à l'aide de la politique téléchargée à l'étape précédente.

```
aws iam create-policy \
```

```
    --policy-name AWSLoadBalancerControllerIAMPolicy \
    --policy-document file://iam_policy.json
```

> **Note**
>
> Si vous consultez la politique dans le AWS Management Console, la console affiche des avertissements pour le service ELB, mais pas pour le service ELB v2. Cela est dû au fait que certaines des actions de la politique sont disponibles pour ELB v2, mais pas pour ELB. Vous pouvez ignorer les avertissements relatifs au service ELB.

3. Remplacez les valeurs pour le nom du cluster, le code régional et l'ID du compte.

```
eksctl create iamserviceaccount \
    --cluster=<cluster-name> \
    --namespace=kube-system \
    --name=aws-load-balancer-controller \
    --attach-policy-arn=arn:aws: iam::<AWS_ACCOUNT_ID>:policy/
AWSLoadBalancerControllerIAMPolicy \
    --override-existing-serviceaccounts \
    --region <aws-region-code> \
    --approve
```

Étape 2 : Installation du AWS Load Balancer Controller

1. Ajoutez le référentiel de `eks-charts` diagrammes Helm. AWS maintient ce dépôt activé GitHub.

```
helm repo add eks https://aws.github.io/eks-charts
```

2. Mettez à jour votre référentiel local pour vous assurer que vous disposez des graphiques les plus récents.

```
helm repo update eks
```

3. Installez le AWS Load Balancer Controller.

   Si vous déployez le contrôleur sur des EC2 nœuds Amazon ayant un accès restreint au service de métadonnées d' EC2 instance Amazon (IMDS), ou si vous déployez sur des nœuds hybrides Fargate ou Amazon EKS, ajoutez les indicateurs suivants `helm` à la commande suivante :

   - `--set region=`*`region-code`*

- `--set vpcId=vpc-xxxxxxxx`

  Remplacez *my-cluster* par le nom de votre cluster. Dans la commande suivante, `aws-load-balancer-controller` est le compte de service Kubernetes que vous avez créé à l'étape précédente.

  Pour plus d'informations sur la configuration du graphique de barre, consultez values.yaml on. GitHub

```
helm install aws-load-balancer-controller eks/aws-load-balancer-controller \
  -n kube-system \
  --set clusterName=my-cluster \
  --set serviceAccount.create=false \
  --set serviceAccount.name=aws-load-balancer-controller \
  --version 1.14.0
```

> **Important**
>
> Le graphique déployé ne reçoit pas automatiquement les mises à jour de sécurité. Vous devez effectuer manuellement une mise à niveau vers un graphique plus récent lorsqu'il sera disponible. Lors de la mise *install* à niveau, `upgrade` passez à la commande précédente.

La `helm install` commande installe automatiquement les définitions de ressources personnalisées (CRDs) pour le contrôleur. Ce n'est pas le cas de la commande `helm upgrade`. Si vous utilisez, `helm upgrade,` vous devez installer manuellement le CRDs. Exécutez la commande suivante pour installer CRDs :

```
wget https://raw.githubusercontent.com/aws/eks-charts/master/stable/aws-load-balancer-controller/crds/crds.yaml
kubectl apply -f crds.yaml
```

Étape 3 : vérifier que le contrôleur est installé

1. Vérifiez que le contrôleur est installé.

```
kubectl get deployment -n kube-system aws-load-balancer-controller
```

L'exemple qui suit illustre un résultat.

```
NAME                           READY   UP-TO-DATE   AVAILABLE   AGE
aws-load-balancer-controller   2/2     2            2           84s
```

Vous recevez la sortie précédente si vous avez déployé à l'aide de Helm. Si vous avez déployé à l'aide du manifeste Kubernetes, vous n'avez qu'un seul réplica.

2. Avant d'utiliser le contrôleur pour provisionner AWS des ressources, votre cluster doit répondre à des exigences spécifiques. Pour plus d'informations, consultez the section called “Répartition de la charge des applications” et the section called “Répartition de charge réseau”.

Installer le AWS Load Balancer Controller avec des manifestes

> **Tip**
>
> Avec le mode automatique Amazon EKS, il n'est pas nécessaire d'installer ou de mettre à niveau de modules complémentaires de mise en réseau. Le mode automatique inclut des fonctionnalités de mise en réseau des pods et d'équilibrage de charge.
> Pour de plus amples informations, veuillez consulter Mode automatique EKS.

Cette rubrique décrit comment installer le contrôleur en téléchargeant et en appliquant des manifestes Kubernetes. Vous pouvez afficher l'intégralité de la documentation pour le contrôleur sur GitHub.

Dans les étapes suivantes, remplacez les valeurs d'exemple par vos propres valeurs.

Conditions préalables

Avant de commencer ce tutoriel, vous devez suivre les étapes suivantes :

- Créez un cluster Amazon EKS. Pour en créer un, consultez Démarrer.
- Installez Helm sur votre machine locale.
- Assurez-vous que votre plug-in Amazon VPC CNI pour Kubernetes, `kube-proxy`, et les modules complémentaires CoreDNS sont au minimum aux versions indiquées dans les jetons de compte de service.

- Découvrez les concepts d' AWS Elastic Load Balancing. Pour plus d'informations, consultez le Guide de l'utilisateur Elastic Load Balancing.
- Découvrez les ressources Kubernetes Service et Ingress.

Considérations

Avant de procéder aux étapes de configuration décrites sur cette page, tenez compte des points suivants :

- La politique et le rôle IAM (`AmazonEKSLoadBalancerControllerRole`) peuvent être réutilisés sur plusieurs clusters EKS du même AWS compte.
- Si vous installez le contrôleur sur le même cluster où le rôle (`AmazonEKSLoadBalancerControllerRole`) a été créé à l'origine, passez à l'Étape 2 : installer cert-manager après avoir vérifié que le rôle existe.
- Si vous utilisez les rôles IAM pour les comptes de service (IRSA), IRSA doit être configuré pour chaque cluster, et l'ARN du fournisseur OpenID Connect (OIDC) dans la politique de confiance du rôle est spécifique à chaque cluster EKS. De plus, si vous installez le contrôleur sur un nouveau cluster avec un `AmazonEKSLoadBalancerControllerRole` existant, mettez à jour la politique de confiance du rôle afin d'inclure le fournisseur OIDC du nouveau cluster et créez un nouveau compte de service avec l'annotation de rôle appropriée. Pour déterminer si vous disposez déjà d'un fournisseur OIDC ou pour en créer un, consultez the section called "Fournisseur IAM OIDC".

Étape 1 : configurer IAM

Les étapes suivantes font référence à la version 2.14.1 du AWS Load Balancer Controller. Pour plus d'informations sur toutes les versions, consultez la page de mise à jour du AWS Load Balancer Controller sur. GitHub

1. Téléchargez une politique IAM pour le AWS Load Balancer Controller qui lui permet de passer des appels en votre AWS APIs nom.

   Example

   AWS

   ```
   curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-
   controller/v2.14.1/docs/install/iam_policy.json
   ```

AWS GovCloud (US)

```
curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-
controller/v2.14.1/docs/install/iam_policy_us-gov.json
```

```
mv iam_policy_us-gov.json iam_policy.json
```

2. Créez une politique IAM à l'aide de la politique téléchargée à l'étape précédente.

```
aws iam create-policy \
    --policy-name AWSLoadBalancerControllerIAMPolicy \
    --policy-document file://iam_policy.json
```

> **Note**
>
> Si vous consultez la politique dans le AWS Management Console, la console affiche des avertissements pour le service ELB, mais pas pour le service ELB v2. Cela est dû au fait que certaines des actions de la politique sont disponibles pour ELB v2, mais pas pour ELB. Vous pouvez ignorer les avertissements relatifs au service ELB.

Example

eksctl

a. Remplacez *my-cluster* par le nom de votre cluster, *111122223333* par votre ID de compte, puis exécutez la commande.

```
eksctl create iamserviceaccount \
  --cluster=my-cluster \
  --namespace=kube-system \
  --name=aws-load-balancer-controller \
  --role-name AmazonEKSLoadBalancerControllerRole \
  --attach-policy-arn=arn:aws: iam::111122223333:policy/
AWSLoadBalancerControllerIAMPolicy \
  --approve
```

AWS CLI and kubectl

a. Récupérez l'ID du fournisseur OIDC de votre cluster et stockez-le dans une variable.

```
oidc_id=$(aws eks describe-cluster --name my-cluster --query
 "cluster.identity.oidc.issuer" --output text | cut -d '/' -f 5)
```

b. Déterminez si un fournisseur IAM OIDC avec l'ID de votre cluster existe déjà dans votre compte. Vous devez configurer OIDC à la fois pour le cluster et pour IAM.

```
aws iam list-open-id-connect-providers | grep $oidc_id | cut -d "/" -f4
```

Si une sortie est renvoyée, cela signifie que vous disposez déjà d'un fournisseur IAM OIDC pour votre cluster. Si aucune sortie n'est renvoyée, vous devez créer un fournisseur OIDC IAM pour votre cluster. Pour de plus amples informations, veuillez consulter the section called "Fournisseur IAM OIDC".

c. Copiez les contenus suivants sur votre appareil. Remplacez *111122223333* par votre ID de compte. Remplacez *region-code* par la AWS région dans laquelle se trouve votre cluster. Remplacez *EXAMPLED539D4633E53DE1B71EXAMPLE* par la sortie renvoyée dans l'étape précédente.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Federated": "arn:aws:iam::111122223333:oidc-provider/oidc.eks.us-
east-1.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
            },
            "Action": "sts:AssumeRoleWithWebIdentity",
            "Condition": {
                "StringEquals": {
                    "oidc.eks.us-east-1.amazonaws.com/id/
EXAMPLED539D4633E53DE1B71EXAMPLE:aud": "sts.amazonaws.com",
                    "oidc.eks.us-east-1.amazonaws.com/id/
EXAMPLED539D4633E53DE1B71EXAMPLE:sub": "system:serviceaccount:kube-system:aws-
load-balancer-controller"
                }
            }
        }
```

```
    ]
}
```

d. Créez le rôle IAM.

```
aws iam create-role \
  --role-name AmazonEKSLoadBalancerControllerRole \
  --assume-role-policy-document file://"load-balancer-role-trust-policy.json"
```

e. Attachez la politique IAM gérée par Amazon EKS au rôle IAM. Remplacez *111122223333* par votre ID de compte.

```
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::111122223333:policy/
AWSLoadBalancerControllerIAMPolicy \
  --role-name AmazonEKSLoadBalancerControllerRole
```

f. Copiez les contenus suivants sur votre appareil. Remplacez *111122223333* par votre ID de compte. Après avoir remplacé le texte, exécutez la commande modifiée pour créer le fichier `aws-load-balancer-controller-service-account.yaml`.

```
cat >aws-load-balancer-controller-service-account.yaml <<EOF
apiVersion: v1
kind: ServiceAccount
metadata:
  labels:
    app.kubernetes.io/component: controller
    app.kubernetes.io/name: aws-load-balancer-controller
  name: aws-load-balancer-controller
  namespace: kube-system
  annotations:
    eks.amazonaws.com/role-arn: arn:aws: iam::111122223333:role/
AmazonEKSLoadBalancerControllerRole
EOF
```

g. Créez le compte de service Kubernetes sur votre cluster. Le compte de service Kubernetes nommé `aws-load-balancer-controller` est annoté avec le nom du rôle IAM que vous avez créé. *AmazonEKSLoadBalancerControllerRole*

```
kubectl apply -f aws-load-balancer-controller-service-account.yaml
```

Étape 2 : Installation **`cert-manager`**

Installez `cert-manager` en utilisant l'une des méthodes suivantes pour injecter la configuration du certificat dans les webhooks. Pour plus d'informations, consultez le guide de démarrage de la documentation cert-manager.

Pour installer `cert-manager`, nous vous recommandons d'utiliser le registre des conteneurs `quay.io`. Si vos nœuds n'ont pas accès au registre de conteneurs `quay.io`, installez `cert-manager` à l'aide d'Amazon ECR (voir ci-dessous).

Example

Quay.io

a. Si vos nœuds ont accès au registre de conteneurs `quay.io`, installez `cert-manager` pour injecter la configuration du certificat dans les webhooks.

```
kubectl apply \
    --validate=false \
    -f https://github.com/jetstack/cert-manager/releases/download/v1.13.5/cert-
manager.yaml
```

Amazon ECR

a. Installez `cert-manager` en utilisant l'une des méthodes suivantes pour injecter la configuration du certificat dans les webhooks. Pour plus d'informations, consultez le guide de démarrage de la documentation cert-manager.
b. Téléchargez le manifeste.

```
curl -Lo cert-manager.yaml https://github.com/jetstack/cert-manager/releases/
download/v1.13.5/cert-manager.yaml
```

c. Extrayez les images suivantes et transférez-les vers un référentiel auquel vos nœuds ont accès. Pour plus d'informations sur l'extraction, le balisage et le transfert d'images vers votre propre référentiel, consultez the section called "Copier une image dans un référentiel".

```
quay.io/jetstack/cert-manager-cainjector:v1.13.5
quay.io/jetstack/cert-manager-controller:v1.13.5
quay.io/jetstack/cert-manager-webhook:v1.13.5
```

d. Remplacez `quay.io` dans le manifeste des trois images par votre propre nom de registre. La commande suivante suppose que le nom de votre référentiel privé est le même que celui du référentiel source. Remplacez *111122223333.dkr.ecr.region-code.amazonaws.com* par votre registre privé.

```
sed -i.bak -e 's|quay.io|111122223333.dkr.ecr.region-code.amazonaws.com|' ./cert-manager.yaml
```

e. Appliquez le manifeste.

```
kubectl apply \
    --validate=false \
    -f ./cert-manager.yaml
```

Étape 3 : Installation du AWS Load Balancer Controller

1. Téléchargez la spécification du contrôleur. Pour plus d'informations sur le contrôleur, consultez la documentation sur GitHub.

```
curl -Lo v2_14_1_full.yaml https://github.com/kubernetes-sigs/aws-load-balancer-controller/releases/download/v2.14.1/v2_14_1_full.yaml
```

2. Effectuez les modifications suivantes dans le fichier.

a. Si vous avez téléchargé le fichier `v2_14_1_full.yaml`, exécutez la commande suivante pour supprimer la section `ServiceAccount` du manifeste. Si vous ne supprimez pas cette section, l'annotation requise que vous avez apportée au compte de service lors d'une étape précédente sera écrasée. La suppression de cette section préserve également le compte de service que vous avez créé à une étape précédente, si vous supprimez le contrôleur.

```
sed -i.bak -e '764,772d' ./v2_14_1_full.yaml
```

Si vous avez téléchargé une version différente du fichier, ouvrez le fichier dans un éditeur et supprimez les lignes suivantes.

```
apiVersion: v1
kind: ServiceAccount
metadata:
  labels:
```

```
    app.kubernetes.io/component: controller
    app.kubernetes.io/name: aws-load-balancer-controller
  name: aws-load-balancer-controller
  namespace: kube-system
---
```

b. Remplacez `your-cluster-name` dans la section `Deployment spec` du fichier par le nom de votre cluster en remplaçant *`my-cluster`* par le nom de votre cluster.

```
sed -i.bak -e 's|your-cluster-name|my-cluster|' ./v2_14_1_full.yaml
```

c. Si vos nœuds n'ont pas accès aux référentiels d'images Amazon EKS Amazon ECR, vous devez extraire l'image suivante et l'envoyer vers un référentiel auquel vos nœuds ont accès. Pour plus d'informations sur l'extraction, l'identification et le transfert d'une image vers votre propre référentiel, consultez the section called "Copier une image dans un référentiel".

```
public.ecr.aws/eks/aws-load-balancer-controller:v2.14.1
```

Ajoutez le nom de votre registre au manifeste. La commande suivante suppose que le nom de votre référentiel privé est le même que celui du référentiel source et ajoute le nom de votre registre privé au fichier. Remplacez *`111122223333.dkr.ecr.region-code.amazonaws.com`* par votre registre. Cette ligne suppose que vous avez nommé votre référentiel privé de la même manière que le référentiel source. Dans le cas contraire, remplacez le texte `eks/aws-load-balancer-controller` après votre nom de registre privé par le nom de votre référentiel.

```
sed -i.bak -e 's|public.ecr.aws/eks/aws-load-balancer-controller|
111122223333.dkr.ecr.region-code.amazonaws.com/eks/aws-load-balancer-
controller|' ./v2_14_1_full.yaml
```

d. (Requis uniquement pour Fargate ou Restricted IMDS)

Si vous déployez le contrôleur sur des EC2 nœuds Amazon ayant un accès restreint au service de métadonnées d' EC2 instance Amazon (IMDS), ou si vous déployez sur des nœuds hybrides Fargate ou Amazon EKS, ajoutez le point ci-dessous. `following parameters - args:`

```
[...]
spec:
      containers:
        - args:
```

```
            - --cluster-name=your-cluster-name
            - --ingress-class=alb
            - --aws-vpc-id=vpc-xxxxxxxx
            - --aws-region=region-code


[...]
```

3. Appliquez le fichier.

```
kubectl apply -f v2_14_1_full.yaml
```

4. Téléchargez l'attaque de l'homme du milieu (HDM) `IngressClass` et `IngressClassParams` à votre cluster.

```
curl -Lo v2.14.1_ingclass.yaml https://github.com/kubernetes-sigs/aws-load-balancer-controller/releases/download/v2.14.1/v2_14_1_ingclass.yaml
```

5. Appliquez le manifeste à votre cluster.

```
kubectl apply -f v2_14_1_ingclass.yaml
```

Étape 4 : vérifier que le contrôleur est installé

1. Vérifiez que le contrôleur est installé.

```
kubectl get deployment -n kube-system aws-load-balancer-controller
```

   L'exemple qui suit illustre un résultat.

```
NAME                           READY   UP-TO-DATE   AVAILABLE   AGE
aws-load-balancer-controller   2/2     2            2           84s
```

   Vous recevez la sortie précédente si vous avez déployé à l'aide de Helm. Si vous avez déployé à l'aide du manifeste Kubernetes, vous n'avez qu'un seul réplica.

2. Avant d'utiliser le contrôleur pour provisionner AWS des ressources, votre cluster doit répondre à des exigences spécifiques. Pour plus d'informations, consultez the section called “Répartition de la charge des applications” et the section called “Répartition de charge réseau”.

Migrer des applications depuis le ALB Ingress Controller

Cette rubrique décrit comment Migration à partir de versions de contrôleur rendues obsolètes. Plus précisément, il décrit comment supprimer les versions obsolètes du AWS Load Balancer Controller.

- Les versions rendues obsolètes ne peuvent pas être mises à niveau. Vous devez d'abord les supprimer, puis installer une version actuelle.
- Les versions obsolètes incluent :
  - AWS Contrôleur d'entrée ALB pour Kubernetes (« Ingress Controller »), prédécesseur du Load Balancer Controller. AWS
  - N'importe quelle `0.1.x` version du AWS Load Balancer Controller

Supprimer la version obsolète du contrôleur

> **Note**
>
> Vous avez peut-être installé la version obsolète à l'aide de Helm ou manuellement avec les manifestes Kubernetes. Terminez la procédure à l'aide de l'outil avec lequel vous l'avez installé à l'origine.

1. Si vous avez installé les Charts de Helm `incubator/aws-alb-ingress-controller`, désinstallez-les.

   ```
   helm delete aws-alb-ingress-controller -n kube-system
   ```

2. Si la version `0.1.x` du graphique `eks-charts/aws-load-balancer-controller` est installée, désinstallez-la. La mise à niveau depuis `0.1.x` vers la version `1.0.0` ne fonctionne pas en raison d'une incompatibilité avec la version de l'API du webhook.

   ```
   helm delete aws-load-balancer-controller -n kube-system
   ```

3. Vérifiez si le contrôleur est actuellement installé.

   ```
   kubectl get deployment -n kube-system alb-ingress-controller
   ```

   Il s'agit de la sortie si le contrôleur n'est pas installé.

```
Error from server (NotFound): deployments.apps "alb-ingress-controller" not found
```

Il s'agit de la sortie si le contrôleur est installé.

```
NAME                      READY UP-TO-DATE AVAILABLE AGE
alb-ingress-controller 1/1   1          1         122d
```

4. Saisissez les commandes suivantes pour supprimer le contrôleur.

```
kubectl delete -f https://raw.githubusercontent.com/kubernetes-sigs/aws-alb-ingress-
controller/v1.1.8/docs/examples/alb-ingress-controller.yaml
kubectl delete -f https://raw.githubusercontent.com/kubernetes-sigs/aws-alb-ingress-
controller/v1.1.8/docs/examples/rbac-role.yaml
```

Migrer vers AWS Load Balancer Controller

Pour migrer de l'ALB Ingress Controller pour Kubernetes vers le Load Balancer Controller, vous AWS devez :

1. Retirez le contrôleur d'entrée ALB (voir ci-dessus).
2. Installez le AWS Load Balancer Controller.
3. Ajoutez une politique supplémentaire au rôle IAM utilisé par le AWS Load Balancer Controller. Cette politique permet au LBC de gérer les ressources créées par le contrôleur d'entrée ALB pour Kubernetes.
4. Téléchargez la politique IAM. Cette politique permet au AWS Load Balancer Controller de gérer les ressources créées par le contrôleur ALB Ingress pour Kubernetes. Vous pouvez également afficher la politique.

```
curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-
controller/v2.14.1/docs/install/iam_policy_v1_to_v2_additional.json
```

5. Si votre cluster se trouve dans les AWS régions AWS GovCloud (USA Est) ou AWS GovCloud (USA Ouest), remplacez-le par `arn:aws:` . `arn:aws-us-gov:`

```
sed -i.bak -e 's|arn:aws:|arn:aws-us-gov:|' iam_policy_v1_to_v2_additional.json
```

6. Créez la politique IAM et notez l'ARN renvoyé.

```
aws iam create-policy \
  --policy-name AWSLoadBalancerControllerAdditionalIAMPolicy \
  --policy-document file://iam_policy_v1_to_v2_additional.json
```

7. Attachez la politique IAM au rôle IAM utilisé par le AWS Load Balancer Controller. Remplacez *`your-role-name`* par le nom du rôle, tel que`AmazonEKSLoadBalancerControllerRole`.

   Si vous avez créé le rôle en utilisant`eksctl`, pour trouver le nom du rôle créé, ouvrez la AWS CloudFormation console et sélectionnez la pile eksctl- ***`my-cluster`*** - addon-iamserviceaccount-kube-system - aws-load-balancer-controller. Sélectionnez l'onglet Resources (Ressources). Le nom du rôle se trouve dans la colonne ID Physique.

```
aws iam attach-role-policy \
  --role-name your-role-name \
  --policy-arn arn:aws: iam::111122223333:policy/
AWSLoadBalancerControllerAdditionalIAMPolicy
```

Gérer CoreDNS pour DNS dans les clusters Amazon EKS

> **Tip**
>
> Avec le mode automatique Amazon EKS, il n'est pas nécessaire d'installer ou de mettre à niveau de modules complémentaires de mise en réseau. Le mode automatique inclut des fonctionnalités de mise en réseau des pods et d'équilibrage de charge.
> Pour de plus amples informations, veuillez consulter Mode automatique EKS.

CoreDNS est un serveur DNS flexible et extensible qui peut servir de DNS de cluster Kubernetes. Lorsque vous lancez un cluster Amazon EKS avec au moins un nœud, deux réplicas de l'image CoreDNS sont déployés par défaut, quel que soit le nombre de nœuds déployés dans votre cluster. Les pods CoreDNS fournissent une résolution de noms pour tous les pods du cluster. Les pods CoreDNS peuvent être déployés sur des nœuds Fargate si votre cluster comprend un profil Fargate avec un espace de noms qui correspond à l'espace de noms de CoreDNS `deployment`. Pour plus d'informations sur CoreDNS, consultez Utilisation de CoreDNS pour la découverte des services dans la documentation Kubernetes.

Versions de CoreDNS

Le tableau suivant répertorie la dernière version du type d'add-on Amazon EKS pour chaque version de Kubernetes.

Version de Kubernetes	Version de CoreDNS
1,35	v1.13.2-eksbuild.1
1,34	v1.13.2-eksbuild.1
1,33	v1.13.2-eksbuild.1
1,32	v1.11.4-eksbuild.28
1,31	v1.11.4-eksbuild.28
1,30	v1.11.4-eksbuild.28
1,29	v1.11.4-eksbuild.28

Important

Si vous gérez vous-même ce module complémentaire, les versions répertoriées dans le tableau peuvent ne pas être les mêmes que les versions autogérées disponibles. Pour plus d'informations sur la mise à jour du type autogéré de ce module complémentaire, consultez la rubrique the section called "Mise à jour (autogéré)".

Considérations importantes concernant la mise à niveau de CoreDNS

- Les mises à jour CoreDNS utilisent PodDisruptionBudget un pour maintenir la disponibilité du service DNS pendant le processus de mise à jour.
- Afin d'améliorer la stabilité et la disponibilité du déploiement CoreDNS, les versions `v1.9.3-eksbuild.6` et ultérieures ainsi que `v1.10.1-eksbuild.3` sont déployées avec un `PodDisruptionBudget`. Si vous avez déployé une version existante `PodDisruptionBudget`, la mise à niveau vers ces versions risque d'échouer. Si la mise à niveau échoue, l'exécution de l'une des tâches suivantes devrait résoudre le problème :

  - Lors de la mise à niveau du module complémentaire Amazon EKS, choisissez de remplacer les paramètres existants comme option de résolution des conflits. Si vous avez défini d'autres paramètres personnalisés pour le déploiement, veillez à sauvegarder vos paramètres avant la mise à niveau afin de pouvoir les réappliquer après la mise à niveau.
  - Supprimez votre version existante `PodDisruptionBudget` et réessayez d'effectuer la mise à niveau.
- Dans les versions EKS add-on `v1.9.3-eksbuild.3` et ultérieures ainsi que `v1.10.1-eksbuild.6` et ultérieures, le déploiement CoreDNS configure `readinessProbe` pour qu'il utilise le point de terminaison `/ready`. Ce point de terminaison est activé dans le fichier de configuration `Corefile` de CoreDNS.

  Si vous utilisez un `Corefile` personnalisé, vous devez ajouter le plug-in `ready` à la configuration afin que le point de terminaison `/ready` soit actif dans CoreDNS pour que la sonde puisse l'utiliser.
- Dans les versions `v1.9.3-eksbuild.7` et ultérieures, et `v1.10.1-eksbuild.4` et ultérieures des modules complémentaires EKS, vous pouvez modifier le `PodDisruptionBudget`. Vous pouvez modifier le module complémentaire et modifier ces paramètres dans les paramètres de configuration facultatifs à l'aide des champs de l'exemple suivant. Cet exemple montre la valeur `PodDisruptionBudget` par défaut.

```
{
    "podDisruptionBudget": {
        "enabled": true,
        "maxUnavailable": 1
        }
}
```

  Vous pouvez définir `maxUnavailable` ou `minAvailable`, mais vous ne pouvez pas définir les deux dans un même `PodDisruptionBudget`. Pour plus d'informations`PodDisruptionBudgets`, consultez la section Spécifier un PodDisruptionBudget dans la documentation de Kubernetes.

  Notez que si vous définissez `enabled` sur `false`, le `PodDisruptionBudget` n'est pas supprimé. Après avoir défini ce champ sur `false`, vous devez supprimer l'objet `PodDisruptionBudget`. De même, si vous modifiez le module complémentaire pour utiliser une ancienne version du module complémentaire (rétrogradage) après la mise à niveau vers une

version avec un `PodDisruptionBudget`, le `PodDisruptionBudget` n'est pas supprimé. Pour supprimer le `PodDisruptionBudget`, exécutez la commande suivante :

```
kubectl delete poddisruptionbudget coredns -n kube-system
```

- Dans les versions du module complémentaire EKS et `v1.10.1-eksbuild.5` et ultérieures, modifiez la tolérance par défaut de `node-role.kubernetes.io/master:NoSchedule` à `node-role.kubernetes.io/control-plane:NoSchedule` pour vous conformer au KEP 2067. Pour plus d'informations sur KEP 2067, voir KEP-2067 : Renommer l'étiquette « master » de kubeadm et altérer les propositions d'amélioration de Kubernetes () sur. KEPs GitHub

  Dans les versions du module complémentaire EKS `v1.8.7-eksbuild.8` et ultérieures ainsi que `v1.9.3-eksbuild.9` et ultérieures, les deux tolérances sont définies pour être compatibles avec toutes les versions de Kubernetes.
- Dans les versions du module complémentaire EKS `v1.9.3-eksbuild.11` et `v1.10.1-eksbuild.7` et ultérieures, le déploiement CoreDNS définit une valeur par défaut pour `topologySpreadConstraints`. La valeur par défaut garantit que les pods CoreDNS sont répartis entre les zones de disponibilité si des nœuds sont disponibles dans plusieurs zones de disponibilité. Vous pouvez définir une valeur personnalisée qui sera utilisée à la place de la valeur par défaut. La valeur par défaut est la suivante :

```
topologySpreadConstraints:
  - maxSkew: 1
    topologyKey: topology.kubernetes.io/zone
    whenUnsatisfiable: ScheduleAnyway
    labelSelector:
      matchLabels:
        k8s-app: kube-dns
```

Considérations relatives à la mise à niveau de CoreDNS **v1.11**

- Dans les versions du module complémentaire EKS `v1.11.1-eksbuild.4` et ultérieures, l'image du conteneur est basée sur une image de base minimale gérée par Amazon EKS Distro, qui contient un nombre minimal de paquets et ne dispose pas de shells. Pour plus d'informations, consultez Amazon EKS Distro. L'utilisation et le dépannage de l'image CoreDNS restent les mêmes.

Créer le module complémentaire CoreDNS Amazon EKS

Créer le module complémentaire CoreDNS Amazon EKS. Vous devez disposer d'un cluster avant de créer le module complémentaire. Pour de plus amples informations, consultez the section called "Créer un cluster".

1. Déterminez la version du module complémentaire actuellement installée sur votre cluster.

   ```
   kubectl describe deployment coredns --namespace kube-system | grep coredns: | cut -
   d : -f 3
   ```

   L'exemple qui suit illustre un résultat.

   ```
   v1.10.1-eksbuild.13
   ```

2. Déterminez le type de module complémentaire installé sur votre cluster. Selon l'outil avec lequel vous avez créé votre cluster, le type de module complémentaire Amazon EKS peut ne pas être actuellement installé sur votre cluster. Remplacez *my-cluster* par le nom de votre cluster.

   ```
   aws eks describe-addon --cluster-name my-cluster --addon-name coredns --query
    addon.addonVersion --output text
   ```

   Si un numéro de version est renvoyé, le type de module complémentaire Amazon EKS est installé sur votre cluster, et vous n'avez donc pas besoin de suivre les étapes restantes de cette procédure. Si une erreur est renvoyée, cela signifie que le type de module complémentaire Amazon EKS n'est pas installé sur votre cluster. Suivez les étapes restantes de cette procédure pour l'installer.

3. Enregistrez la configuration du module complémentaire actuellement installé.

   ```
   kubectl get deployment coredns -n kube-system -o yaml > aws-k8s-coredns-old.yaml
   ```

4. Créez le module complémentaire à l'aide de l'AWS CLI. Si vous souhaitez utiliser la AWS Management Console ou `eksctl` pour créer le module complémentaire, consultez la rubrique the section called "Créer un module complémentaire" et spécifiez `coredns` pour le nom du module complémentaire. Copiez la commande qui suit sur votre appareil. Si nécessaire, apportez les modifications suivantes à la commande, puis exécutez la commande modifiée.

   - Remplacez *my-cluster* par le nom de votre cluster.

- Remplacez *v1.11.3-eksbuild.1* par la dernière version répertoriée dans le tableau des dernières versions pour votre version de cluster.

```
aws eks create-addon --cluster-name my-cluster --addon-name coredns --addon-version
 v1.11.3-eksbuild.1
```

Si vous avez appliqué à votre module complémentaire actuel des paramètres personnalisés qui entrent en conflit avec les paramètres par défaut du module complémentaire Amazon EKS, la création peut échouer. Si la création échoue, vous recevez un message d'erreur qui peut vous aider à résoudre le problème. Vous pouvez également ajouter `--resolve-conflicts OVERWRITE` à la commande précédente. Cela permet au module complémentaire de remplacer les paramètres personnalisés existants. Une fois que vous avez créé le module complémentaire, vous pouvez le mettre à jour avec vos paramètres personnalisés.

5. Vérifiez que la dernière version du module complémentaire pour la version Kubernetes de votre cluster a bien été ajoutée à votre cluster. Remplacez *my-cluster* par le nom de votre cluster.

```
aws eks describe-addon --cluster-name my-cluster --addon-name coredns --query
 addon.addonVersion --output text
```

La création du module complémentaire peut prendre plusieurs secondes.

L'exemple qui suit illustre un résultat.

```
v1.11.3-eksbuild.1
```

6. Si vous avez personnalisé les paramètres du module complémentaire, avant de créer le module complémentaire Amazon EKS, utilisez la configuration que vous avez enregistrée lors d'une étape précédente pour mettre à jour le module complémentaire Amazon EKS avec vos paramètres personnalisés. Pour obtenir des instructions sur la mise à jour du module complémentaire, consultez the section called “Mise à jour (module complémentaire EKS)”.

Mettre à jour le module complémentaire CoreDNS Amazon EKS

Mettez à jour le type de module complémentaire Amazon EKS. Si vous n'avez pas ajouté le module complémentaire Amazon EKS à votre cluster, ajoutez-le ou consultez the section called “Mise à jour (autogéré)”.

Avant de commencer, consultez les considérations relatives à la mise à niveau. Pour de plus amples informations, consultez the section called "Considérations importantes concernant la mise à niveau de CoreDNS".

1. Déterminez la version du module complémentaire actuellement installée sur votre cluster. Remplacez *my-cluster* par le nom de votre cluster.

   ```
   aws eks describe-addon --cluster-name my-cluster --addon-name coredns --query
    "addon.addonVersion" --output text
   ```

   L'exemple qui suit illustre un résultat.

   ```
   v1.10.1-eksbuild.13
   ```

   Si la version renvoyée correspond à la version Kubernetes de votre cluster dans le tableau des dernières versions, cela signifie que vous disposez déjà de la dernière version installée sur votre cluster et que vous n'avez pas besoin de suivre le reste de cette procédure. Si vous recevez une erreur au lieu d'un numéro de version dans votre sortie, cela signifie que le type de module complémentaire Amazon EKS n'est pas installé sur votre cluster. Vous devez créer le module complémentaire avant de pouvoir le mettre à jour à l'aide de cette procédure.
2. Enregistrez la configuration du module complémentaire actuellement installé.

   ```
   kubectl get deployment coredns -n kube-system -o yaml > aws-k8s-coredns-old.yaml
   ```
3. Mettez à jour votre module complémentaire à l'aide de l'AWS CLI. Si vous souhaitez utiliser la AWS Management Console ou `eksctl` pour mettre à jour le module complémentaire, consultez la rubrique the section called "Mettre à jour un module complémentaire". Copiez la commande qui suit sur votre appareil. Si nécessaire, apportez les modifications suivantes à la commande, puis exécutez la commande modifiée.
   - Remplacez *my-cluster* par le nom de votre cluster.
   - Remplacez *v1.11.3-eksbuild.1* par la dernière version répertoriée dans le tableau des dernières versions pour votre version de cluster.
   - Cette option `--resolve-conflicts`*PRESERVE* conserve les valeurs de configuration existantes pour le module complémentaire. Si vous avez défini des valeurs personnalisées pour les paramètres des modules complémentaires et que vous n'utilisez pas cette option, Amazon EKS remplace vos valeurs par ses valeurs par défaut. Si vous utilisez cette option, nous vous recommandons de tester les modifications de champ et de valeur sur un cluster hors

production avant de mettre à jour le module complémentaire sur votre cluster de production. Si vous remplacez cette valeur par `OVERWRITE`, tous les paramètres sont remplacés par les valeurs par défaut d'Amazon EKS. Si vous avez défini des valeurs personnalisées pour certains paramètres, il est possible qu'elles soient remplacées par les valeurs par défaut d'Amazon EKS. Si vous remplacez cette valeur par `none`, Amazon EKS ne modifie la valeur d'aucun paramètre, mais la mise à jour risque d'échouer. Si la mise à jour échoue, vous recevez un message d'erreur pour vous aider à résoudre le conflit.

- Si vous ne mettez à jour aucun paramètre de configuration, supprimez `--configuration-values '{"replicaCount":3}'` de la commande. Si vous mettez à jour un paramètre de configuration, remplacez `"replicaCount":3` par le paramètre que vous voulez définir. Dans cet exemple, le nombre de réplicas de CoreDNS est défini sur `3`. La valeur que vous spécifiez doit être valide pour le schéma de configuration. Si vous ne connaissez pas le schéma de configuration, exécutez `aws eks describe-addon-configuration --addon-name coredns --addon-version v1.11.3-eksbuild.1` , en remplaçant `v1.11.3-eksbuild.1` par le numéro de version du module complémentaire dont vous souhaitez voir la configuration. Le schéma est renvoyé dans la sortie. Si vous disposez déjà d'une configuration personnalisée et vous voulez la supprimer et rétablir les valeurs par défaut d'Amazon EKS pour tous les paramètres, supprimez `"replicaCount":3` de la commande, de sorte que le champ `{}` soit vide. Pour plus d'informations sur les paramètres CoreDNS, consultez la section Personnalisation du service DNS dans la documentation Kubernetes.

```
aws eks update-addon --cluster-name my-cluster --addon-name coredns --addon-version
 v1.11.3-eksbuild.1 \
    --resolve-conflicts PRESERVE --configuration-values '{"replicaCount":3}'
```

La mise à jour peut prendre plusieurs secondes.

4. Assurez-vous que la version du module complémentaire a été mise à jour. Remplacez `my-cluster` par le nom de votre cluster.

```
aws eks describe-addon --cluster-name my-cluster --addon-name coredns
```

La mise à jour peut prendre plusieurs secondes.

L'exemple qui suit illustre un résultat.

```
{
    "addon": {
```

```
        "addonName": "coredns",
        "clusterName": "my-cluster",
        "status": "ACTIVE",
        "addonVersion": "v1.11.3-eksbuild.1",
        "health": {
            "issues": []
        },
        "addonArn": "arn:aws:eks:region:111122223333:addon/my-cluster/coredns/
d2c34f06-1111-2222-1eb0-24f64ce37fa4",
        "createdAt": "2023-03-01T16:41:32.442000+00:00",
        "modifiedAt": "2023-03-01T18:16:54.332000+00:00",
        "tags": {},
        "configurationValues": "{\"replicaCount\":3}"
    }
}
```

Mettre à jour le module complémentaire CoreDNS Amazon EKS autogéré

> **Important**
>
> Nous recommandons d'ajouter le type Amazon EKS du module complémentaire à votre cluster au lieu d'utiliser le type autogéré du module complémentaire. Si la différence entre les types ne vous est pas familière, consultez the section called "Modules complémentaires Amazon EKS". Pour plus d'informations sur l'ajout d'un module complémentaire Amazon EKS à votre cluster, consultez the section called "Créer un module complémentaire". Si vous ne pouvez pas utiliser le module complémentaire Amazon EKS, nous vous encourageons à signaler un problème expliquant pourquoi vous ne pouvez pas le faire dans le référentiel GitHub de la feuille de route des conteneurs.

Avant de commencer, consultez les considérations relatives à la mise à niveau. Pour de plus amples informations, consultez the section called "Considérations importantes concernant la mise à niveau de CoreDNS".

1. Vérifiez que le type autogéré du module complémentaire est installé sur votre cluster. Remplacez *my-cluster* par le nom de votre cluster.

```
aws eks describe-addon --cluster-name my-cluster --addon-name coredns --query
 addon.addonVersion --output text
```

Si un message d'erreur est renvoyé, le type autogéré du module complémentaire est installé sur votre cluster. Suivez les étapes restantes de cette procédure. Si un numéro de version est renvoyé, le type de module complémentaire Amazon EKS est installé sur votre cluster. Pour mettre à jour le type Amazon EKS du module complémentaire, suivez la procédure décrite dans Mettre à jour le module complémentaire CoreDNS Amazon EKS, plutôt que d'utiliser cette procédure. Si vous ne connaissez pas les différences entre les types d'extensions, consultez the section called "Modules complémentaires Amazon EKS".

2. Découvrez quelle version de l'image de conteneur est actuellement installée sur votre cluster.

```
kubectl describe deployment coredns -n kube-system | grep Image | cut -d ":" -f 3
```

L'exemple qui suit illustre un résultat.

```
v1.8.7-eksbuild.2
```

3. Si votre version actuelle de CoreDNS est `v1.5.0` ou ultérieure, mais antérieure à la version indiquée dans le tableau des versions de CoreDNS, ignorez cette étape. Si votre version actuelle est antérieure à la version `1.5.0`, vous devez modifier le `ConfigMap` pour que CoreDNS utilise le module complémentaire, plutôt que le module complémentaire proxy.

   a. Ouvrez le fichier `ConfigMap` à l'aide de la commande suivante.

   ```
   kubectl edit configmap coredns -n kube-system
   ```

   b. Remplacez `proxy` dans la ligne suivante par `forward`. Enregistrez le fichier et quittez l'éditeur.

   ```
   proxy . /etc/resolv.conf
   ```

4. Si vous avez initialement déployé votre cluster sur Kubernetes `1.17` ou une version précédente, vous devrez peut-être supprimer une ligne interrompue de votre manifeste CoreDNS.

> **Important**
>
> Vous devez effectuer cette étape avant de passer à la version CoreDNS `1.7.0`, mais il est recommandé de la faire même si vous passez à une version antérieure.

   a. Vérifiez si votre manifeste CoreDNS possède la ligne.

```
kubectl get configmap coredns -n kube-system -o jsonpath='{$.data.Corefile}' |
 grep upstream
```

Si aucune sortie n'est renvoyée, votre manifeste n'a pas la ligne et vous pouvez passer à l'étape suivante pour mettre à jour CoreDNS. Si la sortie est renvoyée, vous devez supprimer la ligne.

b. Modifiez la `ConfigMap` en utilisant la commande suivante, en supprimant la ligne dans le fichier contenant le mot `upstream`. Ne modifiez rien d'autre dans le fichier. Une fois la ligne supprimée, enregistrez les modifications.

```
kubectl edit configmap coredns -n kube-system -o yaml
```

5. Récupérez la version actuelle de votre image CoreDNS :

```
kubectl describe deployment coredns -n kube-system | grep Image
```

L'exemple qui suit illustre un résultat.

```
602401143452.dkr.ecr.region-code.amazonaws.com/eks/coredns:v1.8.7-eksbuild.2
```

6. Si vous effectuez une mise à jour vers CoreDNS `1.8.3` ou une version ultérieure, vous devez ajouter l'autorisation `endpointslices` à `system:coredns` Kubernetes `clusterrole`.

```
kubectl edit clusterrole system:coredns -n kube-system
```

Ajoutez les lignes suivantes sous les lignes d'autorisation existantes dans la section `rules` du fichier.

```
[...]
- apiGroups:
  - discovery.k8s.io
  resources:
  - endpointslices
  verbs:
  - list
  - watch
[...]
```

7. Mettez à jour le module complémentaire CoreDNS en remplaçant *602401143452* et le *code régional* par les valeurs issues du résultat obtenu à l'étape précédente. Remplacez *v1.11.3-eksbuild.1* par la version CoreDNS indiquée dans le tableau des dernières versions pour votre version Kubernetes.

```
kubectl set image deployment.apps/coredns -n kube-system
 coredns=602401143452.dkr.ecr.region-code.amazonaws.com/eks/coredns:v1.11.3-
eksbuild.1
```

   L'exemple qui suit illustre un résultat.

```
deployment.apps/coredns image updated
```

8. Vérifiez à nouveau la version d'image de conteneur pour confirmer qu'elle a été mise à jour vers la version que vous avez spécifiée à l'étape précédente.

```
kubectl describe deployment coredns -n kube-system | grep Image | cut -d ":" -f 3
```

   L'exemple qui suit illustre un résultat.

```
v1.11.3-eksbuild.1
```

Mise à l'échelle des pods CoreDNS pour un trafic DNS élevé

Lorsque vous lancez un cluster Amazon EKS avec au moins un nœud, un déploiement de deux réplicas de l'image CoreDNS est créé par défaut, quel que soit le nombre total de nœuds déployés dans votre cluster. Les pods CoreDNS fournissent une résolution de noms pour tous les pods du cluster. Les applications utilisent la résolution de noms pour se connecter aux pods et services du cluster, ainsi qu'aux services externes situés hors du cluster. À mesure que le nombre de requêtes de résolution de noms (requêtes) émises par les pods augmente, les pods CoreDNS peuvent être surchargés, ce qui provoque un ralentissement et peut entraîner le rejet de certaines requêtes que les pods ne peuvent pas traiter.

Pour gérer cette augmentation de charge sur les pods CoreDNS, il est recommandé de mettre en place un système d'autoscaling pour CoreDNS. Amazon EKS peut gérer automatiquement la mise à l'échelle du déploiement CoreDNS dans le cadre de la version du module complémentaire EKS pour CoreDNS. Cet autoscaler CoreDNS surveille en continu l'état du cluster, y compris le nombre

de nœuds et le nombre de cœurs de processeur. En fonction de ces informations, le contrôleur ajuste dynamiquement le nombre de réplicas du déploiement CoreDNS dans le cluster EKS. Cette fonctionnalité est disponible pour CoreDNS en version `v1.9` et ultérieure. Pour plus d'informations sur les versions compatibles avec la mise à l'échelle automatique de CoreDNS, consultez la section suivante.

Le système gère automatiquement les réplicas CoreDNS à l'aide d'une formule dynamique basée à la fois sur le nombre de nœuds et sur le nombre de cœurs CPU dans le cluster, calculée comme le maximum entre (numberOfNodes divisé par 16) et (numberOfCPUCores divisé par 256). Il évalue la demande sur des périodes de pointe de 10 minutes : il augmente verticalement et immédiatement le nombre de réplicas si nécessaire pour répondre à une charge de requête DNS plus élevée, tout en reduissant verticalement et progressivement le nombre de réplicas de 33 % toutes les trois minutes afin de maintenir la stabilité du système et d'éviter toute interruption.

Il est recommandé d'utiliser cette fonctionnalité conjointement avec les bonnes pratiques de mise à l'échelle automatique des clusters EKS, afin d'améliorer la disponibilité globale des applications et la capacité de mise à l'échelle du cluster.

Prérequis

Pour qu'Amazon EKS puisse mettre à l'échelle votre déploiement CoreDNS, trois prérequis doivent être remplis :

- Vous devez utiliser la version de module complémentaire EKS requise pour CoreDNS.
- Votre cluster doit exécuter au minimum les versions de cluster et de plateforme requises.
- Votre cluster doit exécuter au minimum la version requise du module complémentaire EKS pour CoreDNS.

Version minimale du cluster

La mise à l'échelle automatique de CoreDNS est effectuée par un nouveau composant intégré au plan de contrôle du cluster, géré directement par Amazon EKS. Pour cette raison, vous devez mettre à niveau votre cluster vers une version d'EKS qui prend en charge la version de plateforme minimale intégrant ce nouveau composant.

Un nouveau cluster Amazon EKS. Pour en déployer un, consultez Démarrer. Le cluster doit exécuter l'une des versions de Kubernetes et de plateforme indiquées dans le tableau suivant, ou une version ultérieure. Toute version de Kubernetes et de la plateforme ultérieure à celles

indiquées est également prise en charge. Vous pouvez vérifier la version actuelle de Kubernetes en remplaçant *my-cluster* par le nom de votre cluster dans la commande suivante, puis en exécutant la commande modifiée :

```
aws eks describe-cluster --name my-cluster --query cluster.version --output text
```

Version de Kubernetes	Version de la plateforme
Non répertoriée	Toutes les versions de plateforme
1.29.3	eks.7
1.28.8	eks.13
1.27.12	eks.17
1.26.15	eks.18

> **Note**
>
> Toutes les versions de plateforme des versions ultérieures de Kubernetes sont également prises en charge, par exemple Kubernetes en version 1.30 à partir de eks.1 et ultérieure est compatible.

Version minimale du module complémentaire EKS

Version de Kubernetes	1.29	1.28
	v1.11.1-eksbuild.9	v1.10.1-eksbuild.11

Configuration de la mise à l'échelle automatique de CoreDNS dans la AWS Management Console

1. Assurez-vous que votre cluster exécute au moins la version minimale requise.

   Amazon EKS met automatiquement à niveau les clusters entre les versions de plateforme d'une même version de Kubernetes, et vous ne pouvez pas lancer cette mise à jour manuellement.

En revanche, vous pouvez mettre à niveau votre cluster vers la version suivante de Kubernetes. Le cluster sera alors mis à niveau vers cette version K8s ainsi que vers la dernière version de plateforme disponible.

Les nouvelles versions de Kubernetes introduisent parfois des modifications importantes. Il est donc recommandé de tester le comportement de vos applications dans un cluster distinct exécutant la nouvelle version de Kubernetes, avant de mettre à jour vos clusters de production.

Pour mettre à niveau un cluster vers une nouvelle version de Kubernetes, suivez la procédure décrite dans Mise à jour un cluster existant vers une nouvelle version de Kubernetes.

2. Assurez-vous que vous utilisez le module complémentaire EKS pour CoreDNS, et non un déploiement CoreDNS autogéré.

   Selon l'outil avec lequel vous avez créé votre cluster, le type de module complémentaire Amazon EKS peut ne pas être actuellement installé sur votre cluster. Pour vérifier quel type de module complémentaire est installé sur votre cluster, exécutez la commande suivante. Remplacez `my-cluster` par le nom de votre cluster.

   ```
   aws eks describe-addon --cluster-name my-cluster --addon-name coredns --query
    addon.addonVersion --output text
   ```

   Si un numéro de version est renvoyé, cela signifie que vous avez installé sur votre cluster le module complémentaire de type Amazon EKS, et vous pouvez passer à l'étape suivante. Si une erreur est renvoyée, cela signifie que le type de module complémentaire Amazon EKS n'est pas installé sur votre cluster. Suivez les étapes restantes de la procédure Création du module complémentaire CoreDNS Amazon EKS afin de remplacer la version autogérée par le module complémentaire Amazon EKS.

3. Assurez-vous que votre module complémentaire EKS pour CoreDNS est d'une version égale ou supérieure à la version minimale requise.

   Déterminez la version du module complémentaire actuellement installée sur votre cluster. Vous pouvez le vérifier dans la AWS Management Console ou en exécutant la commande suivante :

   ```
   kubectl describe deployment coredns --namespace kube-system | grep coredns: | cut -
   d : -f 3
   ```

   L'exemple qui suit illustre un résultat.

```
v1.10.1-eksbuild.13
```

Comparez cette version avec la version minimale du module complémentaire EKS indiquée dans la section précédente. Si nécessaire, mettez à niveau le module complémentaire EKS vers une version supérieure en suivant la procédure Mise à jour du module complémentaire CoreDNS Amazon EKS.

4. Dans la section Paramètres de configuration facultatifs du module complémentaire EKS, ajoutez la configuration de mise à l'échelle automatique.
   a. Ouvrez la console Amazon EKS.
   b. Dans le panneau de navigation de gauche, sélectionnez Clusters, puis sélectionnez le nom du cluster pour lequel vous souhaitez configurer le module complémentaire.
   c. Choisissez l'onglet Modules complémentaires.
   d. Cochez la case située dans le coin supérieur droit du bloc CoreDNS, puis sélectionnez Modifier.
   e. Sur la page Configurer CoreDNS :
      i. Sélectionnez la version que vous souhaitez utiliser. Il est recommandé de conserver la même version que celle de l'étape précédente et d'effectuer la mise à jour de la version et de la configuration séparément.
      ii. Sélectionnez Paramètres de configuration facultatifs.
      iii. Dans le champ Valeurs de configuration, saisissez la clé JSON `"autoscaling":` et la valeur d'un objet JSON imbriqué comportant une clé `"enabled":` et une valeur `true`. Le texte obtenu doit être un objet JSON valide. Si cette clé et cette valeur sont les seules données de la zone de texte, entourez-les d'accolades `{ }`. L'exemple suivant montre la mise à l'échelle automatique activée :

```
{
  "autoScaling": {
    "enabled": true
  }
}
```

      iv. (Facultatif) Vous pouvez spécifier les valeurs minimales et maximales auxquelles l'autoscaling peut mettre à l'échelle le nombre de pods CoreDNS.

      L'exemple suivant montre la mise à l'échelle automatique activée ainsi que toutes les clés facultatives ayant des valeurs définies. Il est recommandé que le nombre minimal de pods

CoreDNS soit toujours supérieur à 2, afin d'assurer la résilience du service DNS dans le cluster.

```
{
  "autoScaling": {
    "enabled": true,
    "minReplicas": 2,
    "maxReplicas": 10
  }
}
```

f. Pour appliquer la nouvelle configuration en remplaçant les pods CoreDNS, sélectionnez Enregistrer les modifications.

Amazon EKS applique les modifications au module complémentaire EKS à l'aide d'un déploiement Kubernetes progressif pour CoreDNS. Vous pouvez suivre l'état du déploiement dans l'Historique des mises à jour du module complémentaire, dans AWS Management Console, ainsi qu'à l'aide de `kubectl rollout status deployment/coredns --namespace kube-system`.

L'outil `kubectl rollout` propose les commandes suivantes :

```
kubectl rollout

history  -- View rollout history
pause    -- Mark the provided resource as paused
restart  -- Restart a resource
resume   -- Resume a paused resource
status   -- Show the status of the rollout
undo     -- Undo a previous rollout
```

Si le déploiement prend trop de temps, Amazon EKS annule automatiquement le déploiement, et un message indiquant un type de Mise à jour du module complémentaire avec un état Échec est ajouté à l'Historique des mises à jour du module complémentaire. Pour analyser les problèmes, commencez par consulter l'historique du déploiement, puis exécutez la commande `kubectl logs` sur un pod CoreDNS afin d'afficher les journaux CoreDNS.

5. Si la nouvelle entrée de l'Historique des mises à jour affiche un état Réussi, cela signifie que le déploiement est terminé et que le module complémentaire utilise la nouvelle configuration sur l'ensemble des pods CoreDNS. Lorsque le nombre de nœuds et de cœurs du processeur du

cluster varie, Amazon EKS ajuste dynamiquement le nombre de réplicas du déploiement CoreDNS en conséquence.

Configuration de la mise à l'échelle automatique de CoreDNS dans l'interface de ligne de commande AWS

1. Assurez-vous que votre cluster exécute au moins la version minimale requise.

   Amazon EKS met automatiquement à niveau les clusters entre les versions de plateforme d'une même version de Kubernetes, et vous ne pouvez pas lancer cette mise à jour manuellement. En revanche, vous pouvez mettre à niveau votre cluster vers la version suivante de Kubernetes. Le cluster sera alors mis à niveau vers cette version K8s ainsi que vers la dernière version de plateforme disponible.

   Les nouvelles versions de Kubernetes introduisent parfois des modifications importantes. Il est donc recommandé de tester le comportement de vos applications dans un cluster distinct exécutant la nouvelle version de Kubernetes, avant de mettre à jour vos clusters de production.

   Pour mettre à niveau un cluster vers une nouvelle version de Kubernetes, suivez la procédure décrite dans Mise à jour un cluster existant vers une nouvelle version de Kubernetes.
2. Assurez-vous que vous utilisez le module complémentaire EKS pour CoreDNS, et non un déploiement CoreDNS autogéré.

   Selon l'outil avec lequel vous avez créé votre cluster, le type de module complémentaire Amazon EKS peut ne pas être actuellement installé sur votre cluster. Pour vérifier quel type de module complémentaire est installé sur votre cluster, exécutez la commande suivante. Remplacez `my-cluster` par le nom de votre cluster.

   ```
   aws eks describe-addon --cluster-name my-cluster --addon-name coredns --query
    addon.addonVersion --output text
   ```

   Si un numéro de version est renvoyé, le type de module complémentaire Amazon EKS est installé sur votre cluster. Si une erreur est renvoyée, cela signifie que le type de module complémentaire Amazon EKS n'est pas installé sur votre cluster. Suivez les étapes restantes de la procédure Création du module complémentaire CoreDNS Amazon EKS afin de remplacer la version autogérée par le module complémentaire Amazon EKS.
3. Assurez-vous que votre module complémentaire EKS pour CoreDNS est d'une version égale ou supérieure à la version minimale requise.

Déterminez la version du module complémentaire actuellement installée sur votre cluster. Vous pouvez le vérifier dans la AWS Management Console ou en exécutant la commande suivante :

```
kubectl describe deployment coredns --namespace kube-system | grep coredns: | cut -
d : -f 3
```

L'exemple qui suit illustre un résultat.

```
v1.10.1-eksbuild.13
```

Comparez cette version avec la version minimale du module complémentaire EKS indiquée dans la section précédente. Si nécessaire, mettez à niveau le module complémentaire EKS vers une version supérieure en suivant la procédure Mise à jour du module complémentaire CoreDNS Amazon EKS.

4. Dans la section Paramètres de configuration facultatifs du module complémentaire EKS, ajoutez la configuration de mise à l'échelle automatique.

   Exécutez la commande AWS CLI suivante. Remplacez `my-cluster` par le nom de votre cluster et l'ARN du rôle IAM par le rôle que vous utilisez.

   ```
   aws eks update-addon --cluster-name my-cluster --addon-name coredns \
       --resolve-conflicts PRESERVE --configuration-values '{"autoScaling":
   {"enabled":true}}'
   ```

   Amazon EKS applique les modifications au module complémentaire EKS à l'aide d'un déploiement Kubernetes progressif pour CoreDNS. Vous pouvez suivre l'état du déploiement dans l'Historique des mises à jour du module complémentaire, dans AWS Management Console, ainsi qu'à l'aide de `kubectl rollout status deployment/coredns --namespace kube-system`.

   L'outil `kubectl rollout` propose les commandes suivantes :

   ```
   kubectl rollout

   history  -- View rollout history
   pause    -- Mark the provided resource as paused
   restart  -- Restart a resource
   resume   -- Resume a paused resource
   status   -- Show the status of the rollout
   ```

```
undo     -- Undo a previous rollout
```

Si le déploiement prend trop de temps, Amazon EKS annule automatiquement le déploiement, et un message indiquant un type de Mise à jour du module complémentaire avec un état Échec est ajouté à l'Historique des mises à jour du module complémentaire. Pour analyser les problèmes, commencez par consulter l'historique du déploiement, puis exécutez la commande `kubectl logs` sur un pod CoreDNS afin d'afficher les journaux CoreDNS.

5. (Facultatif) Vous pouvez spécifier les valeurs minimales et maximales auxquelles l'autoscaling peut mettre à l'échelle le nombre de pods CoreDNS.

   L'exemple suivant montre la mise à l'échelle automatique activée ainsi que toutes les clés facultatives ayant des valeurs définies. Il est recommandé que le nombre minimal de pods CoreDNS soit toujours supérieur à 2, afin d'assurer la résilience du service DNS dans le cluster.

   ```
   aws eks update-addon --cluster-name my-cluster --addon-name coredns \
       --resolve-conflicts PRESERVE --configuration-values '{"autoScaling":
   {"enabled":true,"minReplicas":2,"maxReplicas":10}}'
   ```

6. Pour vérifier l'état de la mise à jour du module complémentaire, exécutez la commande suivante :

   ```
   aws eks describe-addon --cluster-name my-cluster --addon-name coredns
   ```

   Si vous voyez cette ligne :`"status": "ACTIVE"`, cela signifie que le déploiement est terminé et que le module complémentaire utilise la nouvelle configuration dans tous les pods CoreDNS. Lorsque le nombre de nœuds et de cœurs du processeur du cluster varie, Amazon EKS ajuste dynamiquement le nombre de réplicas du déploiement CoreDNS en conséquence.

Surveillance de la résolution DNS Kubernetes à l'aide des métriques CoreDNS

Le module complémentaire CoreDNS pour EKS expose les métriques CoreDNS sur le port `9153`, au format Prometheus, dans le service `kube-dns`. Vous pouvez utiliser Prometheus, l'agent Amazon CloudWatch ou tout autre système compatible pour récupérer (collecter) ces métriques.

Pour un exemple de configuration de récupération compatible à la fois avec Prometheus et l'agent CloudWatch, consultez Configuration de l'agent CloudWatch pour Prometheus dans le Guide de l'utilisateur Amazon CloudWatch.

Gérer **kube-proxy** dans les Clusters Amazon EKS

> **Tip**
>
> Avec le mode automatique Amazon EKS, il n'est pas nécessaire d'installer ou de mettre à niveau de modules complémentaires de mise en réseau. Le mode automatique inclut des fonctionnalités de mise en réseau des pods et d'équilibrage de charge.
> Pour de plus amples informations, veuillez consulter Mode automatique EKS.

Nous recommandons d'ajouter le type Amazon EKS du module complémentaire à votre cluster au lieu d'utiliser le type autogéré du module complémentaire. Si la différence entre les types ne vous est pas familière, consultez the section called "Modules complémentaires Amazon EKS". Pour plus d'informations sur l'ajout d'un module complémentaire Amazon EKS à votre cluster, consultez the section called "Créer un module complémentaire". Si vous ne parvenez pas à utiliser le module complémentaire Amazon EKS, nous vous encourageons à signaler les raisons pour lesquelles vous ne pouvez pas utiliser le module complémentaire Amazon EKS dans le GitHub référentiel de feuilles de route pour les conteneurs.

Le module complémentaire `kube-proxy` est déployé sur chaque nœud Amazon EC2 de votre cluster Amazon EKS. Il maintient les règles du réseau sur vos nœuds et permet la communication du réseau à vos pods. Le module complémentaire n'est pas déployé sur les nœuds Fargate de votre cluster. Pour plus d'informations, consultez kube-proxy dans la documentation Kubernetes.

Installer en tant que module complémentaire Amazon EKS

Versions **kube-proxy**

Le tableau suivant répertorie la dernière version du type d'add-on Amazon EKS pour chaque version de Kubernetes.

Version de Kubernetes	Version de l'**kube-proxy**
1,35	v1.35.0-eksbuild.2
1,34	v1.34.3-eksbuild.2
1,33	v1.33.7-eksbuild.2

Version de Kubernetes	Version de l'**kube-proxy**
1,32	v1.32.11-eksbuild.2
1,31	v1.31.14-eksbuild.2
1,30	v1.30.14-eksbuild.20
1,29	v1.29.15-eksbuild.28

> **Note**
>
> Une version antérieure de la documentation était incorrecte. Les versions `kube-proxy`, `v1.28.5`, `v1.27.9` et `v1.26.12` ne sont pas disponibles.
> Si vous gérez vous-même ce module complémentaire, les versions répertoriées dans le tableau peuvent ne pas être les mêmes que les versions autogérées disponibles.

Image de conteneur **kube-proxy**

L'image du conteneur `kube-proxy` est basée sur une image de base minimale gérée par Amazon EKS Distro, qui contient un nombre minimal de paquets et ne dispose pas de shells. Pour plus d'informations, consultez Amazon EKS Distro.

Le tableau suivant répertorie la dernière version disponible de l'image de conteneur `kube-proxy` autogérée pour chaque version de cluster Amazon EKS.

Version	kube-proxy
1,35	v1.35.0-eksbuild.2
1,34	v1.34.1-eksbuild.2
1,33	v1.33.5-minimal-eksbuild.2
1,32	v1.32.9-minimal-eksbuild.2
1,31	v1.31.13-minimal-eksbuild.2

Version	kube-proxy
1,30	v1.30.14-minimal-eksbuild.8
1,29	v1.29.15-minimal-eksbuild.16

Lorsque vous mettez à jour un type de module complémentaire Amazon EKS, vous spécifiez une version valide du module complémentaire Amazon EKS, qui peut ne pas être répertoriée dans ce tableau. Cela est dû au fait que les versions du module complémentaire Amazon EKS ne correspondent pas toujours aux versions d'image de conteneur spécifiées lors de la mise à jour du type autogéré de ce module complémentaire. Lorsque vous mettez à jour le type autogéré de ce module complémentaire, vous spécifiez une version d'image de conteneur valide répertoriée dans ce tableau.

Mettre à jour le module complémentaire autogéré Kubernetes **`kube-proxy`**

> **Important**
>
> Nous recommandons d'ajouter le type Amazon EKS du module complémentaire à votre cluster au lieu d'utiliser le type autogéré du module complémentaire. Si la différence entre les types ne vous est pas familière, consultez the section called "Modules complémentaires Amazon EKS". Pour plus d'informations sur l'ajout d'un module complémentaire Amazon EKS à votre cluster, consultez the section called "Créer un module complémentaire". Si vous ne pouvez pas utiliser le module complémentaire Amazon EKS, nous vous encourageons à signaler un problème expliquant pourquoi vous ne pouvez pas le faire dans le référentiel GitHub de la feuille de route des conteneurs.

Prérequis

- Un cluster Amazon EKS existant. Pour en déployer un, consultez Démarrer.

Considérations

- `Kube-proxy` sur un cluster Amazon EKS a la même politique de compatibilité et de décalage que Kubernetes. Découvrez comment vérifier la compatibilité de la version du module complémentaire Amazon EKS avec un cluster.

1. Vérifiez que le type autogéré du module complémentaire est installé sur votre cluster. Remplacez *my-cluster* par le nom de votre cluster.

```
aws eks describe-addon --cluster-name my-cluster --addon-name kube-proxy --query
 addon.addonVersion --output text
```

   Si un message d'erreur est renvoyé, le type autogéré du module complémentaire est installé sur votre cluster. Les étapes restantes de cette rubrique concernent la mise à jour du type autogéré du module complémentaire. Si un numéro de version est renvoyé, le type de module complémentaire Amazon EKS est installé sur votre cluster. Pour le mettre à jour, suivez la procédure décrite dans la section Mise à jour d'un module complémentaire Amazon EKS, plutôt que celle décrite dans cette rubrique. Si vous ne connaissez pas les différences entre les types d'extensions, consultez the section called "Modules complémentaires Amazon EKS".

2. Découvrez quelle version de l'image de conteneur est actuellement installée sur votre cluster.

```
kubectl describe daemonset kube-proxy -n kube-system | grep Image
```

   L'exemple qui suit illustre un résultat.

```
Image:    602401143452.dkr.ecr.region-code.amazonaws.com/eks/kube-proxy:v1.29.1-
eksbuild.2
```

   Dans l'exemple de sortie, *v1.29.1-eksbuild.2* est la version installée sur le cluster.

3. Mettez à jour le module complémentaire `kube-proxy` en remplaçant *602401143452* et le *region-code* par les valeurs obtenues à l'étape précédente. Remplacez *v1.30.6-eksbuild.3* par la version `kube-proxy` indiquée dans le tableau des dernières versions disponibles de l'image de conteneur kube-proxy autogérée pour chaque version de cluster Amazon EKS.

   > **Important**
   >
   > Les manifestes pour chaque type d'image sont différents et ne sont pas compatibles entre les types d'images par défaut ou minimales. Vous devez utiliser le même type d'image que l'image précédente afin que le point d'entrée et les arguments correspondent.

```
kubectl set image daemonset.apps/kube-proxy -n kube-system kube-
proxy=602401143452.dkr.ecr.region-code.amazonaws.com/eks/kube-proxy:v1.30.6-
eksbuild.3
```

L'exemple qui suit illustre un résultat.

```
daemonset.apps/kube-proxy image updated
```

4. Vérifiez que la nouvelle version est maintenant installée sur votre cluster.

```
kubectl describe daemonset kube-proxy -n kube-system | grep Image | cut -d ":" -f 3
```

L'exemple qui suit illustre un résultat.

```
v1.30.0-eksbuild.3
```

5. Si vous utilisez des nœuds `x86` et `Arm` dans le même cluster et que votre cluster a été déployé avant le 17 août 2020. Ensuite, modifiez votre manifeste `kube-proxy` pour inclure un sélecteur de noeud pour plusieurs architectures matérielles avec la commande suivante. Il s'agit d'une opération ponctuelle. Une fois que vous avez ajouté le sélecteur à votre manifeste, vous n'avez pas besoin de l'ajouter chaque fois que vous mettez à jour le module complémentaire. Si votre cluster a été déployé à partir du 17 août 2020, `kube-proxy` est déjà compatible avec les architectures multiples.

```
kubectl edit -n kube-system daemonset/kube-proxy
```

Ajoutez le sélecteur de nœuds suivant au fichier dans l'éditeur, puis enregistrez le fichier. Pour obtenir un exemple d'inclusion de ce texte dans l'éditeur, consultez le fichier manifeste CNI sur GitHub. Cela permet à Kubernetes d'extraire l'image matérielle correcte en fonction de l'architecture matérielle du nœud.

```
- key: "kubernetes.io/arch"
  operator: In
  values:
  - amd64
  - arm64
```

6. Si votre cluster a été créé à l'origine avec Kubernetes version `1.14` ou ultérieure, vous pouvez ignorer cette étape, car `kube-proxy` inclut déjà cette `Affinity Rule`. Si vous avez initialement créé un cluster Amazon EKS avec la version Kubernetes `1.13` ou une version antérieure et que vous avez l'intention d'utiliser des nœuds Fargate dans votre cluster, modifiez votre manifeste `kube-proxy` afin d'inclure une règle `NodeAffinity` empêchant les pods `kube-proxy` d'être planifiés sur les nœuds Fargate. Il s'agit d'une modification unique. Une fois que vous avez ajouté `Affinity Rule` à votre manifeste, vous n'avez pas besoin de l'ajouter chaque fois que vous mettez à niveau votre cluster. Modifiez votre Daemonset `kube-proxy`.

```
kubectl edit -n kube-system daemonset/kube-proxy
```

Ajoutez le `Affinity Rule` qui suit à la section DaemonSet `spec` du fichier dans l'éditeur, puis enregistrez le fichier. Pour obtenir un exemple d'inclusion de ce texte dans l'éditeur, consultez le fichier manifeste CNI sur GitHub.

```
- key: eks.amazonaws.com/compute-type
  operator: NotIn
  values:
  - fargate
```

Découvrez comment déployer des charges de travail et des modules complémentaires sur Amazon EKS

Vos applications sont déployées dans des conteneurs, qui sont déployés dans des pods dans Kubernetes. Un pod comprend un ou plusieurs conteneurs. En règle générale, un ou plusieurs pods qui fournissent le même service sont déployés dans un service Kubernetes. Une fois que vous avez déployé plusieurs pods qui fournissent le même service, vous pouvez :

- Afficher des informations sur les applications s'exécutant sur chacun de vos clusters à l'aide de AWS Management Console.
- Augmentez ou réduisez verticalement les pods à l'aide du Kubernetes Vertical Pod Autoscaler.
- Ajustez horizontalement le nombre de pods nécessaires pour répondre à la demande à la hausse ou à la baisse grâce à l'autoscaler horizontal de pods Kubernetes.
- Créer un dispositif d'équilibrage de charge de réseau externe (pour les pods accessibles depuis Internet) ou interne (pour les pods privés) afin d'équilibrer le trafic réseau entre les pods. L'équilibreur de charge achemine le trafic à la couche 4 du modèle OSI.
- Création d'un Application Load Balancer pour équilibrer le trafic des applications entre les pods. L'Application Load Balancer achemine le trafic à la couche 7 du modèle OSI.
- Si vous débutez avec Kubernetes, cette rubrique vous aide. Déployez un exemple d'application.
- Vous pouvez restreindre les adresses IP qui peuvent être affectées à un service avec `externalIPs`.

Déployer un exemple d'application sur Linux

Dans cette rubrique, vous allez déployer un exemple d'application sur votre cluster sur des nœuds Linux.

Prérequis

- Un cluster Kubernetes existant avec au moins un nœud. Si vous ne disposez pas d'un cluster Amazon EKS existant, vous pouvez en déployer un à l'aide de l'un des guides disponibles dans Démarrer.
- `Kubectl` installé sur votre ordinateur. Pour de plus amples informations, consultez the section called "Configurer kubectl et eksctl".

- `Kubectl` configuré pour communiquer avec votre cluster. Pour de plus amples informations, consultez the section called "Accès au cluster avec kubectl".
- Si vous prévoyez de déployer votre charge de travail d'exemple sur Fargate, vous devez disposer d'un profil Fargate existant qui inclut le même espace de noms créé dans ce didacticiel, à savoir `eks-sample-app`, à moins que vous ne changiez le nom. Si vous avez créé un cluster à l'aide de l'un des guides disponibles dans Démarrer, vous devrez créer un nouveau profil ou ajouter l'espace de noms à votre profil existant, car le profil créé dans les guides de démarrage ne spécifie pas l'espace de noms utilisé dans ce tutoriel. Votre VPC doit également disposer d'au moins un sous-réseau privé.

Bien que de nombreuses variables soient modifiables dans les étapes suivantes, nous vous recommandons de ne modifier les valeurs des variables qu'aux endroits spécifiés. Une fois que vous aurez mieux compris les pods, les déploiements et les services Kubernetes, vous pourrez essayer de modifier d'autres valeurs.

Création d'un espace de noms

Un espace de noms vous permet de regrouper des ressources dans Kubernetes. Pour plus d'informations, consultez Espaces de noms dans la documentation Kubernetes. Si vous prévoyez de déployer votre application exemple sur Simplifier la gestion des ressources de calcul avec AWS Fargate, assurez-vous que la valeur du `namespace` dans votre Définir les pods qui utilisent AWS Fargate lors de leur lancement est `eks-sample-app`.

```
kubectl create namespace eks-sample-app
```

Créer un déploiement Kubernetes

Créer un déploiement Kubernetes. Cet exemple de déploiement extrait une image de conteneur d'un référentiel public et en déploie trois réplicas (pods individuels) sur votre cluster. Pour en savoir plus, consultez Déploiements dans la documentation Kubernetes.

1. Enregistrez le contenu suivant dans un fichier nommé `eks-sample-deployment.yaml`. Les conteneurs de l'application exemple n'utilisent pas de stockage réseau, mais vous pouvez avoir des applications qui en ont besoin. Pour de plus amples informations, consultez Stockage de données d'application.
   - Le `amd64` ou `arm64` `values` sous la clé `kubernetes.io/arch` signifient que l'application peut être déployée sur l'une ou l'autre architecture matérielle (si vous avez les deux dans votre

cluster). Cela est possible car cette image est une image multi-architecture, mais toutes ne le sont pas. Vous pouvez déterminer l'architecture matérielle prise en charge par l'image en consultant les détails de l'image dans le référentiel à partir duquel vous l'extrayez. Lorsque vous déployez des images qui ne prennent pas en charge un type d'architecture matérielle ou sur lequel vous ne voulez pas déployer l'image, supprimez ce type du manifeste. Pour plus d'informations, consultez Étiquettes, annotations et rejets connus dans la documentation Kubernetes.

- Le `kubernetes.io/os: linux` `nodeSelector` signifie que si vous possédez des nœuds Linux et Windows (par exemple) dans votre cluster, l'image ne sera déployée que sur des nœuds Linux. Pour plus d'informations, consultez Étiquettes, annotations et rejets connus dans la documentation Kubernetes.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: eks-sample-linux-deployment
  namespace: eks-sample-app
  labels:
    app: eks-sample-linux-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: eks-sample-linux-app
  template:
    metadata:
      labels:
        app: eks-sample-linux-app
    spec:
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: kubernetes.io/arch
                operator: In
                values:
                - amd64
                - arm64
      containers:
      - name: nginx
```

```
        image: public.ecr.aws/nginx/nginx:1.23
        ports:
        - name: http
          containerPort: 80
        imagePullPolicy: IfNotPresent
      nodeSelector:
        kubernetes.io/os: linux
```

2. Appliquer le manifeste de déploiement à votre cluster.

```
kubectl apply -f eks-sample-deployment.yaml
```

Création d'un service

Un service vous permet d'accéder à toutes les réplicas via une seule adresse IP ou un seul nom. Pour plus d'informations, consultez Service dans la documentation Kubernetes. Bien que cela ne soit pas implémenté dans l'application exemple, si vous avez des applications qui doivent interagir avec d'autres services AWS, nous vous recommandons de créer des comptes de service Kubernetes pour vos pods et de les associer à des comptes AWS IAM. En spécifiant des comptes de service, vos pods ne disposent que des autorisations minimales que vous spécifiez pour qu'ils puissent interagir avec d'autres services. Pour de plus amples informations, consultez the section called "Informations d'identification auprès de l'IRSA".

1. Enregistrez le contenu suivant dans un fichier nommé `eks-sample-service.yaml`. Kubernetes attribue au service sa propre adresse IP qui n'est accessible que depuis l'intérieur du cluster. Pour accéder au service depuis l'extérieur de votre cluster, déployez l'AWS Load Balancer Controller pour équilibrer la charge du trafic d'application ou de réseau vers le service.

```
apiVersion: v1
kind: Service
metadata:
  name: eks-sample-linux-service
  namespace: eks-sample-app
  labels:
    app: eks-sample-linux-app
spec:
  selector:
    app: eks-sample-linux-app
  ports:
    - protocol: TCP
```

```
    port: 80
    targetPort: 80
```

2. Appliquer le manifeste du service à votre cluster.

```
kubectl apply -f eks-sample-service.yaml
```

Vérifier les ressources créées

1. Affichez toutes les ressources présentes dans l'espace de noms `eks-sample-app`.

```
kubectl get all -n eks-sample-app
```

L'exemple qui suit illustre un résultat.

```
NAME                                                READY   STATUS    RESTARTS   AGE
pod/eks-sample-linux-deployment-65b7669776-m6qxz    1/1     Running   0          27m
pod/eks-sample-linux-deployment-65b7669776-mmxvd    1/1     Running   0          27m
pod/eks-sample-linux-deployment-65b7669776-qzn22    1/1     Running   0          27m

NAME                                TYPE         CLUSTER-IP      EXTERNAL-IP   PORT(S)
   AGE
service/eks-sample-linux-service    ClusterIP    10.100.74.8     <none>        80/TCP
   32m

NAME                                            READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/eks-sample-linux-deployment 3/3     3            3           27m

NAME                                                       DESIRED   CURRENT   READY
 AGE
replicaset.apps/eks-sample-linux-deployment-776d8f8fd8    3         3         3
 27m
```

Dans la sortie, vous voyez le service et le déploiement spécifiés dans l'exemple de manifestes déployés dans les étapes précédentes. Vous voyez aussi trois pods. Ceci est dû au fait que `3 replicas` ont été spécifiés dans l'exemple de manifeste. Pour plus d'informations sur les pods, consultez pods dans la documentation Kubernetes. Kubernetes crée automatiquement la ressource `replicaset`, même si elle n'est pas spécifiée dans les manifestes d'exemple.

Pour plus d'informations sur `ReplicaSets`, veuillez consulter la section ReplicaSet dans la documentation Kubernetes.

> **Note**
>
> Kubernetes conserve le nombre de réplicas spécifié dans le manifeste. S'il s'agissait d'un déploiement en production et que vous vouliez que Kubernetes fasse mettre à l'échelle horizontalement le nombre de réplicas ou verticalement les ressources de calcul pour les pods, utilisez les options Évoluer les déploiements de pods avec Horizontal Pod Autoscaler et Ajuster les ressources des pods avec Vertical Pod Autoscaler pour ce faire.

2. Affichez les détails du service déployé.

```
kubectl -n eks-sample-app describe service eks-sample-linux-service
```

L'exemple qui suit illustre un résultat.

```
Name:              eks-sample-linux-service
Namespace:         eks-sample-app
Labels:            app=eks-sample-linux-app
Annotations:       <none>
Selector:          app=eks-sample-linux-app
Type:              ClusterIP
IP Families:       <none>
IP:                10.100.74.8
IPs:               10.100.74.8
Port:              <unset>  80/TCP
TargetPort:        80/TCP
Endpoints:         192.168.24.212:80,192.168.50.185:80,192.168.63.93:80
Session Affinity:  None
Events:            <none>
```

Dans la sortie précédente, la valeur de `IP:` est une adresse IP unique accessible depuis n'importe quel nœud ou pod du cluster, mais inaccessible depuis l'extérieur du cluster. Les valeurs de `Endpoints` sont des adresses IP attribuées depuis votre VPC aux pods qui font partie du service.

3. Visualisez les détails de l'un des pods listés en sortie lorsque vous avez visualisé l'espace de noms dans une étape précédente. Remplacez *`776d8f8fd8-78w66`* par la valeur renvoyée pour l'un de vos pods.

```
kubectl -n eks-sample-app describe pod eks-sample-linux-deployment-65b7669776-m6qxz
```

Exemple de sortie abrégée

```
Name:         eks-sample-linux-deployment-65b7669776-m6qxz
Namespace:    eks-sample-app
Priority:     0
Node:         ip-192-168-45-132.us-west-2.compute.internal/192.168.45.132
[...]
IP:           192.168.63.93
IPs:
  IP:           192.168.63.93
Controlled By:  ReplicaSet/eks-sample-linux-deployment-65b7669776
[...]
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
[...]
Events:
  Type    Reason     Age    From
 Message
  ----    ------     ----   ----
 -------
  Normal  Scheduled  3m20s  default-scheduler
 Successfully assigned eks-sample-app/eks-sample-linux-deployment-65b7669776-m6qxz to
 ip-192-168-45-132.us-west-2.compute.internal
[...]
```

Dans la sortie précédente, la valeur de `IP:` est une adresse IP unique attribuée au pod à partir du bloc CIDR attribué au sous-réseau dans lequel se trouve le nœud. Si vous préférez attribuer aux pods des adresses IP à partir de différents blocs d'adresse CIDR, vous pouvez modifier le comportement par défaut. Pour de plus amples informations, consultez the section called "Mise en réseau personnalisée". Vous pouvez également voir que le planificateur Kubernetes a planifié le pod sur le Node avec l'adresse IP *192.168.45.132*.

> **Tip**
>
> Au lieu d'utiliser la ligne de commande, vous pouvez afficher de nombreux détails sur les pods, les services, les déploiements et d'autres ressources Kubernetes dans la AWS Management Console. Pour de plus amples informations, consultez the section called "Accéder aux ressources du cluster".

Exécuter un shell sur un pod

1. Exécutez un shell sur le pod que vous avez décrit à l'étape précédente, en remplaçant *65b7669776-m6qxz* par l'ID de l'un de vos pods.

```
kubectl exec -it eks-sample-linux-deployment-65b7669776-m6qxz -n eks-sample-app -- /
bin/bash
```

2. À partir du shell du pod, affichez la sortie du serveur web qui a été installé avec votre déploiement à l'étape précédente. Vous devez uniquement spécifier le nom du service. Elle est résolue en adresse IP du service par CoreDNS, qui est déployé par défaut avec un cluster Amazon EKS.

```
curl eks-sample-linux-service
```

   L'exemple qui suit illustre un résultat.

```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
[...]
```

3. À partir du shell du pod, affichez le serveur DNS pour le pod.

```
cat /etc/resolv.conf
```

   L'exemple qui suit illustre un résultat.

```
nameserver 10.100.0.10
```

```
search eks-sample-app.svc.cluster.local svc.cluster.local cluster.local us-
west-2.compute.internal
options ndots:5
```

Dans la sortie précédente, `10.100.0.10` est automatiquement affecté en tant que `nameserver` pour tous les pods déployés dans le cluster.

4. Déconnectez-vous du pod en tapant `exit`.
5. Une fois que vous avez terminé avec l'application exemple, vous pouvez supprimer l'espace de noms, le service et le déploiement exemple à l'aide de la commande suivante.

```
kubectl delete namespace eks-sample-app
```

Étapes suivantes

Après avoir déployé l'application exemple, vous pourriez vouloir essayer certains des exercices suivants :

- Router l'application et le trafic HTTP avec des Application Load Balancer
- Router le trafic TCP et UDP avec des Network Load Balancer

Déployer un exemple d'application sur Windows

Dans cette rubrique, vous allez déployer un exemple d'application sur votre cluster sur des nœuds Windows.

Prérequis

- Un cluster Kubernetes existant avec au moins un nœud. Si vous ne disposez pas d'un cluster Amazon EKS existant, vous pouvez en déployer un à l'aide de l'un des guides disponibles dans Démarrer. Vous devez avoir activé la prise en charge Windows pour votre cluster et disposer d'au moins un nœud Amazon EC2 Windows.
- `Kubectl` installé sur votre ordinateur. Pour de plus amples informations, consultez the section called "Configurer kubectl et eksctl".
- `Kubectl` configuré pour communiquer avec votre cluster. Pour de plus amples informations, consultez the section called "Accès au cluster avec kubectl".

- Si vous prévoyez de déployer votre charge de travail d'exemple sur Fargate, vous devez disposer d'un profil Fargate existant qui inclut le même espace de noms créé dans ce didacticiel, à savoir `eks-sample-app`, à moins que vous ne changiez le nom. Si vous avez créé un cluster à l'aide de l'un des guides disponibles dans Démarrer, vous devrez créer un nouveau profil ou ajouter l'espace de noms à votre profil existant, car le profil créé dans les guides de démarrage ne spécifie pas l'espace de noms utilisé dans ce tutoriel. Votre VPC doit également disposer d'au moins un sous-réseau privé.

Bien que de nombreuses variables soient modifiables dans les étapes suivantes, nous vous recommandons de ne modifier les valeurs des variables qu'aux endroits spécifiés. Une fois que vous aurez mieux compris les pods, les déploiements et les services Kubernetes, vous pourrez essayer de modifier d'autres valeurs.

Création d'un espace de noms

Un espace de noms vous permet de regrouper des ressources dans Kubernetes. Pour plus d'informations, consultez Espaces de noms dans la documentation Kubernetes. Si vous prévoyez de déployer votre application exemple sur Simplifier la gestion des ressources de calcul avec AWS Fargate, assurez-vous que la valeur du `namespace` dans votre Définir les pods qui utilisent AWS Fargate lors de leur lancement est `eks-sample-app`.

```
kubectl create namespace eks-sample-app
```

Créer un déploiement Kubernetes

Cet exemple de déploiement extrait une image de conteneur d'un référentiel public et en déploie trois réplicas (pods individuels) sur votre cluster. Pour en savoir plus, consultez Déploiements dans la documentation Kubernetes.

1. Enregistrez le contenu suivant dans un fichier nommé `eks-sample-deployment.yaml`. Les conteneurs de l'application exemple n'utilisent pas de stockage réseau, mais vous pouvez avoir des applications qui en ont besoin. Pour de plus amples informations, consultez Stockage de données d'application.
   - Le `kubernetes.io/os: windows` `nodeSelector` signifie que si vous possédez des nœuds Windows et Linux (par exemple) dans votre cluster, l'image ne sera déployée que sur des nœuds Windows. Pour plus d'informations, consultez Étiquettes, annotations et rejets connus dans la documentation Kubernetes.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: eks-sample-windows-deployment
  namespace: eks-sample-app
  labels:
    app: eks-sample-windows-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: eks-sample-windows-app
  template:
    metadata:
      labels:
        app: eks-sample-windows-app
    spec:
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: kubernetes.io/arch
                operator: In
                values:
                - amd64
      containers:
      - name: windows-server-iis
        image: mcr.microsoft.com/windows/servercore:ltsc2019
        ports:
        - name: http
          containerPort: 80
        imagePullPolicy: IfNotPresent
        command:
        - powershell.exe
        - -command
        - "Add-WindowsFeature Web-Server; Invoke-WebRequest -UseBasicParsing
 -Uri 'https://dotnetbinaries.blob.core.windows.net/servicemonitor/2.0.1.6/
ServiceMonitor.exe' -OutFile 'C:\\ServiceMonitor.exe'; echo '<html><body><br/
><br/><marquee><H1>Hello EKS!!!<H1><marquee></body><html>' > C:\\inetpub\\wwwroot\
\default.html; C:\\ServiceMonitor.exe 'w3svc'; "
      nodeSelector:
```

```
        kubernetes.io/os: windows
```

2. Appliquer le manifeste de déploiement à votre cluster.

```
kubectl apply -f eks-sample-deployment.yaml
```

Création d'un service

Un service vous permet d'accéder à toutes les réplicas via une seule adresse IP ou un seul nom. Pour plus d'informations, consultez Service dans la documentation Kubernetes. Bien que cela ne soit pas implémenté dans l'application exemple, si vous avez des applications qui doivent interagir avec d'autres services AWS, nous vous recommandons de créer des comptes de service Kubernetes pour vos pods et de les associer à des comptes AWS IAM. En spécifiant des comptes de service, vos pods ne disposent que des autorisations minimales que vous spécifiez pour qu'ils puissent interagir avec d'autres services. Pour de plus amples informations, consultez the section called "Informations d'identification auprès de l'IRSA".

1. Enregistrez le contenu suivant dans un fichier nommé `eks-sample-service.yaml`. Kubernetes attribue au service sa propre adresse IP qui n'est accessible que depuis l'intérieur du cluster. Pour accéder au service depuis l'extérieur de votre cluster, déployez l'AWS Load Balancer Controller pour équilibrer la charge du trafic d'application ou de réseau vers le service.

```
apiVersion: v1
kind: Service
metadata:
  name: eks-sample-windows-service
  namespace: eks-sample-app
  labels:
    app: eks-sample-windows-app
spec:
  selector:
    app: eks-sample-windows-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

2. Appliquer le manifeste du service à votre cluster.

```
kubectl apply -f eks-sample-service.yaml
```

Vérifier les ressources créées

1. Affichez toutes les ressources présentes dans l'espace de noms `eks-sample-app`.

```
kubectl get all -n eks-sample-app
```

L'exemple qui suit illustre un résultat.

```
NAME                                                 READY   STATUS    RESTARTS   AGE
pod/eks-sample-windows-deployment-65b7669776-m6qxz   1/1     Running   0          27m
pod/eks-sample-windows-deployment-65b7669776-mmxvd   1/1     Running   0          27m
pod/eks-sample-windows-deployment-65b7669776-qzn22   1/1     Running   0          27m

NAME                                 TYPE         CLUSTER-IP      EXTERNAL-IP   PORT(S)
   AGE
service/eks-sample-windows-service   ClusterIP    10.100.74.8     <none>        80/
TCP    32m

NAME                                            READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/eks-sample-windows-deployment 3/3     3            3           27m

NAME                                                        DESIRED   CURRENT   READY
 AGE
replicaset.apps/eks-sample-windows-deployment-776d8f8fd8    3         3         3
   27m
```

Dans la sortie, vous voyez le service et le déploiement spécifiés dans l'exemple de manifestes déployés dans les étapes précédentes. Vous voyez aussi trois pods. Ceci est dû au fait que 3 `replicas` ont été spécifiés dans l'exemple de manifeste. Pour plus d'informations sur les pods, consultez pods dans la documentation Kubernetes. Kubernetes crée automatiquement la ressource `replicaset`, même si elle n'est pas spécifiée dans les manifestes d'exemple. Pour plus d'informations sur `ReplicaSets`, veuillez consulter la section ReplicaSet dans la documentation Kubernetes.

> **Note**
>
> Kubernetes conserve le nombre de réplicas spécifié dans le manifeste. S'il s'agissait d'un déploiement en production et que vous vouliez que Kubernetes fasse mettre à l'échelle horizontalement le nombre de réplicas ou verticalement les ressources de calcul pour les pods, utilisez les options Évoluer les déploiements de pods avec Horizontal Pod Autoscaler et Ajuster les ressources des pods avec Vertical Pod Autoscaler pour ce faire.

2. Affichez les détails du service déployé.

```
kubectl -n eks-sample-app describe service eks-sample-windows-service
```

L'exemple qui suit illustre un résultat.

```
Name:              eks-sample-windows-service
Namespace:         eks-sample-app
Labels:            app=eks-sample-windows-app
Annotations:       <none>
Selector:          app=eks-sample-windows-app
Type:              ClusterIP
IP Families:       <none>
IP:                10.100.74.8
IPs:               10.100.74.8
Port:              <unset>  80/TCP
TargetPort:        80/TCP
Endpoints:         192.168.24.212:80,192.168.50.185:80,192.168.63.93:80
Session Affinity:  None
Events:            <none>
```

Dans la sortie précédente, la valeur de `IP:` est une adresse IP unique accessible depuis n'importe quel nœud ou pod du cluster, mais inaccessible depuis l'extérieur du cluster. Les valeurs de `Endpoints` sont des adresses IP attribuées depuis votre VPC aux pods qui font partie du service.

3. Visualisez les détails de l'un des pods listés en sortie lorsque vous avez visualisé l'espace de noms dans une étape précédente. Remplacez *776d8f8fd8-78w66* par la valeur renvoyée pour l'un de vos pods.

```
kubectl -n eks-sample-app describe pod eks-sample-windows-deployment-65b7669776-m6qxz
```

Exemple de sortie abrégée

```
Name:         eks-sample-windows-deployment-65b7669776-m6qxz
Namespace:    eks-sample-app
Priority:     0
Node:         ip-192-168-45-132.us-west-2.compute.internal/192.168.45.132
[...]
IP:           192.168.63.93
IPs:
  IP:           192.168.63.93
Controlled By:  ReplicaSet/eks-sample-windows-deployment-65b7669776
[...]
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
[...]
Events:
  Type    Reason     Age    From
 Message
  ----    ------     ----   ----
 -------
  Normal  Scheduled  3m20s  default-scheduler
 Successfully assigned eks-sample-app/eks-sample-windows-deployment-65b7669776-m6qxz
 to ip-192-168-45-132.us-west-2.compute.internal
[...]
```

Dans la sortie précédente, la valeur de `IP:` est une adresse IP unique attribuée au pod à partir du bloc CIDR attribué au sous-réseau dans lequel se trouve le nœud. Si vous préférez attribuer aux pods des adresses IP à partir de différents blocs d'adresse CIDR, vous pouvez modifier le comportement par défaut. Pour de plus amples informations, consultez the section called “Mise en réseau personnalisée”. Vous pouvez également voir que le planificateur Kubernetes a planifié le pod sur le Node avec l'adresse IP *`192.168.45.132`*.

> **Tip**
>
> Au lieu d'utiliser la ligne de commande, vous pouvez afficher de nombreux détails sur les pods, les services, les déploiements et d'autres ressources Kubernetes dans la AWS

Management Console. Pour de plus amples informations, consultez the section called "Accéder aux ressources du cluster".

Exécuter un shell sur un pod

1. Exécutez un shell sur le pod que vous avez décrit à l'étape précédente, en remplaçant *65b7669776-m6qxz* par l'ID de l'un de vos pods.

```
kubectl exec -it eks-sample-windows-deployment-65b7669776-m6qxz -n eks-sample-app --
 powershell.exe
```

2. À partir du shell du pod, affichez la sortie du serveur web qui a été installé avec votre déploiement à l'étape précédente. Vous devez uniquement spécifier le nom du service. Elle est résolue en adresse IP du service par CoreDNS, qui est déployé par défaut avec un cluster Amazon EKS.

```
Invoke-WebRequest -uri eks-sample-windows-service/default.html -UseBasicParsing
```

   L'exemple qui suit illustre un résultat.

```
StatusCode        : 200
StatusDescription : OK
Content           : < h t m l > < b o d y > < b r / > < b r / > < m a r q u e e > < H
 1 > H e l l o
                      E K S ! ! ! < H 1 > < m a r q u e e > < / b o d y > < h t m l >
```

3. À partir du shell du pod, affichez le serveur DNS pour le pod.

```
Get-NetIPConfiguration
```

   Sortie abrégée

```
InterfaceAlias       : vEthernet
[...]
IPv4Address          : 192.168.63.14
[...]
DNSServer            : 10.100.0.10
```

Dans la sortie précédente, `10.100.0.10` est automatiquement affecté en tant que serveur DNS pour tous les pods déployés dans le cluster.

4. Déconnectez-vous du pod en tapant `exit`.
5. Une fois que vous avez terminé avec l'application exemple, vous pouvez supprimer l'espace de noms, le service et le déploiement exemple à l'aide de la commande suivante.

```
kubectl delete namespace eks-sample-app
```

Étapes suivantes

Après avoir déployé l'application exemple, vous pourriez vouloir essayer certains des exercices suivants :

- Router l'application et le trafic HTTP avec des Application Load Balancer
- Router le trafic TCP et UDP avec des Network Load Balancer

Ajuster les ressources des pods avec Vertical Pod Autoscaler

Kubernetes Vertical Pod Autoscaler ajuste automatiquement les réservations de CPU et de mémoire pour vos pods afin de vous aider à « dimensionner correctement » vos applications. Cet ajustement peut améliorer l'utilisation des ressources du cluster et libérer du CPU et de la mémoire pour d'autres pods. Cette rubrique vous aide à déployer le VPA (Vertical Pod Autoscaler) sur votre cluster et à vérifier qu'il fonctionne.

- Vous disposez d'un cluster Amazon EKS existant. Si ce n'est pas le cas, consultez Démarrer.
- Le serveur de métriques Kubernetes est installé. Pour de plus amples informations, consultez the section called "Serveur de métriques".
- Vous utilisez un client `kubectl` qui est configuré pour communiquer avec votre cluster Amazon EKS.
- OpenSSL `1.1.1` ou version ultérieure installé sur votre appareil.

Déploiement du VPA (Vertical Pod Autoscaler)

Dans cette section, vous déployez le Vertical Pod Autoscaler sur votre cluster.

1. Ouvrez une fenêtre de terminal et accédez à un répertoire dans lequel vous souhaitez télécharger le code source du VPA (Vertical Pod Autoscaler).
2. Clonez le référentiel GitHub kubernetes/autoscaler.

```
git clone https://github.com/kubernetes/autoscaler.git
```

3. Passez au répertoire `vertical-pod-autoscaler`.

```
cd autoscaler/vertical-pod-autoscaler/
```

4. (Facultatif) Si vous avez déjà déployé une autre version du VPA (Vertical Pod Autoscaler), supprimez-la avec la commande suivante.

```
./hack/vpa-down.sh
```

5. Si vos nœuds n'ont pas accès à Internet pour se connecter au registre de conteneurs `registry.k8s.io`, vous devez extraire les images suivantes et les envoyer vers votre propre référentiel privé. Pour plus d'informations sur la manière d'extraire les images et de les transférer vers votre propre référentiel privé, consultez the section called "Copier une image dans un référentiel".

```
registry.k8s.io/autoscaling/vpa-admission-controller:0.10.0
registry.k8s.io/autoscaling/vpa-recommender:0.10.0
registry.k8s.io/autoscaling/vpa-updater:0.10.0
```

   Si vous transférez les images vers un référentiel Amazon ECR privé, remplacez `registry.k8s.io` dans les manifestes par votre registre. Remplacez *`111122223333`* par votre ID de compte. Remplacez *`region-code`* par la région AWS dans laquelle votre cluster se situe. Les commandes suivantes supposent que vous avez nommé votre référentiel de la même manière que le référentiel du manifeste. Si vous avez donné un nom différent à votre référentiel, vous devrez également le modifier.

```
sed -i.bak -e 's/registry.k8s.io/111122223333.dkr.ecr.region-code.amazonaws.com/' ./
deploy/admission-controller-deployment.yaml
sed -i.bak -e 's/registry.k8s.io/111122223333.dkr.ecr.region-code.amazonaws.com/' ./
deploy/recommender-deployment.yaml
sed -i.bak -e 's/registry.k8s.io/111122223333.dkr.ecr.region-code.amazonaws.com/' ./
deploy/updater-deployment.yaml
```

6. Déployez le VPA (Vertical Pod Autoscaler) sur votre cluster avec la commande suivante.

```
./hack/vpa-up.sh
```

7. Vérifiez que les pods Vertical Pod Autoscaler ont été créés avec succès.

```
kubectl get pods -n kube-system
```

L'exemple qui suit illustre un résultat.

```
NAME                                        READY   STATUS    RESTARTS   AGE
[...]
metrics-server-8459fc497-kfj8w              1/1     Running   0          83m
vpa-admission-controller-68c748777d-ppspd   1/1     Running   0          7s
vpa-recommender-6fc8c67d85-gljpl            1/1     Running   0          8s
vpa-updater-786b96955c-bgp9d                1/1     Running   0          8s
```

Test de l'installation de votre VPA (Vertical Pod Autoscaler)

Dans cette section, vous déployez un exemple d'application pour vérifier que le Vertical Pod Autoscaler fonctionne.

1. Déployez l'exemple `hamster.yaml` Vertical Pod Autoscaler avec la commande suivante.

```
kubectl apply -f examples/hamster.yaml
```

2. Récupérez les pods à partir de l'application exemple `hamster`.

```
kubectl get pods -l app=hamster
```

L'exemple qui suit illustre un résultat.

```
hamster-c7d89d6db-rglf5   1/1     Running   0          48s
hamster-c7d89d6db-znvz5   1/1     Running   0          48s
```

3. Décrivez l'un des pods pour afficher sa réservation de `cpu` et de `memory`. Remplacez *`c7d89d6db-rglf5`* par l'un des ID renvoyés dans votre sortie de l'étape précédente.

```
kubectl describe pod hamster-c7d89d6db-rglf5
```

L'exemple qui suit illustre un résultat.

```
[...]
Containers:
  hamster:
    Container ID:  docker://
e76c2413fc720ac395c33b64588c82094fc8e5d590e373d5f818f3978f577e24
    Image:         registry.k8s.io/ubuntu-slim:0.1
    Image ID:      docker-pullable://registry.k8s.io/ubuntu-
slim@sha256:b6f8c3885f5880a4f1a7cf717c07242eb4858fdd5a84b5ffe35b1cf680ea17b1
    Port:          <none>
    Host Port:     <none>
    Command:
      /bin/sh
    Args:
      -c
      while true; do timeout 0.5s yes >/dev/null; sleep 0.5s; done
    State:          Running
      Started:      Fri, 27 Sep 2019 10:35:16 -0700
    Ready:          True
    Restart Count:  0
    Requests:
      cpu:        100m
      memory:     50Mi
[...]
```

Vous pouvez constater que le pod d'origine réserve 100 millicpu de CPU et 50 mégioctets de mémoire. Pour cette application exemple, 100 millicpu est inférieur à ce dont le pod a besoin pour fonctionner, il est donc limité en termes de CPU. Il réserve également beaucoup moins de mémoire qu'il n'en a besoin. Le déploiement Vertical Pod Autoscaler `vpa-recommender` analyse les pods hamster pour vérifier si les exigences en matière de CPU et de mémoire sont appropriées. Si des ajustements sont nécessaires, le `vpa-updater` relance les pods avec les valeurs mises à jour.

4. Veuillez attendre que le `vpa-updater` lance un nouveau pod hamster. Cela devrait prendre une minute ou deux. Vous pouvez surveiller les pods à l'aide de la commande suivante.

> **Note**
>
> Si vous n'êtes pas certain qu'un nouveau pod ait été lancé, comparez les noms des pods avec votre liste précédente. Lorsque le nouveau pod sera lancé, vous verrez apparaître un nouveau nom de pod.

```
kubectl get --watch Pods -l app=hamster
```

5. Lorsqu'un nouveau pod hamster est démarré, décrivez-le et consultez les réservations de CPU et de mémoire mises à jour.

```
kubectl describe pod hamster-c7d89d6db-jxgfv
```

L'exemple qui suit illustre un résultat.

```
[...]
Containers:
  hamster:
    Container ID:
 docker://2c3e7b6fb7ce0d8c86444334df654af6fb3fc88aad4c5d710eac3b1e7c58f7db
    Image:         registry.k8s.io/ubuntu-slim:0.1
    Image ID:      docker-pullable://registry.k8s.io/ubuntu-
slim@sha256:b6f8c3885f5880a4f1a7cf717c07242eb4858fdd5a84b5ffe35b1cf680ea17b1
    Port:          <none>
    Host Port:     <none>
    Command:
      /bin/sh
    Args:
      -c
      while true; do timeout 0.5s yes >/dev/null; sleep 0.5s; done
    State:          Running
      Started:      Fri, 27 Sep 2019 10:37:08 -0700
    Ready:          True
    Restart Count:  0
    Requests:
      cpu:        587m
      memory:     262144k
[...]
```

Dans la sortie précédente, vous pouvez voir que la réservation de l'`cpu` est passée à 587 millicpu, ce qui représente plus de cinq fois la valeur d'origine. La `memory` a augmenté jusqu'à 262 144 kilo-octets, soit environ 250 mébioctets ou cinq fois la valeur d'origine. Ce pod manquait de ressources, et le Vertical Pod Autoscaler a corrigé l'estimation avec une valeur beaucoup plus appropriée.

6. Décrivez la ressource `hamster-vpa` pour afficher la nouvelle recommandation.

```
kubectl describe vpa/hamster-vpa
```

L'exemple qui suit illustre un résultat.

```
Name:         hamster-vpa
Namespace:    default
Labels:       <none>
Annotations:  kubectl.kubernetes.io/last-applied-configuration:
                {"apiVersion":"autoscaling.k8s.io/
v1beta2","kind":"VerticalPodAutoscaler","metadata":{"annotations":{},"name":"hamster-
vpa","namespace":"d...
API Version:  autoscaling.k8s.io/v1beta2
Kind:         VerticalPodAutoscaler
Metadata:
  Creation Timestamp:  2019-09-27T18:22:51Z
  Generation:          23
  Resource Version:    14411
  Self Link:           /apis/autoscaling.k8s.io/v1beta2/namespaces/default/
verticalpodautoscalers/hamster-vpa
  UID:                 d0d85fb9-e153-11e9-ae53-0205785d75b0
Spec:
  Target Ref:
    API Version:  apps/v1
    Kind:         Deployment
    Name:         hamster
Status:
  Conditions:
    Last Transition Time:  2019-09-27T18:23:28Z
    Status:                True
    Type:                  RecommendationProvided
  Recommendation:
    Container Recommendations:
      Container Name:  hamster
      Lower Bound:
        Cpu:     550m
```

```
        Memory:  262144k
      Target:
        Cpu:     587m
        Memory:  262144k
      Uncapped Target:
        Cpu:     587m
        Memory:  262144k
      Upper Bound:
        Cpu:     21147m
        Memory:  387863636
Events:          <none>
```

7. Lorsque vous avez terminé de tester votre exemple d'application, supprimez-le avec la commande suivante.

```
kubectl delete -f examples/hamster.yaml
```

Déployez des pods à l'échelle avec Horizontal Pod Autoscaler

Kubernetes Horizontal Pod Autoscaler met automatiquement à l'échelle le nombre de pods dans un déploiement, un contrôleur de réplication ou un ensemble de réplicas en fonction de l'utilisation du CPU de cette ressource. Cela peut permettre à vos applications de monter en puissance pour répondre à une demande accrue ou de baisser en puissance lorsque les ressources ne sont pas nécessaires, libérant ainsi vos nœuds pour d'autres applications. Lorsque vous définissez un pourcentage cible d'utilisation du processeur, Horizontal Pod Autoscaler met à l'échelle votre application pour essayer d'atteindre cet objectif.

Horizontal Pod Autoscaler est une ressource API standard dans Kubernetes qui nécessite simplement qu'une source de métriques (telle que le serveur de métriques Kubernetes) soit installée sur votre cluster Amazon EKS pour fonctionner. Vous n'avez pas besoin de déployer ou d'installer Horizontal Pod Autoscaler sur votre cluster pour mettre vos applications à l'échelle. Pour plus d'informations, consultez Horizontal Pod Autoscaler dans la documentation Kubernetes.

Utilisez cette rubrique pour préparer Horizontal Pod Autoscaler pour votre cluster Amazon EKS et pour vérifier qu'il fonctionne avec un exemple d'application.

> **Note**
>
> Cette rubrique est basée sur la démonstration Horizontal Pod Autoscaler de la documentation Kubernetes.

- Vous disposez d'un cluster Amazon EKS existant. Si ce n'est pas le cas, consultez Démarrer.
- Le serveur de métriques Kubernetes est installé. Pour de plus amples informations, consultez the section called "Serveur de métriques".
- Vous utilisez un client `kubectl` qui est configuré pour communiquer avec votre cluster Amazon EKS.

Exécution d'une application de test du Horizontal Pod Autoscaler

Dans cette section, vous déployez un exemple d'application pour vérifier que Horizontal Pod Autoscale) fonctionne.

> **Note**
>
> Cet exemple est basé sur la démonstration Horizontal Pod Autoscaler de la documentation Kubernetes.

1. Déployez une application simple de serveur web Apache avec la commande suivante.

   ```
   kubectl apply -f https://k8s.io/examples/application/php-apache.yaml
   ```

   Ce pod de serveur web Apache est limité à 500 millicpu par processeur et fonctionne sur le port 80.
2. Créez une ressource Horizontal Pod Autoscaler pour le déploiement `php-apache`.

   ```
   kubectl autoscale deployment php-apache --cpu-percent=50 --min=1 --max=10
   ```

   Cette commande crée un autoscaler qui vise une utilisation de 50 % du CPU pour le déploiement, avec un minimum d'un pod et un maximum de dix pods. Lorsque la charge moyenne du processeur est inférieure à 50 %, l'autoscaler tente de réduire le nombre de pods dans le

déploiement, jusqu'à un minimum d'un. Lorsque la charge est supérieure à 50 %, l'autoscaler essaie d'augmenter le nombre de pods dans le déploiement, jusqu'à un maximum de dix. Pour plus d'informations, consultez la section « Comment fonctionne HorizontalPodAutoscaler ? » dans la documentation Kubernetes.

3. Décrivez l'autoscaler avec la commande suivante pour en afficher les détails.

```
kubectl get hpa
```

L'exemple qui suit illustre un résultat.

```
NAME         REFERENCE                TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
php-apache   Deployment/php-apache   0%/50%    1         10        1          51s
```

Comme vous pouvez le voir, la charge CPU actuelle est `0%`, car le serveur n'est pas encore sollicité. Le nombre de pods est déjà à sa limite inférieure (un), il ne peut donc pas être augmenté.

4. Créez une charge pour le serveur web en exécutant un conteneur .

```
kubectl run -i \
    --tty load-generator \
    --rm --image=busybox \
    --restart=Never \
    -- /bin/sh -c "while sleep 0.01; do wget -q -O- http://php-apache; done"
```

5. Pour observer l'augmentation du déploiement, exécutez régulièrement la commande suivante dans un terminal distinct de celui dans lequel vous avez exécuté l'étape précédente.

```
kubectl get hpa php-apache
```

L'exemple qui suit illustre un résultat.

```
NAME         REFERENCE                TARGETS    MINPODS   MAXPODS   REPLICAS   AGE
php-apache   Deployment/php-apache   250%/50%   1         10        5          4m44s
```

L'augmentation du nombre de réplicas peut prendre plus d'une minute. Tant que le pourcentage de CPU réel est supérieur au pourcentage cible, le nombre de réplicas augmente jusqu'à 10. Dans ce cas, c'est `250%`, donc le nombre de `REPLICAS` continue d'augmenter.

> **Note**
>
> L'atteinte du nombre de réplicas maximum peut prendre un certain temps. Si, par exemple, seules six réplicas sont nécessaires pour que la charge CPU reste inférieure ou égale à 50 %, alors la charge ne dépassera pas six réplicas.

6. Arrêtez la charge. Dans la fenêtre de terminal dans laquelle vous générez la charge, arrêtez la charge en maintenant les touches `Ctrl+C` enfoncées. Vous pouvez voir les réplicas revenir à 1 en exécutant à nouveau la commande suivante dans le terminal dans lequel vous observez la mise à l'échelle horizontale.

```
kubectl get hpa
```

   L'exemple qui suit illustre un résultat.

```
NAME         REFERENCE                TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
php-apache   Deployment/php-apache    0%/50%    1         10        1          25m
```

> **Note**
>
> Le délai par défaut pour la réduction d'échelle est de cinq minutes. Par conséquent, un certain temps s'écoule avant de voir le nombre de réplicas atteindre à nouveau 1, même si le pourcentage de CPU actuel est de 0 %. Le délai est modifiable. Pour plus d'informations, consultez Horizontal Pod Autoscaler dans la documentation Kubernetes.

7. Lorsque vous avez terminé de tester votre exemple d'application, supprimez les ressources `php-apache`.

```
kubectl delete deployment.apps/php-apache service/php-apache
 horizontalpodautoscaler.autoscaling/php-apache
```

Acheminer le trafic TCP et UDP avec des Network Load Balancers

> **Note**
>
> Nouveau : le mode automatique Amazon EKS automatise les tâches courantes liées à l'équilibrage de charge. Pour en savoir plus, consultez :
>
> - the section called "Déploiement de l'équilibreur de charge"
> - the section called "Créer un service"

Le trafic réseau est équilibré en charge au niveau L4 du modèle OSI. Pour équilibrer la charge du trafic des applications surL7, vous déployez un Kubernetes`ingress`, qui fournit un Application AWS Load Balancer. Pour de plus amples informations, veuillez consulter the section called "Répartition de la charge des applications". Pour en savoir plus sur les différences entre les deux types d'équilibrage de charge, consultez les fonctionnalités d'Elastic Load Balancing sur le AWS site Web.

Lorsque vous créez un Kubernetes `Service` de type Kubernetes`LoadBalancer`, le contrôleur d'équilibrage de charge du fournisseur de AWS cloud crée des équilibreurs de charge AWS classiques par défaut, mais peut également créer des équilibreurs de charge réseau. AWS Ce contrôleur ne recevra que des corrections de bugs critiques à l'avenir. Pour plus d'informations sur l'utilisation de l'équilibreur de charge du fournisseur de AWS cloud, consultez le contrôleur d'équilibrage de charge du fournisseur de AWS cloud dans la documentation de Kubernetes. Son utilisation n'est pas abordée dans cette rubrique.

Nous vous recommandons d'utiliser la version `2.7.2` ou une version ultérieure du AWS Load Balancer Controller au lieu du contrôleur d'équilibrage de charge du fournisseur de AWS cloud. Le AWS Load Balancer Controller crée des équilibreurs de charge AWS réseau, mais pas des équilibreurs de charge AWS classiques. Le reste de cette rubrique traite de l'utilisation du AWS Load Balancer Controller.

Un AWS Network Load Balancer peut équilibrer la charge du trafic réseau vers les pods déployés sur Amazon EC2 IP et les cibles d'instance, vers les cibles IP Fargate ou vers AWS les nœuds hybrides Amazon EKS en tant que cibles IP. Pour plus d'informations, consultez AWS Load Balancer Controller activé. GitHub

Conditions préalables

Avant de pouvoir équilibrer la charge du trafic réseau à l'aide du AWS Load Balancer Controller, vous devez satisfaire aux exigences suivantes.

- Disposer d'un cluster. Si vous n'avez pas de cluster, consultez Démarrer. Si vous devez mettre à jour la version d'un cluster existant, consultez the section called "Mises à jour de la version Kubernetes".
- Déployez le AWS Load Balancer Controller sur votre cluster. Pour de plus amples informations, veuillez consulter the section called " AWS Contrôleur Load Balancer". Nous recommandons la version `2.7.2` ou ultérieure.
- Au moins un sous-réseau. Si plusieurs sous-réseaux identifiés sont trouvés dans une zone de disponibilité, le contrôleur choisit le sous-réseau dont l'ID vient en premier dans l'ordre lexicographique. Le sous-réseau doit disposer au moins huit adresses IP disponibles.
- Si vous utilisez la version du AWS Load Balancer Controller `2.1.1` ou une version antérieure, les sous-réseaux doivent être balisés comme suit. Si vous utilisez la version `2.1.2` ou une version ultérieure, cette identification est facultative. Vous souhaiterez peut-être baliser un sous-réseau si plusieurs clusters s'exécutent dans le même VPC, ou si AWS plusieurs services partagent des sous-réseaux dans un VPC, et si vous souhaitez mieux contrôler l'endroit où les équilibreurs de charge sont fournis pour chaque cluster. Si vous spécifiez explicitement un sous-réseau IDs sous forme d'annotation sur un objet de service, Kubernetes et le Load Balancer Controller utilisent directement ces sous-réseaux pour créer l'équilibreur de AWS charge. Vous n'avez pas besoin de baliser les sous-réseaux si vous choisissez d'utiliser cette méthode pour provisionner les équilibreurs de charge et vous pouvez ignorer les exigences suivantes en matière de balisage des sous-réseaux privés et publics. Remplacez *`my-cluster`* par le nom de votre cluster.
  - Clé : `kubernetes.io/cluster/<my-cluster>`
  - Valeur : `shared` ou `owned`
- Vos sous-réseaux publics et privés doivent répondre aux exigences suivantes, sauf si vous le spécifiez explicitement sous IDs forme d'annotation sur un service ou un objet d'entrée. Si vous configurez des équilibreurs de charge en spécifiant explicitement un sous-réseau IDs sous forme d'annotation sur un service ou un objet d'entrée, Kubernetes et le Load AWS Balancer Controller utilisent ces sous-réseaux directement pour créer l'équilibreur de charge et les balises suivantes ne sont pas requises.
  - Sous-réseaux privés : doivent être étiquetés dans le format suivant. Cela permet à Kubernetes et au Load Balancer Controller de savoir que les sous-réseaux peuvent être utilisés pour les

équilibreurs de AWS charge internes. Si vous utilisez `eksctl` un AWS AWS CloudFormation modèle Amazon EKS pour créer votre VPC après le 26 mars 2020, les sous-réseaux sont balisés de manière appropriée lors de leur création. Pour plus d'informations sur les modèles de VPC Amazon EKS AWS AWS CloudFormation , consultez the section called “Création d’un VPC”.

- Clé : `kubernetes.io/role/internal-elb`
- Valeur : `1`

- Sous-réseau publics : doivent être étiquetés dans le format suivant. Kubernetes sait ainsi qu'il ne faut utiliser que ces sous-réseaux pour les équilibreurs de charge externes au lieu de choisir un sous-réseau public dans chaque zone de disponibilité (en fonction de l'ordre lexicographique du sous-réseau). IDs Si vous utilisez `eksctl` un AWS CloudFormation modèle Amazon EKS pour créer votre VPC après le 26 mars 2020, les sous-réseaux sont balisés de manière appropriée lors de leur création. Pour plus d'informations sur les modèles de AWS CloudFormation VPC Amazon EKS, consultez. the section called “Création d’un VPC”
  - Clé : `kubernetes.io/role/elb`
  - Valeur : `1`

Si les balises de rôle du sous-réseau ne sont pas explicitement ajoutées, le contrôleur de service Kubernetes examine la table de routage des sous-réseaux VPC de votre cluster afin de déterminer si le sous-réseau est privé ou public. Nous vous recommandons de ne pas vous fier à ce comportement et d’ajouter explicitement les balises de rôle privées ou publiques. Le AWS Load Balancer Controller n'examine pas les tables de routage et nécessite la présence des balises privées et publiques pour une détection automatique réussie.

Considérations

- La configuration de votre équilibreur de charge est contrôlée par des annotations qui sont ajoutées au manifeste de votre service. Les annotations de service sont différentes lorsque vous utilisez le AWS Load Balancer Controller et lorsque vous utilisez AWS le contrôleur d'équilibrage de charge du fournisseur de cloud. Assurez-vous de consulter les annotations relatives au AWS Load Balancer Controller avant de déployer des services.
- Lorsque vous utilisez le plug-in Amazon VPC CNI pour Kubernetes, le Load AWS Balancer Controller peut équilibrer la charge entre EC2 Amazon IP ou les cibles d'instance et les cibles IP Fargate. Lorsque vous utilisez des modules complémentaires CNI compatibles alternatifs, le contrôleur ne peut équilibrer la charge que vers des cibles d’instance, sauf si vous équilibrez

la charge vers des nœuds hybrides Amazon EKS. Pour les nœuds hybrides, le contrôleur peut équilibrer la charge des cibles IP. Pour plus d'informations sur les types de cibles de Network Load Balancer, veuillez consulter la section Type de cible dans le guide de l'utilisateur de Network Load Balancer

- Si vous voulez ajouter des balises à l'équilibreur de charge lors de sa création ou après celle-ci, ajoutez l'annotation suivante dans votre spécification de service. Pour plus d'informations, consultez la section AWS Resource Tags dans la documentation du AWS Load Balancer Controller.

```
service.beta.kubernetes.io/aws-load-balancer-additional-resource-tags
```

- Vous pouvez affecter des adresses IP Elastic au Network Load Balancer en ajoutant l'annotation suivante. Remplacez les valeurs d'exemple par celles `Allocation IDs` de vos adresses IP Elastic. Le nombre de `Allocation IDs` doit correspondre au nombre de sous-réseaux utilisés pour l'équilibreur de charge. Pour plus d'informations, consultez la documentation du contrôleur AWS Load Balancer Controller.

```
service.beta.kubernetes.io/aws-load-balancer-eip-allocations: eipalloc-
xxxxxxxxxxxxxxxx,eipalloc-yyyyyyyyyyyyyyyyy
```

- Amazon EKS ajoute une règle entrante au groupe de sécurité du nœud pour le trafic client et une règle pour chaque sous-réseau de l'équilibreur de charge dans le VPC pour la surveillance de l'état de chaque Network Load Balancer que vous créez. Le déploiement d'un service de type `LoadBalancer` peut échouer si Amazon EKS tente de créer des règles qui dépassent le quota du nombre maximal de règles autorisées pour un groupe de sécurité. Pour plus d'informations, consultez Groupes de sécurité dans Quotas de VPC Amazon dans le guide de l'utilisateur d'Amazon VPC. Tenez compte des options suivantes pour réduire les possibilités de dépassement du nombre maximum de règles pour un groupe de sécurité :
  - Demandez une augmentation dans vos règles par quota de groupe de sécurité. Pour plus d'informations, consultez Demande d'une augmentation de quota dans le Guide de l'utilisateur de Service Quotas.
  - Utilisez des cibles IP plutôt que des cibles d'instance. Avec les cibles IP, vous pouvez partager des règles pour les mêmes ports cibles. Vous pouvez spécifier manuellement les sous-réseaux des équilibreurs de charge avec une annotation. Pour plus d'informations, consultez la section Annotations sur GitHub.
  - Utilisez une entrée (ingress), au lieu d'un service de type `LoadBalancer`, pour envoyer le trafic vers votre service. L' AWS Application Load Balancer nécessite moins de règles que les Network

Load Balancer. Vous pouvez partager un ALB sur plusieurs entrées. Pour de plus amples informations, veuillez consulter the section called "Répartition de la charge des applications". Vous ne pouvez pas partager un Network Load Balancer entre plusieurs services.

- Déployez vos clusters dans plusieurs comptes.

- Si vos pods s'exécutent sous Windows dans un cluster Amazon EKS, un seul service avec un équilibreur de charge peut prendre en charge jusqu'à 1 024 pods dorsaux. Chaque pod a sa propre adresse IP.
- Nous recommandons de créer uniquement de nouveaux Network Load Balancers à l'aide du AWS Load Balancer Controller. Toute tentative de remplacement des équilibreurs de charge réseau existants créés à l'aide du contrôleur d'équilibrage de charge du fournisseur de AWS cloud peut entraîner la création de plusieurs équilibreurs de charge réseau susceptibles de provoquer des interruptions de service des applications.

Créer un équilibreur de charge de réseau

Vous pouvez créer un équilibreur de charge de réseau avec des cibles IP ou d'instance.

Créer un dispositif d'équilibrage de charge de réseau — Cibles IP

- Vous pouvez utiliser des cibles IP avec des pods déployés sur des EC2 nœuds Amazon, Fargate ou Amazon EKS Hybrid Nodes. Votre service Kubernetes doit être créé en tant que type `LoadBalancer`. Pour plus d'informations, consultez la section Type LoadBalancer dans la documentation de Kubernetes.

  Pour créer un équilibreur de charge qui utilise des cibles IP, ajoutez l'annotation suivante à un manifeste du service et déployez votre service. La `external` valeur de `aws-load-balancer-type` est ce qui pousse le AWS Load Balancer Controller, plutôt que le contrôleur d'équilibrage de charge du fournisseur de AWS cloud, à créer le Network Load Balancer. Vous pouvez visualiser un exemple de manifeste de service avec les annotations.

  ```
  service.beta.kubernetes.io/aws-load-balancer-type: "external"
  service.beta.kubernetes.io/aws-load-balancer-nlb-target-type: "ip"
  ```

  > **Note**
  >
  > Si vous effectuez l'équilibrage de charge vers des pods `IPv6`, ajoutez l'annotation suivante. Vous ne pouvez effectuer un équilibrage de charge sur `IPv6` que vers des cibles

IP, pas vers des cibles d'instance. Sans cette annotation, l'équilibrage de charge passe par `IPv4`.

```
service.beta.kubernetes.io/aws-load-balancer-ip-address-type: dualstack
```

Les Network Load Balancer sont créés avec le `internal` `aws-load-balancer-scheme`, par défaut. Vous pouvez lancer des Network Load Balancers dans n'importe quel sous-réseau du VPC de votre cluster, y compris les sous-réseaux qui n'ont pas été spécifiés lors de la création de votre cluster.

Kubernetes examine la table de routage de vos sous-réseaux afin d'identifier s'ils sont publics ou privés. Les sous-réseaux publics dispose d'un acheminement direct vers Internet en utilisant une passerelle Internet. Ce n'est pas le cas des sous-réseaux privés.

Si vous souhaitez créer un Network Load Balancer dans un sous-réseau public pour équilibrer la charge des nœuds Amazon EC2 (Fargate ne peut être que privé), spécifiez-le avec l'annotation suivante : `internet-facing`

```
service.beta.kubernetes.io/aws-load-balancer-scheme: "internet-facing"
```

Note

Pour des raisons de rétrocompatibilité, l'annotation `service.beta.kubernetes.io/aws-load-balancer-type: "nlb-ip"` est toujours prise en charge. Cependant, nous vous recommandons d'utiliser les annotations précédentes pour les nouveaux équilibreurs de charge au lieu de `service.beta.kubernetes.io/aws-load-balancer-type: "nlb-ip"`.

Important

Ne modifiez pas les annotations après la création de votre service. Si vous devez le modifier, supprimez l'objet de service et créez-le à nouveau avec la valeur souhaitée pour cette annotation.

Créer un Network Load Balancer — Cibles d'instance

- Le contrôleur d'équilibrage de charge du fournisseur de AWS cloud crée des équilibreurs de charge réseau avec des cibles d'instance uniquement. La version `2.2.0` et les versions ultérieures du AWS Load Balancer Controller créent également des Network Load Balancers avec des cibles d'instance. Nous vous recommandons de l'utiliser, plutôt que le contrôleur d'équilibrage de charge du fournisseur de AWS cloud, pour créer de nouveaux équilibreurs de charge réseau. Vous pouvez utiliser des cibles d'instance Network Load Balancer avec des pods déployés sur des EC2 nœuds Amazon, mais pas sur Fargate. Pour équilibrer la charge du trafic réseau entre les pods déployés sur Fargate, vous devez utiliser des cibles IP.

  Pour déployer un Network Load Balancer sur un sous-réseau privé, votre spécification de service doit comporter les annotations suivantes. Vous pouvez visualiser un exemple de manifeste de service avec les annotations. La `external` valeur de `aws-load-balancer-type` est ce qui pousse le AWS Load Balancer Controller, plutôt que le contrôleur d'équilibrage de charge du fournisseur de AWS cloud, à créer le Network Load Balancer.

  ```
  service.beta.kubernetes.io/aws-load-balancer-type: "external"
  service.beta.kubernetes.io/aws-load-balancer-nlb-target-type: "instance"
  ```

  Les Network Load Balancer sont créés avec le `internal` `aws-load-balancer-scheme`, par défaut. Pour les Network Load Balancer internes, votre cluster Amazon EKS doit être configuré pour utiliser au moins un sous-réseau privé dans votre VPC. Kubernetes examine la table de routage de vos sous-réseaux afin d'identifier s'ils sont publics ou privés. Les sous-réseaux publics dispose d'un acheminement direct vers Internet en utilisant une passerelle Internet. Ce n'est pas le cas des sous-réseaux privés.

  Si vous souhaitez créer un Network Load Balancer dans un sous-réseau public afin d'équilibrer la charge des EC2 nœuds Amazon, spécifiez-le à l'`internet-facing`aide de l'annotation suivante :

  ```
  service.beta.kubernetes.io/aws-load-balancer-scheme: "internet-facing"
  ```

  > **Important**
  >
  > Ne modifiez pas les annotations après la création de votre service. Si vous devez le modifier, supprimez l'objet de service et créez-le à nouveau avec la valeur souhaitée pour cette annotation.

(Facultatif) Déployer un exemple d'application

- Au moins un sous-réseau public ou privé dans votre VPC de cluster.
- Déployez le AWS Load Balancer Controller sur votre cluster. Pour de plus amples informations, veuillez consulter the section called " AWS Contrôleur Load Balancer". Nous recommandons la version `2.7.2` ou ultérieure.

  1. Si vous effectuez un déploiement vers Fargate, assurez-vous de disposer d'un sous-réseau privé disponible dans votre VPC et créez un profil Fargate. Si vous ne déployez pas sur Fargate, ignorez cette étape. Vous pouvez créer le profil en exécutant la commande suivante ou dans la AWS Management Console en utilisant les mêmes valeurs pour `name` et `namespace` que dans la commande. Remplacez les exemples de valeurs par les vôtres.

     ```
     eksctl create fargateprofile \
         --cluster my-cluster \
         --region region-code \
         --name nlb-sample-app \
         --namespace nlb-sample-app
     ```

  2. Déployez un exemple d'application.

     a. Créez un espace de nom pour l'application.

     ```
     kubectl create namespace nlb-sample-app
     ```

     b. Enregistrez le contenu suivant dans un fichier nommé `sample-deployment.yaml` sur votre ordinateur.

     ```
     apiVersion: apps/v1
     kind: Deployment
     metadata:
       name: nlb-sample-app
       namespace: nlb-sample-app
     spec:
       replicas: 3
       selector:
         matchLabels:
           app: nginx
       template:
         metadata:
           labels:
             app: nginx
     ```

```
    spec:
      containers:
        - name: nginx
          image: public.ecr.aws/nginx/nginx:1.23
          ports:
            - name: tcp
              containerPort: 80
```

c. Appliquez le fichier manifeste à votre cluster.

```
kubectl apply -f sample-deployment.yaml
```

3. Créez un service avec un Network Load Balancer interne qui équilibre la charge vers les cibles IP.

a. Enregistrez le contenu suivant dans un fichier nommé `sample-service.yaml` sur votre ordinateur. Si vous effectuez un déploiement vers des nœuds Fargate, supprimez la ligne `service.beta.kubernetes.io/aws-load-balancer-scheme: internet-facing`.

```
apiVersion: v1
kind: Service
metadata:
  name: nlb-sample-service
  namespace: nlb-sample-app
  annotations:
    service.beta.kubernetes.io/aws-load-balancer-type: external
    service.beta.kubernetes.io/aws-load-balancer-nlb-target-type: ip
    service.beta.kubernetes.io/aws-load-balancer-scheme: internet-facing
spec:
  ports:
    - port: 80
      targetPort: 80
      protocol: TCP
  type: LoadBalancer
  selector:
    app: nginx
```

b. Appliquez le fichier manifeste à votre cluster.

```
kubectl apply -f sample-service.yaml
```

4. Vérifiez que le service a été déployé.

```
kubectl get svc nlb-sample-service -n nlb-sample-app
```

L'exemple qui suit illustre un résultat.

```
NAME             TYPE           CLUSTER-IP         EXTERNAL-IP
                                                 PORT(S)        AGE
sample-service   LoadBalancer   10.100.240.137   k8s-nlbsampl-nlbsampl-xxxxxxxxxx-
xxxxxxxxxxxxxxxx.elb.region-code.amazonaws.com  80:32400/TCP   16h
```

> **Note**
>
> Les valeurs pour *10.100.240.137* et *xxxxxxxxxx* - *xxxxxxxxxxxxxxxx* seront différentes de celles de l'exemple de sortie (elles seront propres à votre équilibreur de charge) et *us-west-2* peuvent être différentes pour vous, selon AWS la région dans laquelle se trouve votre cluster.

5. Ouvrez l'Amazon EC2 AWS Management Console. Sélectionnez Target Groups (Groupes cibles) (sous Load Balancing [Équilibrage de charge]) dans le panneau de navigation de gauche. Dans la colonne Nom, sélectionnez le nom du groupe cible dont la valeur dans la colonne Équilibreur de charge correspond à une partie du nom dans la colonne `EXTERNAL-IP` de la sortie de l'étape précédente. Par exemple, vous sélectionneriez le groupe cible nommé `k8s-default-samplese-`*xxxxxxxxxx* si votre sortie était identique à la sortie précédente. Le Type de cible est `IP`, car cela a été spécifié dans l'exemple de manifeste de service.
6. Sélectionnez votre Target group (Groupe cible), puis cliquez sur l'onglet Targets (Cibles). Sous Cibles enregistrées, vous devez voir trois adresses IP des trois réplicas déployés à une étape précédente. Attendez que le statut de toutes les cibles soit sain avant de continuer. L'affichage du statut `healthy` pour toutes les cibles peut prendre plusieurs minutes. Les cibles peuvent avoir l'état `unhealthy` avant qu'il soit remplacé par `healthy`.
7. Envoyer le trafic vers le service en remplaçant *xxxxxxxxxx-xxxxxxxxxxxxxxxx* et *us-west-2* par les valeurs renvoyées dans la sortie de l'étape précédente pour `EXTERNAL-IP`. Si vous avez déployé sur un sous-réseau privé, vous devez afficher la page à partir d'un appareil de votre VPC, tel qu'un hôte bastion. Pour plus d'informations, consultez la section Hôtes bastions Linux sur AWS.

```
curl k8s-default-samplese-xxxxxxxxxx-xxxxxxxxxxxxxxxx.elb.region-code.amazonaws.com
```

L'exemple qui suit illustre un résultat.

```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
[...]
```

8. Une fois que vous avez terminé avec le déploiement, le service et l'espace de noms d'exemple, supprimez-les.

```
kubectl delete namespace nlb-sample-app
```

Routage du trafic des applications et du trafic HTTP avec des équilibreurs de charge Application Load Balancer

> **Note**
>
> Nouveau : le mode automatique Amazon EKS automatise les tâches courantes liées à l'équilibrage de charge. Pour en savoir plus, consultez :
>
> - the section called "Déploiement de l'équilibreur de charge"
> - the section called "Création d'une classe de ressources Ingress"

Lorsque vous créez un Kubernetes`ingress`, un Application AWS Load Balancer (ALB) est configuré pour équilibrer la charge du trafic des applications. Pour en savoir plus, consultez Qu'est-ce qu'un Application Load Balancer ? dans le guide de l'utilisateur des équilibreurs de charge d'application et Ingress dans la documentation de Kubernetes. ALBs peut être utilisé avec des pods déployés sur des nœuds ou sur AWS Fargate. Vous pouvez déployer un ALB sur des sous-réseaux publics ou privés.

Le trafic des applications est équilibré au `L7` du modèle OSI. Pour équilibrer la charge du trafic réseau au `L4`, déployez un `service` Kubernetes de type `LoadBalancer`. Ce type fournit un AWS Network Load Balancer. Pour de plus amples informations, veuillez consulter the section called "Répartition de charge réseau". Pour en savoir plus sur les différences entre les deux types d'équilibrage de charge, consultez les fonctionnalités d'Elastic Load Balancing sur le AWS site Web.

Conditions préalables

Pour pouvoir équilibrer la charge du trafic d'une application, vous devez remplir les conditions suivantes.

- Disposer d'un cluster. Si vous n'avez pas de cluster, consultez Démarrer. Si vous devez mettre à jour la version d'un cluster existant, consultez the section called "Mises à jour de la version Kubernetes".
- Déployez le AWS Load Balancer Controller sur votre cluster. Pour de plus amples informations, veuillez consulter the section called " AWS Contrôleur Load Balancer". Nous recommandons la version `2.7.2` ou ultérieure.
- Au moins deux sous-réseaux dans des zones de disponibilité différentes. Le AWS Load Balancer Controller choisit un sous-réseau dans chaque zone de disponibilité. Lorsque plusieurs sous-réseaux étiquetés sont trouvés dans une zone de disponibilité, le contrôleur choisit le sous-réseau dont l'ID vient en premier par ordre lexicographique. Chaque sous-réseau doit disposer au moins huit adresses IP disponibles.

  Si vous utilisez plusieurs groupes de sécurité attachés au composant master, un seul groupe de sécurité doit être étiqueté comme suit. Remplacez *`my-cluster`* par le nom de votre cluster.

  - Clé : `kubernetes.io/cluster/<my-cluster>`
  - Valeur : `shared` ou `owned`
- Si vous utilisez la version du AWS Load Balancer Controller `2.1.1` ou une version antérieure, les sous-réseaux doivent être balisés au format suivant. Si vous utilisez la version `2.1.2` ou une version ultérieure, l'étiquetage est facultatif. Cependant, nous vous recommandons d'étiqueter un sous-réseau dans les cas suivants. Plusieurs clusters s'exécutent dans le même VPC ou plusieurs AWS services partagent des sous-réseaux au sein d'un VPC. Sinon, vous souhaitez avoir plus de contrôle sur l'endroit où les équilibreurs de charge sont alloués pour chaque cluster. Remplacez *`my-cluster`* par le nom de votre cluster.
  - Clé : `kubernetes.io/cluster/<my-cluster>`
  - Valeur : `shared` ou `owned`
- Vos sous-réseaux publics et privés doivent répondre aux critères suivants : C'est le cas sauf si vous spécifiez explicitement un sous-réseau IDs sous forme d'annotation sur un service ou un objet d'entrée. Supposons que vous provisionnez des équilibreurs de charge en spécifiant explicitement le sous-réseau IDs sous forme d'annotation sur un service ou un objet d'entrée. Dans ce cas, Kubernetes et le contrôleur d'équilibreur de AWS charge utilisent directement ces sous-réseaux pour créer l'équilibreur de charge et les balises suivantes ne sont pas requises.

- Sous-réseaux privés : doivent être étiquetés dans le format suivant. Cela permet à Kubernetes et au contrôleur d'équilibreur de AWS charge de savoir que les sous-réseaux peuvent être utilisés pour les équilibreurs de charge internes. Si vous utilisez `eksctl` un AWS CloudFormation modèle Amazon EKS pour créer votre VPC après le 26 mars 2020, les sous-réseaux sont balisés de manière appropriée lors de leur création. Pour plus d'informations sur les modèles de AWS CloudFormation VPC Amazon EKS, consultez. the section called "Création d'un VPC"
  - Clé : `kubernetes.io/role/internal-elb`
  - Valeur : 1
- Sous-réseau publics : doivent être étiquetés dans le format suivant. Ainsi, Kubernetes sait qu'il doit utiliser uniquement les sous-réseaux qui ont été spécifiés pour les équilibreurs de charge externes. Cela empêche Kubernetes de sélectionner un sous-réseau public dans chaque zone de disponibilité (en se basant sur l'ordre lexicographique des ID de sous-réseau). Si vous utilisez `eksctl` un AWS CloudFormation modèle Amazon EKS pour créer votre VPC après le 26 mars 2020, les sous-réseaux sont balisés de manière appropriée lors de leur création. Pour plus d'informations sur les modèles de AWS CloudFormation VPC Amazon EKS, consultez. the section called "Création d'un VPC"
  - Clé : `kubernetes.io/role/elb`
  - Valeur : 1

  Si les étiquettes de rôle des sous-réseaux ne sont pas ajoutées explicitement, le contrôleur de service Kubernetes examine la table de routage des sous-réseaux VPC de votre cluster. Cela permet de déterminer si le sous-réseau est privé ou public. Nous vous recommandons de ne pas vous fier à ce comportement. Ajoutez plutôt explicitement les identifications de rôle privées ou publiques. Le AWS Load Balancer Controller n'examine pas les tables de routage. Il est également nécessaire que les identifications privées et publiques soient présentes pour que la découverte automatique fonctionne.
- Le AWS Load Balancer Controller crée ALBs les AWS ressources de support nécessaires chaque fois qu'une ressource d'entrée Kubernetes est créée sur le cluster avec l'annotation. `kubernetes.io/ingress.class: alb` La ressource Ingress configure l'ALB pour acheminer le trafic HTTP ou HTTPS vers différents pods du cluster. Pour vous assurer que vos objets d'entrée utilisent le AWS Load Balancer Controller, ajoutez l'annotation suivante à votre spécification d'entrée Kubernetes. Pour plus d'informations, consultez Spécification d'entrée sur GitHub.

```
annotations:
    kubernetes.io/ingress.class: alb
```

> **Note**
>
> Si vous effectuez un équilibrage de charge vers des pods `IPv6`, ajoutez l'annotation suivante à votre spécification Ingress. Vous ne pouvez effectuer un équilibrage de charge sur `IPv6` que vers des cibles IP, pas vers des cibles d'instance. Sans cette annotation, l'équilibrage de charge passe par `IPv4`.

```
alb.ingress.kubernetes.io/ip-address-type: dualstack
```

- Le AWS Load Balancer Controller prend en charge les modes de trafic suivants :
  - Instance : enregistre les nœuds de votre cluster comme cibles de l'ALB. Le trafic atteignant l'ALB est acheminé vers le `NodePort` de votre service, puis transmis par proxy à vos pods. Il s'agit du mode de trafic par défaut. Vous pouvez également le spécifier explicitement avec l'annotation `alb.ingress.kubernetes.io/target-type: instance`.

    > **Note**
    >
    > Votre service Kubernetes doit spécifier le type `NodePort` ou `LoadBalancer` pour utiliser ce mode de trafic.

  - IP : enregistre les pods comme cibles pour l'ALB. Le trafic atteignant l'ALB est directement acheminé vers des pods de votre service. Vous devez spécifier l'annotation `alb.ingress.kubernetes.io/target-type: ip` pour pouvoir utiliser ce mode de trafic. Le type de cible IP est requis lorsque les pods cibles s'exécutent sur Fargate ou sur des nœuds hybrides Amazon EKS.
- Pour étiqueter une balise ALBs créée par le contrôleur, ajoutez l'annotation suivante au contrôleur : `alb.ingress.kubernetes.io/tags` Pour une liste de toutes les annotations disponibles prises en charge par le AWS Load Balancer Controller, consultez la section Annotations d'entrée sur. GitHub
- La mise à niveau ou la rétrogradation de la version du contrôleur ALB peut introduire des changements de rupture pour les fonctionnalités qui en dépendent. Pour plus d'informations sur les changements importants introduits dans chaque version, consultez les notes de mise à jour du contrôleur ALB sur GitHub.

Réutilisation ALBs avec des groupes d'entrée

Vous pouvez partager un équilibreur de charge Application Load Balancer sur plusieurs ressources d'entrée à l'aide des `IngressGroups`.

Pour joindre une ressource d'entrée à un groupe, ajoutez l'annotation suivante à une spécification de ressource d'entrée Kubernetes.

```
alb.ingress.kubernetes.io/group.name: my-group
```

Le nom du groupe doit :

- Contenir 63 caractères maximum.
- Contenir des minuscules, des chiffres, des `-` et des `.`.
- Commencer et se terminer par un chiffre ou une lettre.

Le contrôleur fusionne automatiquement les règles d'entrée pour toutes les entrées du même groupe d'entrées. Il les prend en charge avec un seul ALB. La plupart des annotations qui sont définies sur une ressource d'entrée ne s'appliquent qu'aux chemins définis par cette ressource. Par défaut, les ressources Ingress n'appartiennent à aucun groupe d'entrée.

> **Warning**
>
> Risque potentiel pour la sécurité
> Ne spécifiez un groupe d'entrée pour une ressource d'entrée que lorsque tous les utilisateurs Kubernetes qui ont l'autorisation RBAC de créer ou de modifier des ressources Ingress se trouvent dans la même limite d'approbation. Si vous ajoutez l'annotation avec un nom de groupe, d'autres utilisateurs Kubernetes peuvent créer ou modifier leurs ressources d'entrée pour appartenir au même groupe d'entrée. Cela peut entraîner un comportement indésirable, comme l'écrasement des règles existantes par des règles de priorité supérieure.

Vous pouvez ajouter le numéro d'ordre de votre ressource d'entrée.

```
alb.ingress.kubernetes.io/group.order: '10'
```

Le numéro peut être 1-1000. Le numéro le plus bas de toutes les ressources d'entrée d'un même groupe d'entrée est évalué en premier. Toutes les ressources d'entrée sans cette annotation sont

évaluées avec la valeur zéro. Les règles dupliquées avec un numéro supérieur peuvent écraser les règles avec un numéro inférieur. Par défaut, l'ordre des règles entre les ressources d'entrée d'un même groupe d'entrée est déterminé de manière lexicographique sur la base de l'espace de noms et du nom.

> **Important**
>
> Assurez-vous que chaque ressource d'un groupe d'entrée dispose d'un numéro de priorité unique. Vous ne pouvez pas utiliser des numéros d'ordre dupliqués dans les ressources Ingress.

(Facultatif) Déployer un exemple d'application

- Au moins un sous-réseau public ou privé dans votre VPC de cluster.
- Déployez le AWS Load Balancer Controller sur votre cluster. Pour de plus amples informations, veuillez consulter the section called " AWS Contrôleur Load Balancer". Nous recommandons la version `2.7.2` ou ultérieure.

Vous pouvez exécuter l'exemple d'application sur un cluster doté de EC2 nœuds Amazon, de Fargate Pods ou des deux.

1. Si vous ne déployez pas sur Fargate, ignorez cette étape. Si vous déployez sur Fargate, créez un profil Fargate. Vous pouvez créer le profil en exécutant la commande suivante ou dans la AWS Management Console en utilisant les mêmes valeurs pour `name` et `namespace` que dans la commande. Remplacez les exemples de valeurs par les vôtres.

   ```
   eksctl create fargateprofile \
       --cluster my-cluster \
       --region region-code \
       --name alb-sample-app \
       --namespace game-2048
   ```

2. Déployez le jeu 2048 comme exemple d'application pour vérifier que le AWS Load Balancer Controller crée AWS un ALB à la suite de l'objet d'entrée. Effectuez les étapes correspondant au type de sous-réseau sur lequel vous déployez.

   a. Si vous déployez des Pods dans un cluster créé avec la famille `IPv6`, passez à l'étape suivante.

- Publique :

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.14.1/docs/examples/2048/2048_full.yaml
```

- Privé :

  A. Téléchargez le manifeste.

  ```
  curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.14.1/docs/examples/2048/2048_full.yaml
  ```

  B. Modifiez le fichier et trouvez la ligne qui contient `alb.ingress.kubernetes.io/scheme: internet-facing`.

  C. Remplacez *`internet-facing`* par `internal` et enregistrez le fichier.

  D. Appliquez le manifeste à votre cluster.

  ```
  kubectl apply -f 2048_full.yaml
  ```

b. Si vous effectuez un déploiement sur des Pods dans un cluster que vous avez créé avec la IPv6 famille, procédez comme suit.

   i. Téléchargez le manifeste.

   ```
   curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.14.1/docs/examples/2048/2048_full.yaml
   ```

   ii. Ouvrez le fichier dans un éditeur et ajoutez la ligne suivante aux annotations de la spécification d'entrée.

   ```
   alb.ingress.kubernetes.io/ip-address-type: dualstack
   ```

   iii. Si vous effectuez un équilibrage de charge vers des pods internes plutôt que vers des pods accessibles depuis Internet, remplacez la valeur `alb.ingress.kubernetes.io/scheme:` *`internet-facing`* par `alb.ingress.kubernetes.io/scheme: internal`

   iv. Enregistrez le fichier.

   v. Appliquez le manifeste à votre cluster.

   ```
   kubectl apply -f 2048_full.yaml
   ```

3. Après quelques minutes, vérifiez que la ressource d'entrée a été créée en utilisant la commande suivante.

```
kubectl get ingress/ingress-2048 -n game-2048
```

L'exemple qui suit illustre un résultat.

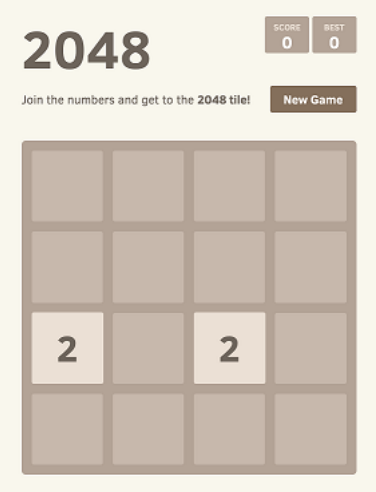
```
NAME           CLASS    HOSTS   ADDRESS
                     PORTS   AGE
ingress-2048   <none>   *       k8s-game2048-ingress2-xxxxxxxxxx-yyyyyyyyyy.region-
code.elb.amazonaws.com   80      2m32s
```

> **Note**
>
> Si vous avez créé l'équilibreur de charge dans un sous-réseau privé, la valeur sous `ADDRESS` dans la sortie précédente est préfacée avec `internal-`.

Si votre entrée n'a pas été correctement créée au bout de quelques minutes, exécutez la commande suivante pour afficher les journaux du AWS Load Balancer Controller. Ces journaux peuvent contenir des messages d'erreur que vous pouvez utiliser pour diagnostiquer les problèmes de votre déploiement.

```
kubectl logs -f -n kube-system -l app.kubernetes.io/instance=aws-load-balancer-
controller
```

1. Si vous l'avez déployé dans un sous-réseau public, ouvrez un navigateur et naviguez vers l'URL `ADDRESS` de la sortie de commande précédente pour voir l'exemple d'application. Si rien ne s'affiche, actualisez votre navigateur et réessayez. Si vous avez déployé sur un sous-réseau privé, vous devez afficher la page à partir d'un appareil de votre VPC, tel qu'un hôte bastion. Pour plus d'informations, consultez la section Hôtes bastions Linux sur AWS.



2. Lorsque vous avez terminé de tester l'exemple d'application, supprimez-le avec les commandes suivantes.
   - Si vous avez appliqué le manifeste plutôt que d'appliquer une copie que vous avez téléchargée, utilisez la commande suivante.

     ```
     kubectl delete -f https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.14.1/docs/examples/2048/2048_full.yaml
     ```

   - Si vous avez téléchargé et modifié le manifeste, utilisez la commande suivante.

     ```
     kubectl delete -f 2048_full.yaml
     ```

Limiter les adresses IP externes pouvant être attribuées aux services

Les services Kubernetes peuvent être atteints depuis l'intérieur d'un cluster via :

- Une adresse IP de cluster affectée automatiquement par Kubernetes
- Toute adresse IP que vous spécifiez pour la propriété `externalIPs` dans une spécification de service. Les adresses IP externes ne sont pas gérées par Kubernetes et relèvent de la responsabilité de l'administrateur du cluster. Les adresses IP externes spécifiées avec `externalIPs` sont différentes de l'adresse IP externe affectée à un service de type `LoadBalancer` par un fournisseur de cloud.

Pour en savoir plus sur les services Kubernetes, consultez Service dans la documentation Kubernetes. Vous pouvez restreindre les adresses IP qui peuvent être spécifiées pour `externalIPs` dans une spécification de service.

1. Déployez `cert-manager` pour gérer les certificats webhook. Pour plus d'informations, consultez la documentation cert-manager.

```
kubectl apply -f https://github.com/jetstack/cert-manager/releases/download/v1.5.4/
cert-manager.yaml
```

2. Vérifiez que les pods `cert-manager` sont en cours d'exécution.

```
kubectl get pods -n cert-manager
```

   L'exemple qui suit illustre un résultat.

```
NAME                                       READY   STATUS    RESTARTS   AGE
cert-manager-58c8844bb8-nlx7q              1/1     Running   0          15s
cert-manager-cainjector-745768f6ff-696h5   1/1     Running   0          15s
cert-manager-webhook-67cc76975b-4v4nk      1/1     Running   0          14s
```

3. Vérifiez vos services existants pour vous assurer qu'aucun d'entre eux ne dispose d'adresses IP externes qui ne sont pas contenues dans le bloc CIDR auquel vous voulez limiter les adresses.

```
kubectl get services -A
```

   L'exemple qui suit illustre un résultat.

```
NAMESPACE                   NAME                                      TYPE
 CLUSTER-IP       EXTERNAL-IP     PORT(S)         AGE
cert-manager                cert-manager                              ClusterIP
 10.100.102.137   <none>          9402/TCP        20m
cert-manager                cert-manager-webhook                      ClusterIP
 10.100.6.136     <none>          443/TCP         20m
default                     kubernetes                                ClusterIP
 10.100.0.1       <none>          443/TCP         2d1h
externalip-validation-system   externalip-validation-webhook-service   ClusterIP
 10.100.234.179   <none>          443/TCP         16s
kube-system                 kube-dns                                  ClusterIP
 10.100.0.10      <none>          53/UDP,53/TCP   2d1h
```

```
my-namespace                   my-service                                  ClusterIP
 10.100.128.10    192.168.1.1     80/TCP          149m
```

Si l'une des valeurs est une adresse IP qui ne se trouve pas dans le bloc auquel vous voulez restreindre l'accès, vous devrez modifier les adresses pour qu'elles se trouvent dans le bloc, puis redéployer les services. Par exemple, le service `my-service` dans la sortie précédente a une adresse IP externe qui ne se trouve pas dans le bloc CIDR de l'exemple de l'étape 5.

4. Téléchargez le manifeste du webhook IP externe. Vous pouvez également afficher le code source pour le webhook sur GitHub.

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/docs/externalip-webhook.yaml
```

5. Spécifiez des blocs d'adresse CIDR. Ouvrez le fichier téléchargé dans l'éditeur et supprimez le `\#` au début des lignes suivantes.

```
#args:
#- --allowed-external-ip-cidrs=10.0.0.0/8
```

Remplacez `10.0.0.0/8` par votre propre bloc d'adresse CIDR. Vous pouvez spécifier autant de blocs que vous le souhaitez. Si vous spécifiez plusieurs blocs, ajoutez une virgule entre les blocs.

6. Si votre cluster ne se trouve pas dans la région AWS `us-west-2`, remplacez `us-west-2`, `602401143452` et `amazonaws.com` dans le fichier par les commandes suivantes. Avant d'exécuter les commandes, remplacez *`region-code`* et *`111122223333`* par la valeur correspondant à votre région AWS dans la liste de la section Afficher les registres d'images de conteneurs Amazon pour les modules complémentaires Amazon EKS.

```
sed -i.bak -e 's|602401143452|111122223333|' externalip-webhook.yaml
sed -i.bak -e 's|us-west-2|region-code|' externalip-webhook.yaml
sed -i.bak -e 's|amazonaws.com||' externalip-webhook.yaml
```

7. Appliquez le manifeste à votre cluster.

```
kubectl apply -f externalip-webhook.yaml
```

Toute tentative de déploiement d'un service sur votre cluster avec une adresse IP spécifiée pour `externalIPs` qui ne figure pas dans les blocs que vous avez spécifiés à l'étape Spécifier les blocs CIDR échouera.

Copier une image de conteneur d'un référentiel vers un autre référentiel

Cette rubrique décrit comment extraire une image de conteneur d'un référentiel auquel vos nœuds n'ont pas accès et envoyer l'image vers un référentiel auquel vos nœuds ont accès. Vous pouvez transmettre l'image à Amazon ECR ou à un référentiel alternatif auquel vos nœuds ont accès.

- Docker Engine installé et configuré sur votre ordinateur. Pour connaître la marche à suivre, consultez la rubrique Installer Docker Engine de la documentation Docker.
- Version `2.12.3` ou version ultérieure `1.27.160` ou version ultérieure de l'interface de ligne de AWS commande (AWS CLI) installée et configurée sur votre appareil ou AWS CloudShell. Pour vérifier votre version actuelle, utilisez `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Les gestionnaires de packages tels que `yum` Homebrew pour macOS ont souvent plusieurs versions de retard sur la dernière version de la AWS CLI. `apt-get` Pour installer la dernière version, consultez la section Installation et configuration rapide avec aws configure dans le Guide de l'utilisateur de l'interface de ligne de AWS commande. La version de la AWS CLI installée AWS CloudShell peut également avoir plusieurs versions de retard par rapport à la dernière version. Pour le mettre à jour, consultez la section Installation de la AWS CLI dans votre répertoire de base dans le guide de AWS CloudShell l'utilisateur.
- Un point de terminaison d'un VPC d'interface pour Amazon ECR si vous souhaitez que vos nœuds extraient des images de conteneur ou transmettent des images de conteneur vers un référentiel Amazon ECR privé sur le réseau d'Amazon. Pour de plus amples informations, veuillez consulter Créer des points de terminaison de VPC pour Amazon ECR dans le Guide de l'utilisateur Amazon Elastic Container Registry.

Effectuez les étapes suivantes pour extraire une image de conteneur d'un référentiel et la transférer vers votre propre référentiel. Dans les exemples suivants fournis dans cette rubrique, l'image du plug-in Amazon VPC CNI pour l'assistant de métriques Kubernetes est extraite. Lorsque vous suivez ces étapes, assurez-vous de remplacer les exemples de valeurs par vos propres valeurs.

1. Si vous ne disposez pas déjà d'un référentiel Amazon ECR ou d'un autre référentiel, créez-en un auquel vos nœuds ont accès. La commande suivante crée un référentiel privé Amazon ECR. Un nom de référentiel privé Amazon ECR doit commencer par une lettre. Il peut uniquement contenir des lettres minuscules, des chiffres, des tirets (-), des traits de soulignement (_) et des barres obliques (/). Pour de plus amples informations, veuillez consulter Créer un référentiel privé dans le Guide de l'utilisateur Amazon Elastic Container Registry.

Vous pouvez remplacer.*cni-metrics-helper* par ce que vous souhaitez. Il est recommandé de créer un référentiel distinct pour chaque image. Nous vous le recommandons, car les identifications d'image doivent être uniques au sein d'un référentiel. Remplacez *region-code* par une AWS région prise en charge par Amazon ECR.

```
aws ecr create-repository --region region-code --repository-name cni-metrics-helper
```

2. Déterminez le registre, le référentiel et l'identification (facultatif) de l'image que vos nœuds doivent extraire. Ces informations se trouvent dans le format `registry/repository[:tag]`.

   De nombreuses rubriques Amazon EKS sur l'installation d'images nécessitent que vous appliquiez un fichier manifeste ou que vous installiez l'image à l'aide des Charts de Helm. Cependant, avant d'appliquer un fichier manifeste ou d'installer une charte de Helm, visualisez d'abord le contenu du fichier `values.yaml` manifeste ou de la charte. De cette façon, vous pouvez déterminer le registre, le référentiel et l'identification à extraire.

   Par exemple, vous pouvez trouver la ligne suivante dans le fichier manifeste du plug-in Amazon VPC CNI pour l'assistant de métriques Kubernetes. Le registre est `602401143452.dkr.ecr.us-west-2.amazonaws.com`, qui est un registre privé Amazon ECR. Le référentiel est `cni-metrics-helper`.

   ```
   image: "602401143452.dkr.ecr.us-west-2.amazonaws.com/cni-metrics-helper:v1.12.6"
   ```

   Vous pouvez voir les variantes suivantes pour un emplacement d'image :

   - Seul `repository-name:tag`. Dans ce cas, `docker.io` est généralement le registre, mais non spécifié puisque Kubernetes l'ajoute par défaut à un nom de référentiel si aucun registre n'est spécifié.
   - `repository-name/repository-namespace/repository:tag`. Un espace de noms de référentiel est facultatif, mais est parfois spécifié par le propriétaire du référentiel pour catégoriser les images. Par exemple, toutes les EC2 images Amazon de la galerie publique Amazon ECR utilisent l'espace de `aws-ec2` noms.

     Avant d'installer une image avec Helm, affichez le fichier `values.yaml` Helm pour déterminer l'emplacement de l'image. Par exemple, le fichier values.yaml du plug-in Amazon VPC CNI pour l'assistant de métriques Kubernetes comprend les lignes suivantes.

     ```
     image:
     ```

```
  region: us-west-2
  tag: v1.12.6
  account: "602401143452"
  domain: "amazonaws.com"
```

3. Extrayez l'image conteneur spécifiée dans le fichier manifeste.

   a. Si vous extrayez des images d'un registre public, tel que la galerie publique Amazon ECR, vous pouvez passer à la sous-étape suivante, car l'authentification n'est pas requise. Dans cet exemple, vous vous authentifiez auprès d'un registre privé Amazon ECR contenant le référentiel pour l'image d'assistance des métriques CNI. Amazon EKS conserve l'image dans chaque registre répertorié dans la section Afficher les registres d'images de conteneurs Amazon pour les modules complémentaires Amazon EKS. Vous pouvez vous authentifier auprès de n'importe quel registre en remplaçant *602401143452* et *region-code* par les informations d'un registre différent. Il existe un registre distinct pour chaque région AWS dans laquelle Amazon EKS est pris en charge.

   ```
   aws ecr get-login-password --region region-code | docker login --username AWS --
   password-stdin 602401143452.dkr.ecr.region-code.amazonaws.com
   ```

   b. Extrayez l'image. Dans cet exemple, vous effectuez une extraction à partir du registre auquel vous vous êtes authentifié au cours de la sous-étape précédente. Remplacez *602401143452* et *region-code* par les informations que vous avez fournies à la sous-étape précédente.

   ```
   docker pull 602401143452.dkr.ecr.region-code.amazonaws.com/cni-metrics-
   helper:v1.12.6
   ```

4. Étiquetez l'image que vous avez extraite avec votre registre, votre référentiel et votre identification. L'exemple suivant suppose que vous avez extrait l'image du fichier manifeste et que vous allez la transmettre au référentiel privé Amazon ECR que vous avez créé à la première étape. Remplacez *111122223333* par votre ID de compte. *region-code*Remplacez-le par la AWS région dans laquelle vous avez créé votre référentiel privé Amazon ECR.

   ```
   docker tag cni-metrics-helper:v1.12.6 111122223333.dkr.ecr.region-code.amazonaws.com/
   cni-metrics-helper:v1.12.6
   ```

5. Authentifiez auprès de votre registre. Dans cet exemple, vous vous authentifiez auprès du registre privé Amazon ECR que vous avez créé à la première étape. Pour plus d'informations, veuillez consulter Authentification de registre dans le Guide de l'utilisateur Amazon Elastic Container Registry.

```
aws ecr get-login-password --region region-code | docker login --username AWS --
password-stdin 111122223333.dkr.ecr.region-code.amazonaws.com
```

6. Transmettez l'image à votre référentiel. Dans cet exemple, vous transmettez l'image au référentiel privé Amazon ECR que vous avez créé à la première étape. Pour plus d'informations, consultez Transmission d'une image Dockerdans le Guide de l'utilisateur Amazon Elastic Container Registry.

```
docker push 111122223333.dkr.ecr.region-code.amazonaws.com/cni-metrics-helper:v1.12.6
```

7. Mettez à jour le fichier manifeste que vous avez utilisé pour déterminer l'image lors d'une étape précédente avec le `registry/repository:tag` pour l'image que vous avez transmise. Si vous effectuez l'installation à l'aide d'une charte de Helm, il existe souvent une option permettant de spécifier le `registry/repository:tag`. Lors de l'installation du graphique, spécifiez le `registry/repository:tag` pour l'image que vous avez transmise à votre référentiel.

Affichage des registres d'images conteneurs Amazon pour les modules complémentaires Amazon EKS

Lorsque vous déployez des modules complémentaires AWS Amazon EKS dans votre cluster, vos nœuds extraient les images de conteneur requises à partir du registre spécifié dans le mécanisme d'installation du module complémentaire, comme un manifeste d'installation ou un fichier Helm `values.yaml`. Les images sont extraites à partir d'un référentiel privé Amazon EKS Amazon ECR. Amazon EKS réplique les images dans un référentiel dans chaque AWS région prise en charge par Amazon EKS. Vos nœuds peuvent extraire l'image du conteneur sur Internet à partir de l'un des registres suivants. Vos nœuds peuvent également extraire l'image sur le réseau d'Amazon si vous avez créé un point de terminaison VPC d'interface pour Amazon ECR (AWS PrivateLink) dans votre VPC. Les registres nécessitent une authentification avec un compte AWS IAM. Vos nœuds s'authentifient à l'aide du rôle IAM du nœud Amazon EKS, qui dispose des autorisations associées dans la politique IAM EC2 ContainerRegistryReadOnly gérée par Amazon.

AWS Région	Registre
af-south-1	877085696533.dkr.ecr.af-south-1.amazonaws.com

AWS Région	Registre
ap-east-1	800184023465.dkr.ecr.ap-east-1.amazonaws.com
ap-east-2	533267051163.dkr. ecr.ap-east-1.amazonaws.com
ap-southeast-3	296578399912.dkr.ecr.ap-southeast-3.amazonaws.com
ap-south-2	900889452093.dkr.ecr.ap-south-2.amazonaws.com
ap-southeast-4	491585149902.dkr.ecr.ap-southeast-4.amazonaws.com
ap-southeast-5	151610086707.dkr. ecr.ap-southeast-5.amazonaws.com
ap-southeast-6	333609536671.dkr. ecr.ap-southeast-6.amazonaws.com
ap-south-1	602401143452.dkr.ecr.ap-south-1.amazonaws.com
ap-northeast-3	602401143452.dkr.ecr.ap-northeast-3.amazonaws.com
ap-northeast-2	602401143452.dkr.ecr.ap-northeast-2.amazonaws.com
ap-southeast-1	602401143452.dkr.ecr.ap-southeast-1.amazonaws.com
ap-southeast-2	602401143452.dkr.ecr.ap-southeast-2.amazonaws.com
ap-southeast-7	121268973566.dkr. ecr.ap-southeast-7.amazonaws.com

AWS Région	Registre
ap-northeast-1	602401143452.dkr.ecr.ap-northeast-1.amazonaws.com
cn-north-1	918309763551.dkr.ecr.cn-north-1.amazonaws.com.cn/
cn-northwest-1	961992271922.dkr.ecr.cn-northwest-1.amazonaws.com.cn/
eu-central-1	602401143452.dkr.ecr.eu-central-1.amazonaws.com
eu-west-1	602401143452.dkr.ecr.eu-west-1.amazonaws.com
eu-west-2	602401143452.dkr.ecr.eu-west-2.amazonaws.com
eu-south-1	590381155156.dkr.ecr.eu-south-1.amazonaws.com
eu-west-3	602401143452.dkr.ecr.eu-west-3.amazonaws.com
eu-south-2	455263428931.dkr.ecr.eu-south-2.amazonaws.com
eu-north-1	602401143452.dkr.ecr.eu-north-1.amazonaws.com
eu-central-2	900612956339.dkr.ecr.eu-central-2.amazonaws.com
il-central-1	066635153087.dkr.ecr.il-central-1.amazonaws.com
mx-central-1	730335286997.dkr.ecr.mx-central-1.amazonaws.com

AWS Région	Registre
me-south-1	558608220178.dkr.ecr.me-south-1.amazonaws.com
me-central-1	759879836304.dkr.ecr.me-central-1.amazonaws.com
us-east-1	602401143452.dkr.ecr.us-east-1.amazonaws.com
us-east-2	602401143452.dkr.ecr.us-east-2.amazonaws.com
us-west-1	602401143452.dkr.ecr.us-west-1.amazonaws.com
us-west-2	602401143452.dkr.ecr.us-west-2.amazonaws.com
ca-central-1	602401143452.dkr.ecr.ca-central-1.amazonaws.com
ca-west-1	761377655185.dkr.ecr.ca-west-1.amazonaws.com
sa-east-1	602401143452.dkr.ecr.sa-east-1.amazonaws.com
us-gov-east-1	151742754352.dkr.ecr. us-gov-east-1. amazonaws.com
us-gov-west-1	013241004608.dkr.ecr. us-gov-west-1. amazonaws.com
eusc-de-east-1	877088126301.dkr.ecr. eusc-de-east-1.amazonaws.eu

Modules complémentaires Amazon EKS

Un module complémentaire est un logiciel qui fournit des capacités opérationnelles de prise en charge aux applications Kubernetes, mais qui n'est pas spécifique à l'application. Cela inclut des logiciels tels que des agents d'observabilité ou des pilotes Kubernetes qui permettent au cluster d'interagir avec les AWS ressources sous-jacentes pour la mise en réseau, le calcul et le stockage. Les logiciels complémentaires sont généralement conçus et maintenus par la communauté Kubernetes, par des fournisseurs de cloud tels que ou par des fournisseurs AWS tiers. Amazon EKS installe automatiquement des modules complémentaires autogérés tels que le plug-in Amazon VPC CNI pour Kubernetes, `kube-proxy`, et CoreDNS pour chaque cluster. Notez que le module complémentaire VPC CNI n'est pas compatible avec les nœuds hybrides Amazon EKS et ne se déploie pas sur les nœuds hybrides. Vous pouvez modifier la configuration par défaut des modules complémentaires et les mettre à jour quand vous le souhaitez.

Les modules complémentaires Amazon EKS fournissent l'installation et la gestion d'un ensemble organisé de modules complémentaires pour les clusters Amazon EKS. Tous les modules complémentaires Amazon EKS incluent les derniers correctifs de sécurité et corrections de bogues et sont validés par Amazon EKS AWS pour fonctionner avec Amazon EKS. Les modules complémentaires Amazon EKS vous permettent de garantir systématiquement que vos clusters Amazon EKS sont sécurisés et stables. Ils vous permettent aussi de réduire le travail nécessaire pour installer, configurer et mettre à jour les modules complémentaires. Si un module complémentaire autogéré tel que `kube-proxy` est déjà en cours d'exécution sur votre cluster et est disponible en tant que module complémentaire Amazon EKS, vous pouvez installer le module complémentaire Amazon EKS `kube-proxy` pour commencer à bénéficier des fonctionnalités des modules complémentaires Amazon EKS.

Vous pouvez mettre à jour des champs de configuration spécifiques gérés par Amazon EKS pour les modules complémentaires Amazon EKS via l'API Amazon EKS. Une fois le module complémentaire démarré, vous pouvez également modifier les champs de configuration non gérés par Amazon EKS directement dans le cluster Kubernetes. Cela inclut la définition de champs de configuration spécifiques pour un module complémentaire, le cas échéant. Ces modifications ne sont pas remplacées par Amazon EKS une fois qu'elles sont effectuées. Ceci est rendu possible par la fonction apply côté serveur de Kubernetes. Pour de plus amples informations, veuillez consulter the section called "Champs que vous pouvez personnaliser".

Vous pouvez utiliser des modules complémentaires Amazon EKS avec n'importe quel type de nœud Amazon EKS. Pour de plus amples informations, veuillez consulter Gestion du Calcul.

Vous pouvez ajouter, mettre à jour ou supprimer des modules complémentaires Amazon EKS à l'aide de l'API Amazon EKS AWS Management Console, de la AWS CLI et`eksctl`. Vous pouvez également créer des modules complémentaires Amazon EKS à l'aide de AWS CloudFormation.

Considérations

Tenez compte des éléments suivants lorsque vous utilisez les modules complémentaires Amazon EKS :

- Pour configurer des modules complémentaires pour le cluster, votre principal IAM doit disposer des autorisations IAM pour utiliser des modules complémentaires. Pour plus d'informations, consultez les actions ayant Addon dans leur nom dans Actions définies par Amazon Elastic Kubernetes Service.
- Les modules complémentaires Amazon EKS s'exécutent sur les nœuds que vous approvisionnez ou configurez pour votre cluster. Les types de nœuds incluent EC2 les instances Amazon, Fargate et les nœuds hybrides.
- Vous pouvez modifier les champs qui ne sont pas gérés par Amazon EKS pour personnaliser l'installation d'un module complémentaire Amazon EKS. Pour de plus amples informations, veuillez consulter the section called "Champs que vous pouvez personnaliser".
- Si vous créez un cluster avec les modules AWS Management Console complémentaires Amazon EKS`kube-proxy`, Amazon VPC CNI pour Kubernetes et CoreDNS Amazon EKS sont automatiquement ajoutés à votre cluster. Si vous utilisez `eksctl` pour créer votre cluster avec un fichier `config`, `eksctl` peut également créer le cluster avec les modules complémentaires Amazon EKS. Si vous créez votre cluster avec `eksctl` sans fichier `config` ou à l'aide d'un autre outil, les modules complémentaires auto-gérés `kube-proxy`, Amazon VPC CNI plug-in pour Kubernetes et CoreDNS sont installés à la place des modules complémentaires Amazon EKS. Vous pouvez les gérer vous-même ou ajouter manuellement les modules complémentaires Amazon EKS après la création du cluster. Quelle que soit la méthode utilisée pour créer votre cluster, le module complémentaire VPC CNI ne s'installe pas sur les nœuds hybrides.
- La ressource `eks:addon-cluster-admin` `ClusterRoleBinding` lie la ressource `cluster-admin` `ClusterRole` à l'identité Kubernetes `eks:addon-manager`. Le rôle dispose des autorisations nécessaires pour que l'identité `eks:addon-manager` puisse créer des espaces de noms Kubernetes et installer des modules complémentaires dans les espaces de noms. Si la ressource `ClusterRoleBinding` `eks:addon-cluster-admin` est supprimée, le cluster Amazon EKS continuera de fonctionner, mais Amazon EKS ne sera plus en mesure de gérer les modules complémentaires. Tous les clusters qui démarrent avec les versions de plateforme suivantes utilisent la nouvelle ressource `ClusterRoleBinding`.

- La compatibilité d'un sous-ensemble de AWS modules complémentaires EKS avec Amazon EKS Hybrid Nodes a été validée. Pour plus d'informations, consultez le tableau de compatibilité sur the section called " AWS modules complémentaires".

Espace de noms personnalisé pour les modules complémentaires

Pour la communauté et les AWS modules complémentaires, vous pouvez éventuellement spécifier un espace de noms personnalisé lors de la création du module complémentaire. Une fois que vous avez installé un module complémentaire dans un espace de noms spécifique, vous devez le supprimer et le recréer pour modifier son espace de noms.

Si vous ne spécifiez pas d'espace de noms, l'espace de noms prédéfini pour le module complémentaire sera utilisé.

Utilisez des espaces de noms personnalisés pour mieux organiser et isoler les objets complémentaires au sein de votre cluster EKS. Cette flexibilité vous aide à aligner les modules complémentaires sur vos besoins opérationnels et votre stratégie existante en matière d'espace de noms.

Vous pouvez définir un espace de noms personnalisé lors de la création d'un module complémentaire. Pour de plus amples informations, veuillez consulter the section called "Créer un module complémentaire".

Obtenir un espace de noms prédéfini pour le module complémentaire

L'espace de noms prédéfini pour un module complémentaire est l'espace de noms dans lequel il sera installé si vous n'en spécifiez pas.

Pour obtenir l'espace de noms prédéfini d'un module complémentaire, utilisez la commande suivante :

```
aws eks describe-addon-versions --addon-name <addon-name> --query
 "addons[].defaultNamespace"
```

Exemple de sortie :

```
[
    "kube-system"
]
```

Considérations relatives au mode automatique Amazon EKS

Le mode automatique Amazon EKS comprend des fonctionnalités qui offrent des fonctionnalités essentielles pour les clusters, notamment :

- Mise en réseau des pods
- Réseau de services
- DNS de cluster
- Autoscaling
- Stockage en mode bloc
- Contrôleur d’équilibreur de charge
- Agent d’identité du pod
- Agent de surveillance des nœuds

Avec le calcul en mode automatique, de nombreux modules complémentaires EKS couramment utilisés deviennent redondants, tels que :

- CNI Amazon VPC
- kube-proxy
- CoreDNS
- Pilote CSI Amazon EBS
- Agent d'identité du pod EKS

Toutefois, si votre cluster combine le mode automatique avec d'autres options de calcul telles que les EC2 instances autogérées, les groupes de nœuds gérés ou AWS Fargate, ces modules complémentaires restent nécessaires. AWS a amélioré les modules complémentaires EKS avec des règles anti-affinité qui garantissent automatiquement que les modules complémentaires sont planifiés uniquement sur les types de calcul pris en charge. De plus, les utilisateurs peuvent désormais exploiter l’API des modules complémentaires EKS `DescribeAddonVersions` pour vérifier les types de calcul pris en charge pour chaque module complémentaire et ses versions spécifiques. De plus, avec le mode EKS Auto, les contrôleurs répertoriés ci-dessus fonctionnent sur AWS une infrastructure propre. Ainsi, vous ne les verrez peut-être même pas apparaître dans vos comptes, sauf si vous utilisez le mode automatique EKS avec d’autres types de calcul, auquel cas vous verrez les contrôleurs que vous avez installés sur votre cluster.

Si vous prévoyez d'activer le mode automatique EKS sur un cluster existant, vous devrez peut-être mettre à niveau la version de certains modules complémentaires. Pour plus d'informations, consultez the section called "Versions requises des modules complémentaires" pour le mode automatique EKS.

Support

AWS publie plusieurs types de modules complémentaires avec différents niveaux de support.

- Modules complémentaires AWS : Ces modules complémentaires sont conçus et entièrement pris en charge par AWS.
  - Utilisez un AWS module complémentaire pour travailler avec d'autres AWS services, tels qu'Amazon EFS.
  - Pour de plus amples informations, veuillez consulter the section called " AWS modules complémentaires".
- AWS Modules complémentaires Marketplace : ces modules complémentaires sont analysés AWS et pris en charge par un AWS partenaire indépendant.
  - Utilisez un module complémentaire du Marketplace pour ajouter des fonctionnalités utiles et sophistiquées à votre cluster, telles que la surveillance avec Splunk.
  - Pour de plus amples informations, veuillez consulter the section called "Modules complémentaires Marketplace".
- Modules complémentaires communautaires : Ces modules complémentaires sont analysés par la communauté open source AWS mais pris en charge par celle-ci.
  - Utilisez un module complémentaire communautaire pour réduire la complexité de l'installation de logiciels open source courants, tels que Kubernetes Metrics Server.
  - Les modules complémentaires communautaires sont fournis à partir du code source par AWS. AWS valide uniquement les modules complémentaires communautaires pour la compatibilité des versions.
  - Pour de plus amples informations, veuillez consulter the section called "Extensions communautaires".

Le tableau suivant détaille l'étendue de la prise en charge pour chaque type de module complémentaire :

Catégorie	Fonctionnalité	AWS modules complémentaires	AWS Modules complémentaires Marketplace	Modules complémentaires communautaires
Développement	Construit par AWS	Oui	Non	Oui
Développement	Validé par AWS	Oui	Non	Oui*
Développement	Validé par le AWS partenaire	Non	Oui	Non
Maintenance	Scanné par AWS	Oui	Oui	Oui
Maintenance	Corrigé par AWS	Oui	Non	Oui
Maintenance	Corrigé par un partenaire AWS	Non	Oui	Non
Distribution	Publié par AWS	Oui	Non	Oui
Distribution	Publié par AWS Partner	Non	Oui	Non
Support	Support d'installation de base par AWS	Oui	Oui	Oui
Support	AWS Support complet	Oui	Non	Non
Support	Support complet AWS pour les partenaires	Non	Oui	Non

* : La validation des modules complémentaires communautaires inclut uniquement la compatibilité des versions de Kubernetes. Par exemple, si vous installez un module complémentaire communautaire sur un cluster, vérifiez AWS s'il est compatible avec la version Kubernetes de votre cluster.

AWS Les modules complémentaires Marketplace peuvent télécharger des dépendances logicielles supplémentaires à partir de sources externes extérieures à AWS. Ces dépendances externes ne sont ni scannées ni validées par AWS. Tenez compte de vos exigences en matière de sécurité lorsque vous déployez des modules complémentaires AWS Marketplace qui récupèrent des dépendances externes.

AWS modules complémentaires

Les modules complémentaires Amazon EKS suivants peuvent être créés sur votre cluster. Vous pouvez consulter la liste la plus récente des modules complémentaires disponibles à l'aide `eksctl` de la AWS Management Console CLI ou de la AWS CLI. Pour voir tous les modules complémentaires disponibles ou pour installer un module complémentaire, consultez the section called "Créer un module complémentaire". Si un module complémentaire nécessite des autorisations IAM, vous devez disposer d'un fournisseur IAM OpenID Connect (OIDC) pour votre cluster. Pour déterminer si vous en avez un, ou pour en créer un, consultez the section called "Fournisseur IAM OIDC". Vous pouvez créer ou supprimer un module complémentaire après l'avoir installé. Pour plus d'informations, consultez the section called "Mettre à jour un module complémentaire" ou the section called "Supprimer un module complémentaire". Pour plus d'informations sur les considérations spécifiques à l'exécution des modules complémentaires EKS avec des nœuds hybrides Amazon EKS, consultez the section called "Configurer les modules complémentaires".

Vous pouvez utiliser l'un des modules complémentaires Amazon EKS suivants.

Description	En savoir plus	Types de calcul compatibles
Fournir un réseau VPC natif pour votre cluster	the section called "Plug-in CNI Amazon VPC pour Kubernetes"	EC2
Un serveur DNS flexible et extensible qui peut servir de DNS pour le cluster Kubernetes	the section called "CoreDNS"	EC2, Fargate, mode automatique EKS, nœuds hybrides EKS
Maintenir les règles réseau sur chaque nœud Amazon EC2	the section called "`Kube-proxy`"	EC2, nœuds hybrides EKS

Description	En savoir plus	Types de calcul compatibles
Fournir un stockage Amazon EBS pour votre cluster	the section called "Pilote CSI Amazon EBS"	EC2
Fournir un stockage Amazon EFS pour votre cluster	the section called "Pilote CSI Amazon EFS"	EC2, mode automatique EKS
Fournissez un espace de stockage Amazon FSx for Lustre pour votre cluster	the section called "pilote Amazon FSx CSI"	EC2, mode automatique EKS
Fournir un stockage Amazon S3 pour votre cluster	the section called "Pilote CSI Mountpoint pour Amazon S3"	EC2, mode automatique EKS
Détecter les problèmes supplémentaires d'état des nœuds	the section called "Agent de surveillance des nœuds"	EC2, nœuds hybrides EKS
Activer l'utilisation de la fonctionnalité d'instantané dans les pilotes CSI compatibles, tels que le pilote Amazon EBS CSI	the section called "Contrôleur d'instantané CSI"	EC2, Fargate, mode automatique EKS, nœuds hybrides EKS
SageMaker HyperPod la gouvernance des tâches optimise l'allocation et l'utilisation des ressources informatiques entre les équipes dans les clusters Amazon EKS, remédiant ainsi aux inefficiences en matière de priorisation des tâches et de partage des ressources.	the section called "Gouvernance des SageMaker HyperPod tâches Amazon"	EC2, mode automatique EKS,

Description	En savoir plus	Types de calcul compatibles
Amazon SageMaker HyperPod Observability AddOn fournit des fonctionn alités complètes de surveilla nce et d'observabilité pour HyperPod les clusters.	the section called "Module complémentaire Amazon SageMaker HyperPod Observability"	EC2, mode automatique EKS,
Amazon SageMaker HyperPod Training Operator permet une formation distribué e efficace sur les clusters Amazon EKS grâce à des fonctionnalités avancées de planification et de gestion des ressources.	the section called "Opérateu r SageMaker HyperPod de formation Amazon"	EC2, mode automatique EKS
Amazon SageMaker HyperPod Inference Operator permet le déploiement et la gestion de charges de travail d'inférence basées sur l'IA à hautes performances avec une utilisation des ressources et une rentabilité optimisées.	the section called "Opérateu r SageMaker HyperPod d'inférence Amazon"	EC2, mode automatique EKS
Un agent Kubernetes qui collecte et transmet les données de flux réseau à Amazon CloudWatch, permettant ainsi une surveilla nce complète des connexion s TCP entre les nœuds du cluster.	the section called "AWS Agent de surveillance du flux réseau"	EC2, mode automatique EKS

Description	En savoir plus	Types de calcul compatibles
Distribution sécurisée, prête pour la production et AWS prise en charge du projet OpenTelemetry	the section called "AWS Distro pour OpenTelemetry"	EC2, Fargate, mode automatique EKS, nœuds hybrides EKS
Service de surveillance de la sécurité qui analyse et traite les sources de données de base, notamment les événements AWS CloudTrail de gestion et les journaux de flux Amazon VPC. Amazon traite GuardDuty également des fonctionnalités, telles que les journaux d'audit Kubernetes et la surveillance du temps d'exécution	the section called "GuardDuty Agent Amazon"	EC2, mode automatique EKS
Service de surveillance et d'observabilité fourni par AWS. Ce module complémentaire installe l' CloudWatch agent et active à la fois CloudWatch Application Signals et CloudWatch Container Insights avec une observabilité améliorée pour Amazon EKS	the section called "Agent Amazon CloudWatch Observability"	EC2, mode automatique EKS, nœuds hybrides EKS

Description	En savoir plus	Types de calcul compatibles
Possibilité de gérer les informations d'identification de vos applications, de la même manière que les profils d'instance EC2 fournissent des informations d'identification aux instances EC2	the section called "Agent d'identité du pod EKS"	EC2, nœuds hybrides EKS
Activez cert-manager pour émettre des certificats X.509 à partir de AWS Private CA. Nécessite cert-manager.	the section called "AWS Connecteur CA privé pour Kubernetes"	EC2, Fargate, mode automatique EKS, nœuds hybrides EKS
Génère des métriques Prometheus sur les performances des périphériques réseau SR-IOV	the section called "Exportateur de métriques réseau SR-IOV"	EC2
Récupérez des AWS secrets dans Secrets Manager et des paramètres dans le magasin de paramètres de AWS Systems Manager et montez-les sous forme de fichiers dans des pods Kubernetes.	the section called "AWS Fournisseur de pilotes CSI Secrets Store"	EC2, mode automatique EKS, nœuds hybrides EKS
Avec Spaces, vous pouvez créer JupyterLab et gérer des applications d'éditeur de code pour exécuter des charges de travail ML interactives.	the section called "Amazon SageMaker Spaces"	Hyperpode

Plug-in CNI Amazon VPC pour Kubernetes

Le module complémentaire CNI VPC pour Kubernetes Amazon EKS est un module complémentaire d'interface réseau de conteneur (CNI) Kubernetes qui fournit une mise en réseau VPC native pour votre cluster. Le type autogéré ou géré de ce module complémentaire est installé sur chaque nœud Amazon EC2, par défaut. Pour plus d'informations, consultez Module complémentaire d'interface réseau de conteneur (CNI) Kubernetes.

> **Note**
>
> Vous n'avez pas besoin d'installer ce module complémentaire sur les clusters du mode automatique Amazon EKS. Pour de plus amples informations, veuillez consulter the section called "Considérations relatives au mode automatique Amazon EKS".

Le nom du module complémentaire Amazon EKS est `vpc-cni`.

Autorisations IAM requises

Ce module complémentaire utilise la fonctionnalité de rôles IAM pour les comptes de service d'Amazon EKS. Pour de plus amples informations, veuillez consulter the section called "Informations d'identification auprès de l'IRSA".

Si votre cluster utilise la famille `IPv4`, les autorisations définies dans la politique AmazonEKS_CNI_Policy sont requises. Si votre cluster utilise cette `IPv6` famille, vous devez créer une politique IAM avec les autorisations en IPv6 mode. Vous pouvez créer un rôle IAM, y associer l'une des politiques et annoter le compte de service Kubernetes utilisé par le module complémentaire à l'aide de la commande suivante.

Remplacez *`my-cluster`* par le nom de votre cluster et *`AmazonEKSVPCCNIRole`* par le nom que vous souhaitez pour votre rôle. Si votre cluster utilise la famille `IPv6`, remplacez *`AmazonEKS_CNI_Policy`* par le nom de la politique que vous avez créée. Cette commande nécessite que eksctl soit installé sur votre appareil. Si vous devez utiliser un autre outil pour créer le rôle, y associer la politique et annoter le compte de service Kubernetes, consultez the section called "Attribution d'un rôle IAM".

```
eksctl create iamserviceaccount --name aws-node --namespace kube-system --cluster my-
cluster --role-name AmazonEKSVPCCNIRole \
```

```
    --role-only --attach-policy-arn arn:aws: iam::aws:policy/AmazonEKS_CNI_Policy --
approve
```

Mise à jour des informations

Vous ne pouvez mettre à jour qu'une seule version mineure à la fois. Par exemple, si votre version actuelle est `1.28.x-eksbuild.y` et que vous voulez la mettre à jour vers `1.30.x-eksbuild.y`, vous devez d'abord mettre à jour votre version actuelle vers `1.29.x-eksbuild.y` et ensuite la mettre à nouveau à jour vers `1.30.x-eksbuild.y`. Pour plus d'informations sur la mise à jour du module complémentaire, consultez la rubrique the section called “Mise à jour (module complémentaire EKS)”.

CoreDNS

Le module complémentaire CoreDNS Amazon EKS est un serveur DNS flexible et extensible qui peut servir de DNS pour le cluster Kubernetes. Le type autogéré ou géré de ce module complémentaire a été installé par défaut lorsque vous avez créé votre cluster. Lorsque vous lancez un cluster Amazon EKS avec au moins un nœud, deux réplicas de l'image CoreDNS sont déployés par défaut, quel que soit le nombre de nœuds déployés dans votre cluster. Les pods CoreDNS fournissent une résolution de noms pour tous les pods du cluster. Vous pouvez déployer les pod CoreDNS sur les nœuds Fargate si votre cluster comprend un profil Fargate avec un espace de noms qui correspond à l’espace de noms du déploiement CoreDNS. Pour de plus amples informations, consultez the section called “Définir des profils”.

> **Note**
>
> Vous n’avez pas besoin d’installer ce module complémentaire sur les clusters du mode automatique Amazon EKS. Pour de plus amples informations, veuillez consulter the section called “Considérations relatives au mode automatique Amazon EKS”.

Le nom du module complémentaire Amazon EKS est `coredns`.

Autorisations IAM requises

Ce module complémentaire ne nécessite aucune autorisation.

Informations supplémentaires

Pour en savoir plus sur CoreDNS, consultez Utilisation de CoreDNS pour la découverte de services et Personnalisation du service DNS dans la documentation Kubernetes.

`Kube-proxy`

Le module complémentaire `Kube-proxy` Amazon EKS gère les règles réseau sur chaque nœud Amazon EC2. Il permet la communication réseau avec vos pods. Le type autogéré ou géré de ce module complémentaire est installé par défaut sur chaque nœud Amazon EC2 de votre cluster.

> **Note**
>
> Vous n'avez pas besoin d'installer ce module complémentaire sur les clusters du mode automatique Amazon EKS. Pour de plus amples informations, veuillez consulter the section called "Considérations relatives au mode automatique Amazon EKS".

Le nom du module complémentaire Amazon EKS est `kube-proxy`.

Autorisations IAM requises

Ce module complémentaire ne nécessite aucune autorisation.

Mise à jour des informations

Avant de mettre à jour votre version actuelle, veuillez tenir compte des exigences suivantes :

- `Kube-proxy` sur un cluster Amazon EKS a la même politique de compatibilité et de décalage que Kubernetes.

Informations supplémentaires

Pour en savoir plus sur `kube-proxy`, consultez kube-proxy dans la documentation Kubernetes.

Pilote CSI Amazon EBS

Le module complémentaire Amazon EKS, le pilote CSI Amazon EBS, est un module complémentaire Kubernetes Container Storage Interface (CSI) qui fournit un stockage Amazon EBS pour votre cluster.

> **Note**
>
> Vous n'avez pas besoin d'installer ce module complémentaire sur les clusters du mode automatique Amazon EKS. Le mode automatique inclut une fonctionnalité de stockage en bloc. Pour de plus amples informations, veuillez consulter the section called "Déployer une charge de travail avec état".

Le nom du module complémentaire Amazon EKS est `aws-ebs-csi-driver`.

Autorisations IAM requises

Ce module complémentaire utilise les rôles IAM pour la fonctionnalité des comptes de service d'Amazon EKS. Pour de plus amples informations, veuillez consulter the section called "Informations d'identification auprès de l'IRSA". Les autorisations définies dans la EBSCSIDriverpolitique AWS gérée par Amazon Policy sont obligatoires. Vous pouvez créer un rôle IAM et y attacher la politique gérée à l'aide de la commande suivante. Remplacez *`my-cluster`* par le nom de votre cluster et *`AmazonEKS_EBS_CSI_DriverRole`* par le nom que vous souhaitez pour votre rôle. Cette commande nécessite que eksctl soit installé sur votre appareil. Si vous devez utiliser un autre outil ou une clé KMS personnalisée pour le chiffrement, reportez-vous à la section the section called "Étape 1 : Créer un rôle IAM".

```
eksctl create iamserviceaccount \
    --name ebs-csi-controller-sa \
    --namespace kube-system \
    --cluster my-cluster \
    --role-name AmazonEKS_EBS_CSI_DriverRole \
    --role-only \
    --attach-policy-arn arn:aws: iam::aws:policy/service-role/AmazonEBSCSIDriverPolicy
 \
    --approve
```

Informations supplémentaires

Pour plus d'informations sur le module complémentaire, consultez the section called "Amazon EBS".

Pilote CSI Amazon EFS

Le module complémentaire Amazon EKS, le pilote CSI Amazon EFS, est un plug-in Kubernetes Container Storage Interface (CSI) qui fournit un stockage Amazon EFS pour votre cluster.

Le nom du module complémentaire Amazon EKS est `aws-efs-csi-driver`.

Autorisations IAM requises

Autorisations IAM requises : ce module complémentaire utilise les rôles IAM pour la fonctionnalité des comptes de service d'Amazon EKS. Pour de plus amples informations, veuillez consulter the section called "Informations d'identification auprès de l'IRSA". Les autorisations définies dans la EFSCSIDriverpolitique AWS gérée par Amazon Policy sont obligatoires. Vous pouvez créer un rôle IAM et y attacher la politique gérée à l'aide des commandes suivantes. Remplacez *`my-cluster`* par le nom de votre cluster et *`AmazonEKS_EFS_CSI_DriverRole`* par le nom que vous souhaitez pour votre rôle. Ces commandes nécessitent que eksctl soit installé sur votre appareil. Si vous devez utiliser un autre outil, voir the section called "Étape 1 : Créer un rôle IAM".

```
export cluster_name=my-cluster
export role_name=AmazonEKS_EFS_CSI_DriverRole
eksctl create iamserviceaccount \
    --name efs-csi-controller-sa \
    --namespace kube-system \
    --cluster $cluster_name \
    --role-name $role_name \
    --role-only \
    --attach-policy-arn arn:aws: iam::aws:policy/service-role/AmazonEFSCSIDriverPolicy
 \
    --approve
TRUST_POLICY=$(aws iam get-role --output json --role-name $role_name --query
 'Role.AssumeRolePolicyDocument' | \
    sed -e 's/efs-csi-controller-sa/efs-csi-*/' -e 's/StringEquals/StringLike/')
aws iam update-assume-role-policy --role-name $role_name --policy-document
 "$TRUST_POLICY"
```

Informations supplémentaires

Pour plus d'informations sur le module complémentaire, consultez the section called "Amazon EFS".

pilote Amazon FSx CSI

Le module complémentaire Amazon EKS du pilote Amazon FSx CSI est un plugin Kubernetes Container Storage Interface (CSI) qui fournit le stockage Amazon FSx for Lustre pour votre cluster.

Le nom du module complémentaire Amazon EKS est `aws-fsx-csi-driver`.

Note

- Les installations préexistantes du pilote Amazon FSx CSI dans le cluster peuvent provoquer l'échec de l'installation des modules complémentaires. Lorsque vous tentez d'installer la version complémentaire Amazon EKS alors qu'un pilote FSx CSI autre qu'EKS existe, l'installation échoue en raison de conflits de ressources. Utilisez l'indicateur `OVERWRITE` lors de l'installation pour résoudre ce problème :

```
aws eks create-addon --addon-name aws-fsx-csi-driver --cluster-name my-cluster
 --resolve-conflicts OVERWRITE
```

- Le module complémentaire Amazon FSx CSI Driver EKS nécessite l'agent EKS Pod Identity pour l'authentification. Sans ce composant, le module complémentaire échouera avec l'erreur `Amazon EKS Pod Identity agent is not installed in the cluster`, empêchant les opérations sur le volume. Installez l'agent Pod Identity avant ou après le déploiement du module complémentaire FSx CSI Driver. Pour de plus amples informations, veuillez consulter the section called "Configurer l'agent".

Autorisations IAM requises

Ce module complémentaire utilise les rôles IAM pour la fonctionnalité des comptes de service d'Amazon EKS. Pour de plus amples informations, veuillez consulter the section called "Informations d'identification auprès de l'IRSA". Les autorisations définies dans la politique FSx FullAccess AWS gérée par Amazon sont obligatoires. Vous pouvez créer un rôle IAM et y attacher la politique gérée à l'aide de la commande suivante. Remplacez *`my-cluster`* par le nom de votre cluster et *`AmazonEKS_FSx_CSI_DriverRole`* par le nom que vous souhaitez pour votre rôle. Cette commande nécessite que eksctl soit installé sur votre appareil.

```
eksctl create iamserviceaccount \
    --name fsx-csi-controller-sa \
    --namespace kube-system \
    --cluster my-cluster \
    --role-name AmazonEKS_FSx_CSI_DriverRole \
    --role-only \
    --attach-policy-arn arn:aws: iam::aws:policy/AmazonFSxFullAccess \
    --approve
```

Informations supplémentaires

Pour plus d'informations sur le module complémentaire, consultez the section called "Amazon FSx pour Lustre".

Pilote CSI Mountpoint pour Amazon S3

Le module complémentaire Amazon EKS, le pilote CSI Mountpoint pour Amazon S3, est un module complémentaire Kubernetes Container Storage Interface (CSI) qui fournit un stockage Amazon S3 pour votre cluster.

Le nom du module complémentaire Amazon EKS est `aws-mountpoint-s3-csi-driver`.

Autorisations IAM requises

Ce module complémentaire utilise la fonctionnalité de rôles IAM pour les comptes de service d'Amazon EKS. Pour de plus amples informations, veuillez consulter the section called "Informations d'identification auprès de l'IRSA".

Le rôle IAM créé nécessitera une stratégie d'accès à S3. Veuillez suivre les Recommandations relatives aux autorisations IAM pour Mountpoint lors de la création de la politique. Vous pouvez également utiliser la politique AWS gérée AmazonS3 FullAccess, mais cette politique gérée accorde plus d'autorisations que ce qui est nécessaire pour Mountpoint.

Vous pouvez créer un rôle IAM et y attacher votre politique à l'aide des commandes suivantes. Remplacez-le *`my-cluster`* par le nom de votre cluster, *`region-code`* par le code de AWS région correct, *`AmazonEKS_S3_CSI_DriverRole`* par le nom de votre rôle et *`AmazonEKS_S3_CSI_DriverRole_ARN`* par l'ARN du rôle. Ces commandes nécessitent que eksctl soit installé sur votre appareil. Pour obtenir des instructions sur l'utilisation de la console ou de la AWS CLI IAM, consultezthe section called "Étape 2 : Créer un rôle IAM".

```
CLUSTER_NAME=my-cluster
REGION=region-code
ROLE_NAME=AmazonEKS_S3_CSI_DriverRole
POLICY_ARN=AmazonEKS_S3_CSI_DriverRole_ARN
eksctl create iamserviceaccount \
    --name s3-csi-driver-sa \
    --namespace kube-system \
    --cluster $CLUSTER_NAME \
    --attach-policy-arn $POLICY_ARN \
    --approve \
```

```
    --role-name $ROLE_NAME \
    --region $REGION \
    --role-only
```

Informations supplémentaires

Pour plus d'informations sur le module complémentaire, consultez the section called "Mountpoint pour Amazon S3".

Contrôleur d'instantané CSI

Le contrôleur d'instantanés Container Storage Interface (CSI) permet d'utiliser la fonctionnalité d'instantanés dans les pilotes CSI compatibles, tels que le pilote CSI Amazon EBS.

Le nom du module complémentaire Amazon EKS est `snapshot-controller`.

Autorisations IAM requises

Ce module complémentaire ne nécessite aucune autorisation.

Informations supplémentaires

Pour plus d'informations sur le module complémentaire, consultez the section called "Contrôleur d'instantané CSI".

Gouvernance des SageMaker HyperPod tâches Amazon

SageMaker HyperPod la gouvernance des tâches est un système de gestion robuste conçu pour rationaliser l'allocation des ressources et garantir une utilisation efficace des ressources informatiques au sein des équipes et des projets pour vos clusters Amazon EKS. Cela permet aux administrateurs de définir :

- des niveaux de priorité pour différentes tâches ;
- l'allocation de ressources de calcul pour chaque équipe ;
- comment chaque équipe prête et emprunte des ressources de calcul inactives ;
- si une équipe préempte ses propres tâches.

HyperPod la gouvernance des tâches fournit également l'observabilité du cluster Amazon EKS, offrant une visibilité en temps réel sur la capacité du cluster. Cela inclut la disponibilité et l'utilisation des ressources informatiques, l'allocation et l'utilisation par équipe, ainsi que des informations

sur l'exécution des tâches et les temps d'attente, vous permettant ainsi de prendre des décisions éclairées et de gérer vos ressources de manière proactive.

Le nom du module complémentaire Amazon EKS est `amazon-sagemaker-hyperpod-taskgovernance`.

Autorisations IAM requises

Ce module complémentaire ne nécessite aucune autorisation.

Informations supplémentaires

Pour en savoir plus sur le module complémentaire, consultez la section Gouvernance des SageMaker HyperPod tâches

Module complémentaire Amazon SageMaker HyperPod Observability

Le module complémentaire Amazon SageMaker HyperPod Observability fournit des fonctionnalités complètes de surveillance et d'observabilité pour HyperPod les clusters. Ce module complémentaire déploie et gère automatiquement les composants de surveillance essentiels, notamment l'exportateur de nœuds, l'exportateur DCGM et l'exportateur EFA. kube-state-metrics Il collecte et transmet les métriques à une instance Amazon Managed Prometheus (AMP) désignée par la clientèle et expose un point de terminaison OTLP pour les métriques personnalisées et l'ingestion d'événements provenant des tâches d'entraînement de la clientèle.

Le module complémentaire s'intègre à l'ensemble de l' HyperPod écosystème en extrayant les métriques de divers composants, notamment le module complémentaire HyperPod Task Governance, HyperPod Training Operator, Kubeflow et KEDA. Toutes les métriques collectées sont centralisées dans Amazon Managed Prometheus, ce qui permet aux clients d'obtenir une vue d'observabilité unifiée via les tableaux de bord Amazon Managed Grafana. Cela fournit une end-to-end visibilité sur l'état du cluster, l'utilisation des ressources et les performances des tâches de formation dans l'ensemble de HyperPod l'environnement.

Le nom du module complémentaire Amazon EKS est `amazon-sagemaker-hyperpod-observability`.

Autorisations IAM requises

Ce module complémentaire utilise la fonctionnalité de rôles IAM pour les comptes de service d'Amazon EKS. Pour de plus amples informations, veuillez consulter the section called "Informations d'identification auprès de l'IRSA". Les politiques gérées suivantes sont requises :

- `AmazonPrometheusRemoteWriteAccess` : pour l'écriture à distance des métriques du cluster vers AMP
- `CloudWatchAgentServerPolicy`- pour écrire à distance les journaux du cluster vers CloudWatch

Informations supplémentaires

Pour en savoir plus sur le module complémentaire et ses fonctionnalités, consultez SageMaker HyperPod Observabilité.

Opérateur SageMaker HyperPod de formation Amazon

L'opérateur de SageMaker HyperPod formation Amazon vous aide à accélérer le développement de modèles d'IA générative en gérant efficacement la formation distribuée sur de grands clusters de GPU. Il intègre des fonctionnalités intelligentes de restauration après panne, de détection des interruptions de tâche et de gestion au niveau des processus qui minimisent les interruptions d'entraînement et réduisent les coûts. Contrairement à l'infrastructure d'entraînement traditionnelle qui nécessite le redémarrage complet des tâches en cas de défaillance, cet opérateur met en œuvre la récupération chirurgicale des processus pour maintenir le bon déroulement de vos tâches d'entraînement.

L'opérateur utilise également les fonctions de surveillance HyperPod de l'état de santé et d'observabilité de l'opérateur, qui fournissent une visibilité en temps réel sur l'exécution de la formation et une surveillance automatique des indicateurs critiques tels que les pics de pertes et la dégradation du débit. Vous pouvez définir des politiques de récupération par le biais de configurations YAML simples sans modifier le code, ce qui vous permet de réagir rapidement et de récupérer à partir d'états d'entraînement irrécupérables. Ces capacités de surveillance et de récupération fonctionnent ensemble pour maintenir des performances de formation optimales tout en minimisant les frais généraux opérationnels.

Le nom du module complémentaire Amazon EKS est `amazon-sagemaker-hyperpod-training-operator`.

Pour plus d'informations, consultez la section Utilisation de l'opérateur de HyperPod formation dans le manuel Amazon SageMaker Developer Guide.

Autorisations IAM requises

Ce module complémentaire nécessite des autorisations IAM et utilise l'identité du pod Amazon EKS.

AWS suggère la politique `AmazonSageMakerHyperPodTrainingOperatorAccess` gérée.

Pour plus d'informations, consultez la section Installation de l'opérateur de formation dans le manuel Amazon SageMaker Developer Guide.

Informations supplémentaires

Pour en savoir plus sur le module complémentaire, consultez la section SageMaker HyperPod Formation des opérateurs.

Opérateur SageMaker HyperPod d'inférence Amazon

Amazon SageMaker HyperPod propose une end-to-end expérience qui prend en charge le cycle de vie complet du développement de l'IA, de l'expérimentation interactive à la formation, en passant par les flux de travail d'inférence et de post-formation. Elle fournit désormais une plateforme d'inférence complète qui associe la flexibilité de Kubernetes à l'excellence opérationnelle d'une expérience gérée. Déployez, dimensionnez et optimisez vos modèles GenAI avec une fiabilité de niveau professionnel en utilisant le même HyperPod calcul tout au long du cycle de vie du modèle.

Amazon SageMaker HyperPod propose des interfaces de déploiement flexibles qui vous permettent de déployer des modèles par le biais de plusieurs méthodes, notamment kubectl, le SDK Python, l'interface utilisateur Amazon SageMaker Studio ou la CLI. HyperPod Cette fonctionnalité fournit des fonctionnalités avancées de mise à l'échelle automatique avec une allocation dynamique des ressources qui s'ajuste automatiquement en fonction de la demande. En outre, il inclut des fonctionnalités complètes d'observabilité et de surveillance qui suivent les indicateurs critiques tels que time-to-first-token la latence et l'utilisation du GPU pour vous aider à optimiser les performances.

Le nom du module complémentaire Amazon EKS est `amazon-sagemaker-hyperpod-inference`.

Méthodes d'installation

Vous pouvez installer ce module complémentaire en utilisant l'une des méthodes suivantes :

- SageMaker Console (recommandée) : fournit une expérience d'installation rationalisée grâce à une configuration guidée.
- Console ou CLI EKS Add-ons : nécessite l'installation manuelle des modules complémentaires de dépendance avant d'installer l'opérateur d'inférence. Consultez la section sur les prérequis ci-dessous pour connaître les dépendances requises.

Conditions préalables

Avant d'installer le module complémentaire d'opérateur d'inférence via la console EKS Add-ons ou la CLI, assurez-vous que les dépendances suivantes sont installées.

Modules complémentaires EKS requis :

- Pilote CSI Amazon S3 Mountpoint (version minimale : v1.14.1-eksbuild.1)
- Serveur de métriques (version minimale : v0.7.2-eksbuild.4)
- Pilote Amazon FSx CSI (version minimale : v1.6.0-eksbuild.1)
- Gestionnaire de certificats (version minimale : v1.18.2-eksbuild.2)

Pour obtenir des instructions d'installation détaillées pour chaque dépendance, voir Installation de l'opérateur d'inférence.

Autorisations IAM requises

Ce module complémentaire nécessite des autorisations IAM et utilise OIDC/IRSA.

Les politiques gérées suivantes sont recommandées car elles fournissent les autorisations limitées minimales :

- `AmazonSageMakerHyperPodInferenceAccess`- fournit les privilèges d'administrateur requis pour configurer l'opérateur d'inférence
- `AmazonSageMakerHyperPodGatedModelAccess`- donne SageMaker HyperPod accès à des modèles sécurisés dans SageMaker Jumpstart (par exemple, Meta Llama, GPT-Neo)

Pour plus d'informations, consultez la section Installation de l'opérateur d'inférence.

Informations supplémentaires

Pour en savoir plus sur l'opérateur d' SageMaker HyperPod inférence Amazon, consultez Opérateur SageMaker HyperPod d'inférence.

Pour obtenir des informations de dépannage, consultez la section Résolution des problèmes liés au déploiement du SageMaker HyperPod modèle.

AWS Agent de surveillance du flux réseau

L'agent Amazon CloudWatch Network Flow Monitor est une application Kubernetes qui collecte les statistiques de connexion TCP à partir de tous les nœuds d'un cluster et publie des rapports sur le flux réseau dans Amazon CloudWatch Network Flow Monitor Ingestion. APIs

Le nom du module complémentaire Amazon EKS est `aws-network-flow-monitoring-agent`.

Autorisations IAM requises

Ce module complémentaire nécessite des autorisations IAM.

Vous devez associer la politique gérée `CloudWatchNetworkFlowMonitorAgentPublishPolicy` au module complémentaire.

Pour plus d'informations sur la configuration IAM requise, consultez la politique IAM sur le référentiel d'agents GitHub Amazon CloudWatch Network Flow Monitor.

Pour plus d'informations sur la politique gérée, consultez CloudWatchNetworkFlowMonitorAgentPublishPolicyle guide de CloudWatch l'utilisateur Amazon.

Informations supplémentaires

Pour en savoir plus sur le module complémentaire, consultez le GitHub dépôt de l'agent Amazon CloudWatch Network Flow Monitor.

Agent de surveillance des nœuds

Le module complémentaire Amazon EKS de l'agent de surveillance des nœuds permet de détecter d'autres problèmes d'état des nœuds. Ces signaux d'état supplémentaires peuvent également être exploités par la fonctionnalité optionnelle de réparation automatique des nœuds afin de remplacer automatiquement les nœuds si nécessaire.

> **Note**
>
> Vous n'avez pas besoin d'installer ce module complémentaire sur les clusters du mode automatique Amazon EKS. Pour de plus amples informations, veuillez consulter the section called "Considérations relatives au mode automatique Amazon EKS".

Le nom du module complémentaire Amazon EKS est `eks-node-monitoring-agent`.

Autorisations IAM requises

Ce module complémentaire ne nécessite pas d'autorisations supplémentaires.

Informations supplémentaires

Pour de plus amples informations, veuillez consulter the section called "État du nœud".

AWS Distro pour OpenTelemetry

Le module complémentaire AWS Distro for OpenTelemetry Amazon EKS est une distribution sécurisée, prête pour la production et AWS prise en charge du projet. OpenTelemetry Pour plus d'informations, consultez AWS Distro for OpenTelemetry on GitHub.

Le nom du module complémentaire Amazon EKS est `adot`.

Autorisations IAM requises

Ce module complémentaire ne nécessite des autorisations IAM que si vous utilisez l'une des ressources personnalisées préconfigurées qui peuvent être activées via la configuration avancée.

Informations supplémentaires

Pour plus d'informations, consultez Getting Started with AWS Distro pour OpenTelemetry l'utilisation des modules complémentaires EKS dans la AWS distribution pour OpenTelemetry obtenir de la documentation.

ADOT exige que le `cert-manager` module complémentaire soit déployé sur le cluster comme condition préalable, sinon ce module complémentaire ne fonctionnera pas s'il est déployé directement à l'aide du https://registry.terraform. io/modules/terraform-aws-modules/eks/aws/ latest`cluster_addons`propriété. Pour plus d'informations sur les exigences, reportez-vous à la section Conditions requises pour démarrer avec AWS Distro pour OpenTelemetry l'utilisation des modules complémentaires EKS dans la AWS distribution pour OpenTelemetry obtenir de la documentation.

GuardDuty Agent Amazon

Le module complémentaire Amazon EKS de l' GuardDuty agent Amazon collecte les événements d'exécution (accès aux fichiers, exécution des processus, connexions réseau) à partir des nœuds de votre cluster EKS pour les analyser par GuardDuty Runtime Monitoring. GuardDuty lui-même (et non l'agent) est le service de surveillance de la sécurité qui analyse et traite les sources de données fondamentales, notamment les événements de AWS CloudTrail gestion et les journaux

de flux Amazon VPC, ainsi que des fonctionnalités telles que les journaux d'audit Kubernetes et la surveillance du temps d'exécution.

Le nom du module complémentaire Amazon EKS est `aws-guardduty-agent`.

Autorisations IAM requises

Ce module complémentaire ne nécessite aucune autorisation.

Informations supplémentaires

Pour plus d'informations, consultez la section Surveillance du temps d'exécution pour les clusters Amazon EKS sur Amazon GuardDuty.

- Pour détecter les menaces de sécurité potentielles dans vos clusters Amazon EKS, activez la surveillance de l' GuardDuty exécution Amazon et déployez l'agent de GuardDuty sécurité sur vos clusters Amazon EKS.

Agent Amazon CloudWatch Observability

L'agent Amazon CloudWatch Observability (Amazon EKS) ajoute le service de surveillance et d'observabilité fourni par. AWS Ce module complémentaire installe l' CloudWatch agent et active à la fois CloudWatch Application Signals et CloudWatch Container Insights avec une observabilité améliorée pour Amazon EKS. Pour plus d'informations, consultez Amazon CloudWatch Agent.

Le nom du module complémentaire Amazon EKS est `amazon-cloudwatch-observability`.

Autorisations IAM requises

Ce module complémentaire utilise la fonctionnalité de rôles IAM pour les comptes de service d'Amazon EKS. Pour de plus amples informations, veuillez consulter the section called "Informations d'identification auprès de l'IRSA". Les autorisations indiquées dans AWSXrayWriteOnlyAccessset dans les politiques CloudWatchAgentServerPolicy AWS gérées sont obligatoires. Vous pouvez créer un rôle IAM, y associer les politiques gérées et annoter le compte de service Kubernetes utilisé par le module complémentaire à l'aide de la commande suivante. Remplacez *`my-cluster`* par le nom de votre cluster et *`AmazonEKS_Observability_role`* par le nom que vous souhaitez pour votre rôle. Cette commande nécessite que eksctl soit installé sur votre appareil. Si vous devez utiliser un autre outil pour créer le rôle, y associer la politique et annoter le compte de service Kubernetes, consultez the section called "Attribution d'un rôle IAM".

```
eksctl create iamserviceaccount \
```

```
    --name cloudwatch-agent \
    --namespace amazon-cloudwatch \
    --cluster my-cluster \
    --role-name AmazonEKS_Observability_Role \
    --role-only \
    --attach-policy-arn arn:aws: iam::aws:policy/AWSXrayWriteOnlyAccess \
    --attach-policy-arn arn:aws: iam::aws:policy/CloudWatchAgentServerPolicy \
    --approve
```

Informations supplémentaires

Pour plus d'informations, consultez la section Installation de l' CloudWatch agent.

AWS Connecteur CA privé pour Kubernetes

Le connecteur AWS Private CA pour Kubernetes est un module complémentaire pour cert-manager qui permet aux utilisateurs d'obtenir des certificats auprès d'une autorité de certification privée (autorité de certification AWS privée).AWS

- Le nom du module complémentaire Amazon EKS est `aws-privateca-connector-for-kubernetes`.
- L'espace de noms du module complémentaire est `aws-privateca-issuer`.

Ce module complémentaire nécessite `cert-manager`. `cert-manager` est disponible sur Amazon EKS en tant que module complémentaire communautaire. Pour plus d'informations sur ce module complémentaire, consultez the section called "Gestionnaire de certificats". Pour plus d'informations sur l'installation des modules complémentaires, consultez the section called "Créer un module complémentaire".

Autorisations IAM requises

Ce module complémentaire nécessite des autorisations IAM.

Utilisez les identités du pod EKS pour associer la politique IAM `AWSPrivateCAConnectorForKubernetesPolicy` au compte de service Kubernetes `aws-privateca-issuer`. Pour de plus amples informations, veuillez consulter the section called "Utiliser les identités du pod".

Pour plus d'informations sur les autorisations requises, consultez le AWSPrivateCAConnectorForKubernetesPolicymanuel AWS Managed Policy Reference.

Informations supplémentaires

Pour plus d'informations, consultez le référentiel AWS Private CA Issuer for Kubernetes GitHub .

Pour plus d'informations sur la configuration du module complémentaire, consultez values.yaml dans le dépôt. `aws-privateca-issuer` GitHub Veuillez vérifier que la version de values.yaml correspond à la version du module complémentaire installé sur votre cluster.

Ce module complémentaire tolère les `CriticalAddonsOnly` taches produites par le mode automatique `system` NodePool d'EKS. Pour de plus amples informations, veuillez consulter the section called “Exécuter les modules complémentaires critiques”.

Agent d'identité du pod EKS

Le module complémentaire Agent d’identité du pod Amazon EKS permet de gérer les informations d’identification de vos applications, de la même manière que les profils d’instance EC2 fournissent des informations d’identification aux instances EC2.

> **Note**
>
> Vous n’avez pas besoin d’installer ce module complémentaire sur les clusters du mode automatique Amazon EKS. Le mode automatique Amazon EKS s’intègre à l’identité du pod Amazon EKS. Pour de plus amples informations, veuillez consulter the section called “Considérations relatives au mode automatique Amazon EKS”.

Le nom du module complémentaire Amazon EKS est `eks-pod-identity-agent`.

Autorisations IAM requises

Le module complémentaire Agent d’identité du pod Amazon EKS ne nécessite pas de rôle IAM. Il utilise les autorisations du rôle IAM du nœud Amazon EKS pour fonctionner, mais ne nécessite pas de rôle IAM dédié pour le module complémentaire.

Mise à jour des informations

Vous ne pouvez mettre à jour qu'une seule version mineure à la fois. Par exemple, si votre version actuelle est `1.28.x-eksbuild.y` et que vous voulez la mettre à jour vers `1.30.x-eksbuild.y`, vous devez d'abord mettre à jour votre version actuelle vers `1.29.x-eksbuild.y` et ensuite la mettre à nouveau à jour vers `1.30.x-eksbuild.y`. Pour plus d'informations sur la mise à

jour du module complémentaire, consultez la rubrique the section called "Mettre à jour un module complémentaire".

Exportateur de métriques réseau SR-IOV

Le module complémentaire Exportateur de métriques réseau SR-IOV Amazon EKS collecte et expose les métriques relatives aux périphériques réseau SR-IOV au format Prometheus. Il permet de surveiller les performances du réseau SR-IOV sur les nœuds de matériel nu EKS. L'exportateur fonctionne comme un nœud DaemonSet doté d'interfaces SR-IOV-capable réseau et exporte des métriques qui peuvent être récupérées par Prometheus.

> **Note**
>
> Ce module complémentaire nécessite des nœuds dotés d'interfaces SR-IOV-capable réseau.

Propriété	Value
Nom du module complémentaire	`sriov-network-metrics-exporter`
Namespace	`monitoring`
Documentation	Dépôt de l'exportateur de métriques réseau SR-IOV GitHub
Nom du compte de service	Aucune
Politique IAM gérée	Aucune
Autorisations IAM personnalisées	Aucune

AWS Fournisseur de pilotes CSI Secrets Store

Le AWS fournisseur du pilote CSI Secrets Store est un module complémentaire qui permet de récupérer des AWS secrets depuis Secrets Manager et des paramètres depuis AWS Systems Manager Parameter Store et de les monter sous forme de fichiers dans des pods Kubernetes.

Autorisations IAM requises

Le module complémentaire ne nécessite pas d'autorisations IAM. Toutefois, les modules d'application nécessiteront des autorisations IAM pour récupérer les AWS secrets depuis Secrets Manager et les paramètres depuis le magasin de paramètres AWS Systems Manager. Après avoir installé le module complémentaire, l'accès doit être configuré via IAM Roles for Service Accounts (IRSA) ou EKS Pod Identity. Pour utiliser IRSA, reportez-vous à la documentation de configuration de Secrets Manager IRSA. Pour utiliser EKS Pod Identity, reportez-vous à la documentation de configuration de Secrets Manager Pod Identity.

AWS suggère la politique `AWSSecretsManagerClientReadOnlyAccess` gérée.

Pour plus d'informations sur les autorisations requises, consultez `AWSSecretsManagerClientReadOnlyAccess` la référence des politiques AWS gérées.

Informations supplémentaires

Pour plus d'informations, consultez le référentiel secrets-store-csi-driver -provider-awsGitHub .

Pour en savoir plus sur le module complémentaire, consultez sa documentation relative àAWS Secrets Manager.

Amazon SageMaker Spaces

Le module complémentaire Amazon SageMaker Spaces permet d'exécuter IDEs des blocs-notes sur des clusters EKS ou HyperPod -EKS. Les administrateurs peuvent utiliser la console EKS pour installer le module complémentaire sur leur cluster et définir des configurations d'espace par défaut telles que les images, les ressources de calcul, le stockage local pour les paramètres du bloc-notes (stockage supplémentaire à associer à leurs espaces), les systèmes de fichiers et les scripts d'initialisation.

Les développeurs d'IA peuvent utiliser kubectl pour créer, mettre à jour et supprimer des espaces. Ils ont la possibilité d'utiliser les configurations par défaut fournies par les administrateurs ou de personnaliser les paramètres. Les développeurs d'IA peuvent accéder à leurs espaces sur EKS ou HyperPod -EKS à l'aide de leur code VS local IDEs, de and/or leur navigateur Web qui héberge leur espace JupyterLab ou de leur CodeEditor IDE sur un domaine DNS personnalisé configuré par leurs administrateurs. Ils peuvent également utiliser la fonction de redirection de port de Kubernetes pour accéder aux espaces de leurs navigateurs Web.

Le nom du module complémentaire Amazon EKS est `amazon-sagemaker-spaces`.

Autorisations IAM requises

Ce module complémentaire nécessite des autorisations IAM. Pour plus d'informations sur la configuration IAM requise, consultez la section Configuration des autorisations IAM dans le manuel Amazon SageMaker Developer Guide.

Informations supplémentaires

Pour en savoir plus sur le module complémentaire et ses fonctionnalités, consultez SageMaker AI Notebooks HyperPod dans le manuel Amazon SageMaker Developer Guide.

Extensions communautaires

Vous pouvez l'utiliser AWS APIs pour installer des modules complémentaires communautaires, tels que le serveur Kubernetes Metrics. Vous pouvez choisir d'installer ces modules complémentaires communautaires sous forme de modules complémentaires Amazon EKS afin de réduire la complexité de gestion du logiciel sur plusieurs clusters.

Par exemple, vous pouvez utiliser l' AWS API, la CLI ou la console de gestion pour installer des modules complémentaires communautaires. Il est également possible d'installer un module complémentaire communautaire lors de la création d'un cluster.

Vous gérez ces modules complémentaires communautaires de la même manière que les modules complémentaires Amazon EKS existants. Les modules complémentaires communautaires se distinguent des modules complémentaires existants par un champ d'assistance spécifique.

> **Note**
>
> L'utilisation des modules complémentaires communautaires est à votre discrétion. Dans le cadre du modèle de responsabilité partagée entre vous et AWS, vous êtes censé comprendre ce que vous installez dans votre cluster pour ces plugins tiers. Vous êtes également responsable de la conformité de ces modules complémentaires communautaires avec vos exigences de sécurité de cluster. Pour de plus amples informations, veuillez consulter the section called "Prise en charge des logiciels déployés sur EKS".

Les modules complémentaires communautaires ne sont pas créés par AWS. AWS valide uniquement les modules complémentaires communautaires pour la compatibilité des versions. Par exemple, si vous installez un module complémentaire communautaire sur un cluster, vérifiez AWS s'il est compatible avec la version Kubernetes de votre cluster.

Surtout, AWS ne fournit pas un support complet pour les modules complémentaires communautaires. AWS prend uniquement en charge les opérations de cycle de vie effectuées à l'aide de modules complémentaires AWS APIs, telles que l'installation ou la suppression de modules complémentaires.

Si vous avez besoin d'une assistance technique pour un module complémentaire communautaire, vous devez utiliser les ressources existantes du projet. Par exemple, vous pouvez créer un GitHub problème sur le dépôt du projet.

Détermination du type de module complémentaire

Vous pouvez utiliser la AWS CLI pour déterminer le type d'un module complémentaire Amazon EKS.

Exécutez la commande CLI suivante pour extraire des informations détaillées sur un module complémentaire. Vous pouvez remplacer `metrics-server` par le nom de n'importe quel module complémentaire.

```
aws eks describe-addon-versions --addon-name metrics-server
```

Vérifiez le résultat de la commande CLI dans le champ `owner`.

```
{
    "addons": [
        {
            "addonName": "metrics-server",
            "type": "observability",
            "owner": "community",
            "addonVersions": [
```

Si la valeur de `owner` est`community`, le module complémentaire est un module complémentaire communautaire. AWS fournit uniquement un support pour l'installation, la mise à jour et la suppression du module complémentaire. Si vous avez des questions sur les fonctionnalités et le fonctionnement du module complémentaire lui-même, utilisez les ressources communautaires telles que GitHub les problèmes.

Installation ou mise à jour d'un module complémentaire communautaire

L'installation ou la mise à jour des modules complémentaires communautaires se fait de la même manière que pour les autres modules complémentaires Amazon EKS.

- the section called "Créer un module complémentaire"

- the section called "Mettre à jour un module complémentaire"
- the section called "Supprimer un module complémentaire"

Modules complémentaires communautaires disponibles

Les modules complémentaires communautaires suivants sont disponibles à partir d'Amazon EKS.

- the section called "Serveur de métriques Kubernetes"
- the section called "kube-state-metrics"
- the section called "Exportateur de nœuds Prometheus"
- the section called "Gestionnaire de certificats"
- the section called "DNS externe"
- the section called "Fluent Bit"

Serveur de métriques Kubernetes

Le serveur de métriques Kubernetes est une source évolutive et efficace de métriques sur les ressources des conteneurs, utilisée dans les pipelines de mise à l'échelle automatique intégrés à Kubernetes. Il collecte les métriques de ressources à partir des Kubelets et les rend accessibles dans l'API Server de Kubernetes via l'API Metrics, utilisée par le Horizontal Pod Autoscaler et le Vertical Pod Autoscaler.

Propriété	Value
Nom du module complémentaire	`metrics-server`
Namespace	`kube-system`
Documentation	GitHub Readme
Nom du compte de service	Aucune
Politique IAM gérée	Aucune
Autorisations IAM personnalisées	Aucune

kube-state-metrics

Agent de module complémentaire conçu pour générer et exposer des métriques au niveau du cluster.

L'état des objets Kubernetes présents dans l'API Kubernetes peut être exposé sous forme de métriques. Un agent complémentaire appelé kube-state-metrics peut se connecter au serveur d'API Kubernetes et exposer un point de terminaison HTTP avec des métriques générées à partir de l'état des objets individuels du cluster. Il expose diverses informations telles que les étiquettes et annotations, les temps de démarrage et d'arrêt, l'état ou la phase actuelle de l'objet.

Propriété	Value
Nom du module complémentaire	`kube-state-metrics`
Namespace	`kube-state-metrics`
Documentation	Métriques relatives aux états des objets Kubernetes dans la documentation Kubernetes
Nom du compte de service	Aucune
Politique IAM gérée	Aucune
Autorisations IAM personnalisées	Aucune

Exportateur de nœuds Prometheus

Exportateur Prometheus chargé de collecter les métriques matérielles et système d'exploitation exposées par les noyaux de systèmes de type *NIX, développé en Go et doté de collecteurs de métriques modulables. L'exportateur de nœuds Prometheus expose une large gamme de métriques liées au matériel et au noyau.

Propriété	Value
Nom du module complémentaire	`prometheus-node-exporter`
Namespace	`prometheus-node-exporter`

Propriété	Value
Documentation	Surveillance des métriques des hôtes Linux avec l'exportateur de nœuds dans la documentation Prometheus
Nom du compte de service	Aucune
Politique IAM gérée	Aucune
Autorisations IAM personnalisées	Aucune

Gestionnaire de certificats

Le gestionnaire de certificats permet de gérer la création et le renouvellement des certificats.

Propriété	Value
Nom du module complémentaire	`cert-manager`
Namespace	`cert-manager`
Documentation	Documents relatifs au gestionnaire de certificats
Nom du compte de service	Aucune
Politique IAM gérée	Aucune
Autorisations IAM personnalisées	Aucune

DNS externe

Le module complémentaire EKS DNS externe peut être utilisé pour gérer les enregistrements DNS Route53 via les ressources Kubernetes.

Les autorisations DNS externes peuvent être réduites à `route53:ChangeResourceRecordSets`, `route53:ListHostedZones` et `route53:ListResourceRecordSets`pour les zones hébergées que vous souhaitez administrer.

Propriété	Value
Nom du module complémentaire	`external-dns`
Namespace	`external-dns`
Documentation	GitHub Readme
Nom du compte de service	`external-dns`
Politique IAM gérée	`arn:aws: iam::aws:policy/AmazonRoute53FullAccess`
Autorisations IAM personnalisées	Aucune

Fluent Bit

Fluent Bit est un processeur et collecteur de journaux léger et à haut rendement. Il vous permet data/logs de collecter des données auprès de différentes sources, de les unifier et de les envoyer vers plusieurs destinations, notamment Amazon CloudWatch Logs, Amazon S3 et Amazon Kinesis Data Firehose. Fluent Bit est conçu pour offrir d'excellentes performances et une grande efficacité en matière d'utilisation des ressources, ce qui le rend idéal pour les environnements Kubernetes.

Ce module complémentaire ne nécessite pas d'autorisations IAM dans sa configuration par défaut. Toutefois, vous devrez peut-être accorder à ce module complémentaire des autorisations IAM si vous configurez un emplacement AWS de sortie. Pour de plus amples informations, veuillez consulter the section called "Utiliser les identités du pod".

Propriété	Value
Nom du module complémentaire	`fluent-bit`
Namespace	`fluent-bit`
Documentation	Documentation Fluent Bit
Nom du compte de service	`fluent-bit`
Politique IAM gérée	Aucune

Propriété	Value
Autorisations IAM personnalisées	Aucune

Consultation des attributions

Vous pouvez télécharger les attributions open source et les informations de licence pour les modules complémentaires communautaires.

1. Déterminez le nom et la version du module complémentaire pour lequel vous souhaitez télécharger les attributions.
2. Mettez à jour la commande suivante avec le nom et la version :

   ```
   curl -O https://amazon-eks-docs.s3.amazonaws.com/attributions/<add-on-name>/<add-on-version>/attributions.zip
   ```

   Par exemple :

   ```
   curl -O https://amazon-eks-docs.s3.amazonaws.com/attributions/kube-state-metrics/v2.14.0-eksbuild.1/attributions.zip
   ```
3. Exécutez la commande pour télécharger le fichier.

Utilisez ensuite le fichier ZIP obtenu pour consulter les informations sur les licences et attributions.

AWS Modules complémentaires Marketplace

Outre la liste précédente de modules complémentaires Amazon EKS, vous pouvez également ajouter une large sélection de modules complémentaires Amazon EKS pour logiciels opérationnels proposés par des fournisseurs indépendants de logiciels. Sélectionnez un module complémentaire pour en savoir plus à son sujet et sur les conditions d'installation de celui-ci.

Accuknox

Le nom du module complémentaire est `accuknox_kubearmor` et l'espace de noms est `kubearmor`. Accuknox diffuse le module complémentaire.

Pour plus d'informations sur le module complémentaire, consultez Getting Started with KubeArmor dans la KubeArmor documentation.

Nom du compte de service

Aucun compte de service n'est utilisé avec ce module complémentaire.

AWS politique IAM gérée

Aucune politique gérée n'est utilisée avec ce module complémentaire.

Autorisations IAM personnalisées

Les autorisations personnalisées ne sont pas utilisées avec ce module complémentaire.

Akuity

Le nom du module complémentaire est `akuity_agent` et l'espace de noms est `akuity`. Akuity diffuse le module complémentaire.

Pour plus d'informations sur le module complémentaire, consultez Installation de l'agent Akuity sur Amazon EKS avec le module complémentaire Akuity pour Amazon EKS dans la documentation de la plateforme Akuity.

Nom du compte de service

Aucun compte de service n'est utilisé avec ce module complémentaire.

AWS politique IAM gérée

Aucune politique gérée n'est utilisée avec ce module complémentaire.

Autorisations IAM personnalisées

Les autorisations personnalisées ne sont pas utilisées avec ce module complémentaire.

Calyptia

Le nom du module complémentaire est `calyptia_fluent-bit` et l'espace de noms est `calytia-fluentbit`. Calyptia diffuse le module complémentaire.

Pour plus d'informations sur le module complémentaire, consultez Prise en main de Calyptia Core Agent sur le site Web de documentation Calyptia.

Nom du compte de service

Le nom du compte de service est `clyptia-fluentbit`.

AWS politique IAM gérée

Ce module complémentaire utilise la politique gérée `AWSMarketplaceMeteringRegisterUsage`. Pour plus d'informations, consultez AWSMarketplaceMeteringRegisterUsagele Guide de référence des politiques AWS gérées.

Commande pour créer le rôle IAM requis

La commande suivante nécessite que vous disposiez d'un fournisseur IAM OpenID Connect (OIDC) pour votre cluster. Pour déterminer si vous en avez un, ou pour en créer un, consultez the section called "Fournisseur IAM OIDC". Remplacez *`my-cluster`* par le nom de votre cluster et *`my-calyptia-role`* par le nom que vous souhaitez pour votre rôle. Cette commande nécessite que eksctl soit installé sur votre appareil. Si vous devez utiliser un autre outil pour créer le rôle et annoter le compte de service Kubernetes, consultez the section called "Attribution d'un rôle IAM".

```
eksctl create iamserviceaccount --name service-account-name  --namespace calyptia-
fluentbit --cluster my-cluster --role-name my-calyptia-role \
    --role-only --attach-policy-arn arn:aws: iam::aws:policy/
AWSMarketplaceMeteringRegisterUsage --approve
```

Collecteur d'observabilité Cisco

Le nom du module complémentaire est `cisco_cisco-cloud-observability-collectors` et l'espace de noms est `appdynamics`. Cisco publie le module complémentaire.

Pour plus d'informations sur le module complémentaire, voir Utiliser les modules complémentaires Cisco Cloud Observability AWS Marketplace dans la AppDynamics documentation Cisco.

Nom du compte de service

Aucun compte de service n'est utilisé avec ce module complémentaire.

AWS politique IAM gérée

Aucune politique gérée n'est utilisée avec ce module complémentaire.

Autorisations IAM personnalisées

Les autorisations personnalisées ne sont pas utilisées avec ce module complémentaire.

Opérateur d'observabilité Cisco

Le nom du module complémentaire est `cisco_cisco-cloud-observability-operators` et l'espace de noms est `appdynamics`. Cisco publie ce module complémentaire.

Pour plus d'informations sur le module complémentaire, voir Utiliser les modules complémentaires Cisco Cloud Observability AWS Marketplace dans la AppDynamics documentation Cisco.

Nom du compte de service

Aucun compte de service n'est utilisé avec ce module complémentaire.

AWS politique IAM gérée

Aucune politique gérée n'est utilisée avec ce module complémentaire.

Autorisations IAM personnalisées

Les autorisations personnalisées ne sont pas utilisées avec ce module complémentaire.

CLOUDSOFT

Le nom du module complémentaire est `cloudsoft_cloudsoft-amp` et l'espace de noms est `cloudsoft-amp`. CLOUDSOFT diffuse ce module complémentaire.

Pour plus d'informations sur ce module complémentaire, consultez Module complémentaire Amazon EKS dans la documentation CLOUDSOFT.

Nom du compte de service

Aucun compte de service n'est utilisé avec ce module complémentaire.

AWS politique IAM gérée

Aucune politique gérée n'est utilisée avec ce module complémentaire.

Autorisations IAM personnalisées

Les autorisations personnalisées ne sont pas utilisées avec ce module complémentaire.

Cribl

Le nom du module complémentaire est `cribl_cribledge` et l'espace de noms est `cribledge`. Cribl diffuse ce module complémentaire.

Pour plus d'informations sur ce module complémentaire, consultez la section Installation du module complémentaire Cribl pour Amazon EKS pour Edge dans la documentation Cribl

Nom du compte de service

Aucun compte de service n'est utilisé avec ce module complémentaire.

AWS politique IAM gérée

Aucune politique gérée n'est utilisée avec ce module complémentaire.

Autorisations IAM personnalisées

Les autorisations personnalisées ne sont pas utilisées avec ce module complémentaire.

Dynatrace

Le nom du module complémentaire est `dynatrace_dynatrace-operator` et l'espace de noms est `dynatrace`. Dynatrace diffuse ce module complémentaire.

Pour plus d'informations sur le module complémentaire, consultez Surveillance Kubernetes dans la documentation Dynatrace.

Nom du compte de service

Aucun compte de service n'est utilisé avec ce module complémentaire.

AWS politique IAM gérée

Aucune politique gérée n'est utilisée avec ce module complémentaire.

Autorisations IAM personnalisées

Les autorisations personnalisées ne sont pas utilisées avec ce module complémentaire.

Datree

Le nom du module complémentaire est `datree_engine-pro` et l'espace de noms est `datree`. Datree publie ce module complémentaire.

Pour plus d'informations sur le module complémentaire, consultez Intégration Amazon EKS dans la documentation Datree.

Nom du compte de service

Le nom du compte de service est datree-webhook-server-awsmp.

AWS politique IAM gérée

La politique gérée est AWSLicenseManagerConsumptionPolicy. Pour plus d'informations, consultez AWSLicenseManagerConsumptionPolicyle Guide de référence des politiques AWS gérées.

Commande pour créer le rôle IAM requis

La commande suivante nécessite que vous disposiez d'un fournisseur IAM OpenID Connect (OIDC) pour votre cluster. Pour déterminer si vous en avez un, ou pour en créer un, consultez the section called "Fournisseur IAM OIDC". Remplacez *my-cluster* par le nom de votre cluster et *my-datree-role* par le nom que vous souhaitez pour votre rôle. Cette commande nécessite que eksctl soit installé sur votre appareil. Si vous devez utiliser un autre outil pour créer le rôle et annoter le compte de service Kubernetes, consultez the section called "Attribution d'un rôle IAM".

```
eksctl create iamserviceaccount --name datree-webhook-server-awsmp --namespace datree
 --cluster my-cluster --role-name my-datree-role \
    --role-only --attach-policy-arn arn:aws: iam::aws:policy/service-role/
AWSLicenseManagerConsumptionPolicy --approve
```

Autorisations personnalisées

Les autorisations personnalisées ne sont pas utilisées avec ce module complémentaire.

Datadog

Le nom du module complémentaire est `datadog_operator` et l'espace de noms est `datadog-agent`. Datadog diffuse le module complémentaire.

Pour plus d'informations sur le module complémentaire, consultez Installation de l'agent Datadog sur Amazon EKS avec le module complémentaire Opérateur Datadog dans la documentation Datadog.

Nom du compte de service

Aucun compte de service n'est utilisé avec ce module complémentaire.

AWS politique IAM gérée

Aucune politique gérée n'est utilisée avec ce module complémentaire.

Autorisations IAM personnalisées

Les autorisations personnalisées ne sont pas utilisées avec ce module complémentaire.

Groundcover

Le nom du module complémentaire est `groundcover_agent` et l'espace de noms est `groundcover`. Groundcover diffuse le module complémentaire.

Pour plus d'informations sur le module complémentaire, consultez Installation du module complémentaire Groundcover pour Amazon EKS dans la documentation Groundcover.

Nom du compte de service

Aucun compte de service n'est utilisé avec ce module complémentaire.

AWS politique IAM gérée

Aucune politique gérée n'est utilisée avec ce module complémentaire.

Autorisations IAM personnalisées

Les autorisations personnalisées ne sont pas utilisées avec ce module complémentaire.

IBM Instana

Le nom du module complémentaire est `instana-agent` et l'espace de noms est `instana-agent`. IBM diffuse le module complémentaire.

Pour plus d'informations sur le module complémentaire, consultez Implémenter l'observabilité pour les charges de travail Amazon EKS à l'aide du module complémentaire Instana Amazon EKS et surveillez et optimisez les coûts Amazon EKS avec IBM Instana et Kubecost sur le blog. AWS

Observabilité Instana (Instana) propose un module complémentaire Amazon EKS qui déploie des agents Instana sur les clusters Amazon EKS. Les clients peuvent utiliser ce module complémentaire pour collecter et analyser des données de performances en temps réel afin d'obtenir des informations sur leurs applications conteneurisées. Le module complémentaire Instana pour Amazon EKS offre une visibilité sur l'ensemble de vos environnements Kubernetes. Une fois déployé, l'agent Instana détecte automatiquement les composants de vos clusters Amazon EKS, notamment les nœuds, les espaces de noms, les déploiements, les services et les pods.

Nom du compte de service

Aucun compte de service n'est utilisé avec ce module complémentaire.

AWS politique IAM gérée

Aucune politique gérée n'est utilisée avec ce module complémentaire.

Autorisations IAM personnalisées

Les autorisations personnalisées ne sont pas utilisées avec ce module complémentaire.

Grafana Labs

Le nom du module complémentaire est `grafana-labs_kubernetes-monitoring` et l'espace de noms est `monitoring`. Grafana Labs diffuse ce module complémentaire.

Pour plus d'informations sur ce module complémentaire, consultez Configurer la surveillance Kubernetes en tant que module complémentaire avec Amazon EKS dans la documentation Grafana Labs.

Nom du compte de service

Aucun compte de service n'est utilisé avec ce module complémentaire.

AWS politique IAM gérée

Aucune politique gérée n'est utilisée avec ce module complémentaire.

Autorisations IAM personnalisées

Les autorisations personnalisées ne sont pas utilisées avec ce module complémentaire.

Guance

- Diffuseur : GUANCE
- Nom – `guance_datakit`
- Espace de noms – `datakit`
- Nom du compte de service : aucun compte de service n'est utilisé avec ce module complémentaire.
- Politique IAM gérée par AWS : aucune politique gérée n'est utilisée avec ce module complémentaire.

- Autorisations IAM personnalisées : aucune autorisation personnalisée n'est utilisée avec ce module complémentaire.
- Instructions de configuration et d'utilisation : consultez Utilisation du module complémentaire Amazon EKS dans la documentation Guance.

HA Proxy

Son nom est `haproxy-technologies_kubernetes-ingress-ee` et son espace de noms est `haproxy-controller`. HA Proxy diffuse le module complémentaire.

Pour plus d'informations sur le module complémentaire, consultez Intégration Amazon EKS dans la documentation Datree.

Nom du compte de service

Le nom du compte de service est `customer defined`.

AWS politique IAM gérée

La politique gérée est AWSLicenseManagerConsumptionPolicy. Pour plus d'informations, consultez AWSLicenseManagerConsumptionPolicyle Guide de référence des politiques AWS gérées.

Commande pour créer le rôle IAM requis

La commande suivante nécessite que vous disposiez d'un fournisseur IAM OpenID Connect (OIDC) pour votre cluster. Pour déterminer si vous en avez un, ou pour en créer un, consultez the section called "Fournisseur IAM OIDC". Remplacez *`my-cluster`* par le nom de votre cluster et *`my-haproxy-role`* par le nom que vous souhaitez pour votre rôle. Cette commande nécessite que eksctl soit installé sur votre appareil. Si vous devez utiliser un autre outil pour créer le rôle et annoter le compte de service Kubernetes, consultez the section called "Attribution d'un rôle IAM".

```
eksctl create iamserviceaccount --name service-account-name  --namespace haproxy-
controller --cluster my-cluster --role-name my-haproxy-role \
    --role-only --attach-policy-arn arn:aws: iam::aws:policy/service-role/
AWSLicenseManagerConsumptionPolicy --approve
```

Autorisations personnalisées

Les autorisations personnalisées ne sont pas utilisées avec ce module complémentaire.

Kpow

Le nom du module complémentaire est `factorhouse_kpow` et l'espace de noms est `factorhouse`. Factorhouse diffuse le module complémentaire.

Pour plus d'informations sur le module complémentaire, consultez AWS Marketplace LM dans la documentation Kpow.

Nom du compte de service

Le nom du compte de service est `kpow`.

AWS politique IAM gérée

La politique gérée est AWSLicenseManagerConsumptionPolicy. Pour plus d'informations, consultez AWSLicenseManagerConsumptionPolicyle Guide de référence des politiques AWS gérées.

Commande pour créer le rôle IAM requis

La commande suivante nécessite que vous disposiez d'un fournisseur IAM OpenID Connect (OIDC) pour votre cluster. Pour déterminer si vous en avez un, ou pour en créer un, consultez the section called "Fournisseur IAM OIDC". Remplacez *my-cluster* par le nom de votre cluster et *my-kpow-role* par le nom que vous souhaitez pour votre rôle. Cette commande nécessite que eksctl soit installé sur votre appareil. Si vous devez utiliser un autre outil pour créer le rôle et annoter le compte de service Kubernetes, consultez the section called "Attribution d'un rôle IAM".

```
eksctl create iamserviceaccount --name kpow --namespace factorhouse --cluster my-
cluster --role-name my-kpow-role \
    --role-only --attach-policy-arn arn:aws: iam::aws:policy/service-role/
AWSLicenseManagerConsumptionPolicy --approve
```

Autorisations personnalisées

Les autorisations personnalisées ne sont pas utilisées avec ce module complémentaire.

Kubecost

Le nom du module complémentaire est `kubecost_kubecost` et l'espace de noms est `kubecost`. Kubecost diffuse le module complémentaire.

Pour plus d'informations sur le module complémentaire, consultez Intégration de la facturation cloud AWS dans la documentation Kubecost.

Vous devez avoir installé Stocker des volumes Kubernetes avec Amazon EBS sur votre cluster. Sinon, vous recevrez une erreur.

Nom du compte de service

Aucun compte de service n'est utilisé avec ce module complémentaire.

AWS politique IAM gérée

Aucune politique gérée n'est utilisée avec ce module complémentaire.

Autorisations IAM personnalisées

Les autorisations personnalisées ne sont pas utilisées avec ce module complémentaire.

Kasten

Le nom du module complémentaire est `kasten_k10` et l'espace de noms est `kasten-io`. Kasten by Veeam diffuse le module complémentaire.

Pour plus d'informations sur le module complémentaire, consultez la section Installation du K10 à l' AWS aide du module complémentaire Amazon EKS dans la documentation de Kasten.

Vous devez avoir installé le pilote CSI Amazon EBS sur votre cluster avec un `StorageClass` par défaut.

Nom du compte de service

Le nom du compte de service est `k10-k10`.

AWS politique IAM gérée

La politique gérée est AWSLicenseManagerConsumptionPolicy. Pour plus d'informations, consultez AWSLicenseManagerConsumptionPolicyle Guide de référence des politiques AWS gérées.

Commande pour créer le rôle IAM requis

La commande suivante nécessite que vous disposiez d'un fournisseur IAM OpenID Connect (OIDC) pour votre cluster. Pour déterminer si vous en avez un, ou pour en créer un, consultez the section called "Fournisseur IAM OIDC". Remplacez *my-cluster* par le nom de votre cluster et *my-kasten-*

role par le nom que vous souhaitez pour votre rôle. Cette commande nécessite que eksctl soit installé sur votre appareil. Si vous devez utiliser un autre outil pour créer le rôle et annoter le compte de service Kubernetes, consultez the section called "Attribution d'un rôle IAM".

```
eksctl create iamserviceaccount --name k10-k10 --namespace kasten-io --cluster my-
cluster --role-name my-kasten-role \
    --role-only --attach-policy-arn arn:aws: iam::aws:policy/service-role/
AWSLicenseManagerConsumptionPolicy --approve
```

Autorisations personnalisées

Les autorisations personnalisées ne sont pas utilisées avec ce module complémentaire.

Kong

Le nom du module complémentaire est `kong_konnect-ri` et l'espace de noms est `kong`. Kong diffuse le module complémentaire.

Pour plus d'informations sur ce module complémentaire, consultez Installation du module complémentaire Kong Gateway EKS dans la documentation Kong.

Vous devez avoir installé Stocker des volumes Kubernetes avec Amazon EBS sur votre cluster. Sinon, vous recevrez une erreur.

Nom du compte de service

Aucun compte de service n'est utilisé avec ce module complémentaire.

AWS politique IAM gérée

Aucune politique gérée n'est utilisée avec ce module complémentaire.

Autorisations IAM personnalisées

Les autorisations personnalisées ne sont pas utilisées avec ce module complémentaire.

LeakSignal

Le nom du module complémentaire est `leaksignal_leakagent` et l'espace de noms est`leakagent`. LeakSignal publie le module complémentaire.

Pour plus d'informations sur le module complémentaire, consultez https://www.leaksignal.com/docs/LeakAgent /Deployment/AWS%20Eks%20Addon/ [Installer le module complémentaire] dans la documentation LeakAgent LeakSignal

Vous devez avoir installé Stocker des volumes Kubernetes avec Amazon EBS sur votre cluster. Sinon, vous recevrez une erreur.

Nom du compte de service

Aucun compte de service n'est utilisé avec ce module complémentaire.

AWS politique IAM gérée

Aucune politique gérée n'est utilisée avec ce module complémentaire.

Autorisations IAM personnalisées

Les autorisations personnalisées ne sont pas utilisées avec ce module complémentaire.

NetApp

Le nom du module complémentaire est `netapp_trident-operator` et l'espace de noms est`trident`. NetApp publie le module complémentaire.

Pour plus d'informations sur le module complémentaire, voir Configurer le module complémentaire Trident EKS dans la NetApp documentation.

Nom du compte de service

Aucun compte de service n'est utilisé avec ce module complémentaire.

AWS politique IAM gérée

Aucune politique gérée n'est utilisée avec ce module complémentaire.

Autorisations IAM personnalisées

Les autorisations personnalisées ne sont pas utilisées avec ce module complémentaire.

New Relic

Le nom du module complémentaire est `new-relic_kubernetes-operator` et l'espace de noms est `newrelic`. New Relic diffuse le module complémentaire.

Pour plus d'informations sur le module complémentaire, consultez Installer le module complémentaire New Relic pour EKS dans la documentation New Relic.

Nom du compte de service

Aucun compte de service n'est utilisé avec ce module complémentaire.

AWS politique IAM gérée

Aucune politique gérée n'est utilisée avec ce module complémentaire.

Autorisations IAM personnalisées

Les autorisations personnalisées ne sont pas utilisées avec ce module complémentaire.

Rafay

Le nom du module complémentaire est `rafay-systems_rafay-operator` et l'espace de noms est `rafay-system`. Rafay diffuse ce module complémentaire.

Pour plus d'informations sur ce module complémentaire, consultez Installation du module complémentaire Rafay pour Amazon EKS dans la documentation Rafay.

Nom du compte de service

Aucun compte de service n'est utilisé avec ce module complémentaire.

AWS politique IAM gérée

Aucune politique gérée n'est utilisée avec ce module complémentaire.

Autorisations IAM personnalisées

Les autorisations personnalisées ne sont pas utilisées avec ce module complémentaire.

Rad Security

- Diffuseur : RAD SECURITY
- Nom – `rad-security_rad-security`
- Espace de noms – `ksoc`

- Nom du compte de service : aucun compte de service n'est utilisé avec ce module complémentaire.
- Politique IAM gérée par AWS : aucune politique gérée n'est utilisée avec ce module complémentaire.
- Autorisations IAM personnalisées : aucune autorisation personnalisée n'est utilisée avec ce module complémentaire.
- Instructions de configuration et d'utilisation — Voir Installation de Rad Through The AWS Marketplace dans la documentation de Rad Security.

SolarWinds

- Diffuseur : SOLARWINDS
- Nom – `solarwinds_swo-k8s-collector-addon`
- Espace de noms – `solarwinds`
- Nom du compte de service : aucun compte de service n'est utilisé avec ce module complémentaire.
- Politique IAM gérée par AWS : aucune politique gérée n'est utilisée avec ce module complémentaire.
- Autorisations IAM personnalisées : aucune autorisation personnalisée n'est utilisée avec ce module complémentaire.
- Instructions de configuration et d'utilisation : consultez la section Surveiller un cluster Amazon EKS dans la SolarWinds documentation.

Solo

Le nom du module complémentaire est `solo-io_istio-distro` et l'espace de noms est `istio-system`. Solo diffuse le module complémentaire.

Pour plus d'informations sur l'extension, consultez Installation d'Istio dans la documentation Solo.io.

Nom du compte de service

Aucun compte de service n'est utilisé avec ce module complémentaire.

AWS politique IAM gérée

Aucune politique gérée n'est utilisée avec ce module complémentaire.

Autorisations IAM personnalisées

Les autorisations personnalisées ne sont pas utilisées avec ce module complémentaire.

Snyk

- Diffuseur : SNYK
- Nom – `snyk_runtime-sensor`
- Espace de noms – `snyk_runtime-sensor`
- Nom du compte de service : aucun compte de service n'est utilisé avec ce module complémentaire.
- Politique IAM gérée par AWS : aucune politique gérée n'est utilisée avec ce module complémentaire.
- Autorisations IAM personnalisées : aucune autorisation personnalisée n'est utilisée avec ce module complémentaire.
- Instructions d'installation et d'utilisation : consultez Capteur d'exécution Snyk dans la documentation utilisateur Snyk.

Stormforge

Le nom du module complémentaire est `stormforge_optimize-Live` et l'espace de noms est `stormforge-system`. Stormforge diffuse le module complémentaire.

Pour plus d'informations sur le module complémentaire, consultez la section Installation de l' StormForge agent dans la StormForge documentation.

Nom du compte de service

Aucun compte de service n'est utilisé avec ce module complémentaire.

AWS politique IAM gérée

Aucune politique gérée n'est utilisée avec ce module complémentaire.

Autorisations IAM personnalisées

Les autorisations personnalisées ne sont pas utilisées avec ce module complémentaire.

SUSE

- Diffuseur : SUSE

- Nom – `suse_observability-agent`
- Espace de noms – `suse-observability`
- Nom du compte de service : aucun compte de service n'est utilisé avec ce module complémentaire.
- Politique IAM gérée par AWS : aucune politique gérée n'est utilisée avec ce module complémentaire.
- Autorisations IAM personnalisées : aucune autorisation personnalisée n'est utilisée avec ce module complémentaire.
- Instructions d'installation et d'utilisation : consultez Démarrage rapide dans la documentation SUSE.

Splunk

Le nom du module complémentaire est `splunk_splunk-otel-collector-chart` et l'espace de noms est `splunk-monitoring`. Splunk diffuse ce module complémentaire.

Pour plus d'informations sur ce module complémentaire, consultez Installation du module complémentaire Splunk pour Amazon EKS dans la documentation Splunk.

Nom du compte de service

Aucun compte de service n'est utilisé avec ce module complémentaire.

AWS politique IAM gérée

Aucune politique gérée n'est utilisée avec ce module complémentaire.

Autorisations IAM personnalisées

Les autorisations personnalisées ne sont pas utilisées avec ce module complémentaire.

Teleport

Le nom du module complémentaire est `teleport_teleport` et l'espace de noms est `teleport`. Teleport diffuse le module complémentaire.

Pour plus d'informations sur le module complémentaire, consultez Fonctionnement de Teleport dans la documentation Teleport.

Nom du compte de service

Aucun compte de service n'est utilisé avec ce module complémentaire.

AWS politique IAM gérée

Aucune politique gérée n'est utilisée avec ce module complémentaire.

Autorisations IAM personnalisées

Les autorisations personnalisées ne sont pas utilisées avec ce module complémentaire.

Tetrate

Le nom du module complémentaire est `tetrate-io_istio-distro` et l'espace de noms est `istio-system`. Tetrate Io diffuse le module complémentaire.

Pour plus d'informations sur le module complémentaire, consultez le site Web Tetrate Istio Distro.

Nom du compte de service

Aucun compte de service n'est utilisé avec ce module complémentaire.

AWS politique IAM gérée

Aucune politique gérée n'est utilisée avec ce module complémentaire.

Autorisations IAM personnalisées

Les autorisations personnalisées ne sont pas utilisées avec ce module complémentaire.

Upbound Universal Crossplane

Le nom du module complémentaire est `upbound_universal-crossplane` et l'espace de noms est `upbound-system`. Upbound diffuse le module complémentaire.

Pour plus d'informations sur le module complémentaire, consultez Upbound Universal Crossplane (UXP) dans la documentation Upbound.

Nom du compte de service

Aucun compte de service n'est utilisé avec ce module complémentaire.

AWS politique IAM gérée

Aucune politique gérée n'est utilisée avec ce module complémentaire.

Autorisations IAM personnalisées

Les autorisations personnalisées ne sont pas utilisées avec ce module complémentaire.

Upwind

Le nom du module complémentaire est `upwind` et l'espace de noms est `upwind`. Upwind diffuse le module complémentaire.

Pour plus d'informations sur le module complémentaire, consultez la documentation Upwind.

Nom du compte de service

Aucun compte de service n'est utilisé avec ce module complémentaire.

AWS politique IAM gérée

Aucune politique gérée n'est utilisée avec ce module complémentaire.

Autorisations IAM personnalisées

Les autorisations personnalisées ne sont pas utilisées avec ce module complémentaire.

Créer un module complémentaire Amazon EKS

Les modules complémentaires Amazon EKS sont des logiciels complémentaires destinés aux clusters Amazon EKS. Tous les modules complémentaires Amazon EKS :

- Inclure les derniers correctifs de sécurité et corrections de bogues.
- Sont validés par AWS pour fonctionner avec Amazon EKS.
- Réduire la charge de travail nécessaire à la gestion du logiciel complémentaire.

Vous pouvez créer un module complémentaire Amazon EKS à l'aide `eksctl` de l'interface de AWS Management Console ligne de commande ou de la AWS CLI. Si le module complémentaire nécessite un rôle IAM, consultez les détails du module complémentaire spécifique dans Amazon EKS add-ons pour plus d'informations sur la création du rôle.

Conditions préalables

Avant de créer un module complémentaire, procédez comme suit :

- Le cluster doit exister avant que vous ne puissiez créer un module complémentaire pour celui-ci. Pour de plus amples informations, veuillez consulter the section called "Créer un cluster".
- Vérifiez si votre module complémentaire nécessite un rôle IAM. Pour de plus amples informations, veuillez consulter the section called "Vérification de la compatibilité".
- Vérifiez que la version du module complémentaire Amazon EKS est compatible avec votre cluster. Pour de plus amples informations, veuillez consulter the section called "Vérification de la compatibilité".
- Vérifiez que la version 0.190.0 ou ultérieure de l'outil de ligne de `eksctl` commande est installée sur votre ordinateur ou. AWS CloudShell Pour plus d'informations, consultez Installation sur le site Web `eksctl`.

Procédure

Vous pouvez créer un module complémentaire Amazon EKS à l'aide `eksctl` de l'interface de AWS Management Console ligne de commande ou de la AWS CLI. Si le module complémentaire nécessite un rôle IAM, consultez les détails du module complémentaire spécifique dans Modules complémentaires Amazon EKS disponibles AWS pour plus d'informations sur la création du rôle.

Créer un module complémentaire (eksctl)

1. Affichez les noms des modules complémentaires disponibles pour une version de cluster. Remplacez *1.33* par la version de votre cluster.

```
eksctl utils describe-addon-versions --kubernetes-version 1.33 | grep AddonName
```

L'exemple qui suit illustre un résultat.

```
"AddonName": "aws-ebs-csi-driver",
                        "AddonName": "coredns",
                        "AddonName": "kube-proxy",
                        "AddonName": "vpc-cni",
                        "AddonName": "adot",
                        "AddonName": "dynatrace_dynatrace-operator",
                        "AddonName": "upbound_universal-crossplane",
```

```
                        "AddonName": "teleport_teleport",
                        "AddonName": "factorhouse_kpow",
                        [...]
```

2. Consultez les versions disponibles pour le module complémentaire que vous souhaitez créer. Remplacez *1.33* par la version de votre cluster. Remplacez *name-of-addon* par le nom du module complémentaire dont vous souhaitez afficher les versions. Le nom doit être l'un des noms renvoyés à l'étape précédente.

```
eksctl utils describe-addon-versions --kubernetes-version 1.33 --name name-of-addon |
 grep AddonVersion
```

La sortie suivante est un exemple de ce qui est renvoyé pour le module complémentaire nommé `vpc-cni`. Vous pouvez voir que le module complémentaire dispose de plusieurs versions disponibles.

```
"AddonVersions": [
    "AddonVersion": "v1.12.0-eksbuild.1",
    "AddonVersion": "v1.11.4-eksbuild.1",
    "AddonVersion": "v1.10.4-eksbuild.1",
    "AddonVersion": "v1.9.3-eksbuild.1",
```

a. Déterminez si le module complémentaire que vous souhaitez créer est un module complémentaire Amazon EKS ou AWS Marketplace. The AWS Marketplace propose des modules complémentaires tiers qui vous obligent à effectuer des étapes supplémentaires pour créer le module complémentaire.

```
eksctl utils describe-addon-versions --kubernetes-version 1.33 --name name-of-
addon | grep ProductUrl
```

Si aucune sortie n'est renvoyée, le module complémentaire est un module Amazon EKS. Si une sortie est renvoyée, le module complémentaire est un module complémentaire AWS Marketplace. La sortie suivante concerne un module complémentaire nommé `teleport_teleport`.

```
"ProductUrl": "https://aws.amazon.com/marketplace/pp?sku=3bda70bb-566f-4976-806c-
f96faef18b26"
```

Vous pouvez en savoir plus sur le module complémentaire sur le AWS Marketplace avec l'URL renvoyée. Si le module complémentaire nécessite un abonnement, vous pouvez vous y abonner via le AWS Marketplace. Si vous souhaitez créer un module complémentaire depuis AWS Marketplace, le principal IAM que vous utilisez pour créer le module complémentaire doit être autorisé à créer le rôle lié à un AWSServiceRoleForAWSLicenseManagerRoleservice. Pour plus d'informations sur l'attribution d'autorisations à une entité IAM, consultez la rubrique Ajout et suppression d'autorisations basées sur l'identité IAM dans le Guide de l'utilisateur IAM.

3. Créez un module complémentaire Amazon EKS. Copiez la commande et remplacez la commande *user-data* comme suit :
   - Remplacez *my-cluster* par le nom de votre cluster.
   - Remplacez *name-of-addon* par le nom du module complémentaire que vous souhaitez créer.
   - Si vous souhaitez une version du module complémentaire antérieure à la dernière version, remplacez-la *latest* par le numéro de version renvoyé dans le résultat de l'étape précédente que vous souhaitez utiliser.
   - Si le module complémentaire utilise un rôle de compte de service, remplacez *111122223333* par votre ID de compte et *role-name* par le nom du rôle. Pour obtenir des instructions sur la création d'un rôle pour votre compte de service, consultez la documentation relative au module complémentaire que vous créez. Pour obtenir la liste des modules complémentaires, consultez the section called " AWS modules complémentaires". Pour spécifier un rôle de compte de service, vous devez disposer d'un fournisseur IAM OpenID Connect (OIDC) pour votre cluster. Pour déterminer si vous en avez un pour votre cluster ou pour en créer un, consultez the section called "Fournisseur IAM OIDC".

     Si le module complémentaire n'utilise pas de rôle de compte de service, supprimez `--service-account-role-arn arn:aws: iam::111122223333:role/role-name`.
   - Cet exemple de commande remplace la configuration de toute version autogérée existante du module complémentaire, le cas échéant. Si vous ne souhaitez pas remplacer la configuration d'un module complémentaire autogéré existant, supprimez l'*--force*option. Si vous supprimez cette option et que le module complémentaire doit remplacer la configuration d'un module complémentaire autogéré existant, la création du module complémentaire Amazon EKS échoue et vous recevez un message d'erreur pour vous aider à résoudre le conflit. Avant de spécifier cette option, assurez-vous que le module complémentaire Amazon EKS ne gère pas les paramètres que vous devez gérer, car ces paramètres sont remplacés par cette option.

```
eksctl create addon --cluster my-cluster --name name-of-addon --version latest \
```

```
    --service-account-role-arn arn:aws: iam::111122223333:role/role-name --force
```

Vous pouvez consulter la liste complète des options disponibles pour la commande.

```
eksctl create addon --help
```

Pour plus d'informations sur les options disponibles, consultez Modules complémentaires dans la documentation `eksctl`.

Créer un module complémentaire (AWS console)

1. Ouvrez la console Amazon EKS.
2. Dans le volet de navigation de gauche, choisissez Clusters.
3. Choisissez le nom du cluster pour lequel vous souhaitez créer le module complémentaire.
4. Choisissez l'onglet Modules complémentaires.
5. Choisissez Obtenez plus de modules complémentaires.
6. Sur la page Select add-ons (Sélectionner les modules complémentaires), choisissez les modules complémentaires que vous souhaitez ajouter à votre cluster. Vous pouvez ajouter autant de modules complémentaires Amazon EKS et de modules complémentaires AWS Marketplace que vous le souhaitez.

   Pour les modules complémentaires AWS Marketplace, le principal IAM que vous utilisez pour créer le module complémentaire doit être autorisé à lire les droits relatifs au module complémentaire depuis le. AWS LicenseManager AWS LicenseManager nécessite un rôle AWSServiceRoleForAWSLicenseManagerRolelié au service (SLR) qui permet aux AWS ressources de gérer les licences en votre nom. Le SLR est une exigence unique, par compte, et vous n'aurez pas à créer de SLR distincts pour chaque module complémentaire ou chaque cluster. Pour plus d'informations sur l'attribution d'autorisations à un principal IAM, consultez la rubrique Ajout et suppression d'autorisations basées sur l'identité IAM du Guide de l'utilisateur IAM.

   Si les modules complémentaires AWS Marketplace que vous souhaitez installer ne figurent pas dans la liste, vous pouvez cliquer sur la numérotation des pages pour afficher des résultats supplémentaires ou effectuer une recherche dans le champ de recherche. Dans les Options de filtrage, vous pouvez également filtrer par catégorie, fournisseur, ou modèle de tarification puis choisir les modules complémentaires dans les résultats de la recherche. Une fois que vous avez sélectionné les modules complémentaires que vous souhaitez installer, cliquez sur Suivant.

7. Sur la page Configurer les paramètres des modules complémentaires sélectionnés, procédez comme suit :

   a. Choisissez Afficher les options d'abonnement pour ouvrir le formulaire des Options d'abonnement. Passez en revue les sections Détails de la tarification et Mentions légales, puis cliquez sur le bouton S'abonner pour continuer.

   b. Pour la version, choisissez celle que vous souhaitez installer. Nous recommandons la version marquée comme la plus récente, à moins que le module complémentaire que vous créez ne recommande une version différente. Pour déterminer si un module complémentaire dispose d'une version recommandée, consultez la documentation relative au module complémentaire que vous créez. Pour obtenir la liste des modules complémentaires, consultez the section called " AWS modules complémentaires".

   c. Vous disposez de deux options pour configurer les rôles des modules complémentaires : le rôle IAM EKS Pod Identities et les rôles IAM pour les comptes de service (IRSA). Suivez l'étape appropriée ci-dessous pour l'option que vous préférez. Si tous les modules complémentaires que vous avez sélectionnés ont la mention Nécessite un abonnement sous Statut, sélectionnez Suivant. Vous ne pouvez pas configurer davantage ces modules complémentaires tant que vous ne vous y êtes pas abonné après la création de votre cluster. Pour les modules complémentaires qui n'ont pas la mention Abonnement requis sous Statut, procédez comme suit :

      i. Pour le rôle IAM d'identité du pod pour le compte de service, vous pouvez soit utiliser un rôle IAM d'identité du pod EKS existant, soit en créer un à l'aide du bouton Créer un rôle recommandé. Ce champ ne proposera que les options correspondant à la politique de confiance appropriée. Si aucun rôle n'est sélectionnable, cela signifie qu'aucun rôle existant ne correspond à la politique de confiance. Pour configurer un rôle IAM d'identité du pod EKS pour les comptes de service du module complémentaire sélectionné, choisissez Créer un rôle recommandé. L'assistant de création de rôle s'ouvre dans une fenêtre séparée. L'assistant remplira automatiquement les informations relatives au rôle comme suit. Pour chaque module complémentaire pour lequel vous souhaitez créer le rôle IAM d'identité du pod EKS, suivez les étapes de l'assistant IAM comme suit.

         - À l'étape Sélectionner une entité de confiance, l'option de AWS service pour EKS et le cas d'utilisation pour EKS - Pod Identity sont présélectionnés, et la politique de confiance appropriée sera automatiquement renseignée pour le module complémentaire. Par exemple, le rôle sera créé avec la politique de confiance appropriée contenant le principal IAM pods.eks.amazonaws.com, comme détaillé dans the section called "Avantages des identités du pod EKS". Choisissez Suivant.

- À l'étape Ajouter des autorisations, la stratégie gérée appropriée pour la stratégie de rôle est présélectionnée pour le module complémentaire. Par exemple, pour le module complémentaire Amazon VPC CNI, le rôle sera créé avec la politique gérée `AmazonEKS_CNI_Policy` telle que détaillée dans the section called "Plug-in CNI Amazon VPC pour Kubernetes". Choisissez Suivant.
- À l'étape Nom, révision et création, dans Nom du rôle, le nom de rôle par défaut est automatiquement renseigné pour le module complémentaire. Par exemple, pour le module complémentaire Amazon VPC CNI, le rôle sera créé sous le nom Amazon. EKSPod IdentityAmazon VPCCNIRole Dans Description, la description par défaut est automatiquement remplie avec la description appropriée pour le module complémentaire. Par exemple, pour le module complémentaire Amazon VPC CNI, le rôle sera créé avec la description Autorise les pods exécutés dans le cluster Amazon EKS à accéder aux ressources. AWS Dans la stratégie de confiance, affichez la stratégie de confiance renseignée pour le module complémentaire. Choisissez Créer un rôle.

  REMARQUE : Conserver le nom de rôle par défaut permet à EKS de présélectionner le rôle pour les modules complémentaires dans les nouveaux clusters ou lors de l'ajout de modules complémentaires à des clusters existants. Vous pouvez toujours remplacer ce nom et le rôle sera disponible pour le module complémentaire dans tous vos clusters, mais il devra être sélectionné manuellement dans le menu déroulant.

ii. Pour les modules complémentaires qui ne comportent pas la mention Abonnement requis sous Statut et pour lesquels vous souhaitez configurer des rôles à l'aide d'IRSA, consultez la documentation relative au module complémentaire que vous créez afin de créer une politique IAM et de l'associer à un rôle. Pour obtenir la liste des modules complémentaires, consultez the section called " AWS modules complémentaires". Pour sélectionner un rôle IAM, vous devez disposer d'un fournisseur IAM OpenID Connect (OIDC) pour votre cluster. Pour déterminer si vous en avez un pour votre cluster ou pour en créer un, consultez the section called "Fournisseur IAM OIDC".

iii. Sélectionnez Optional configuration settings (Paramètres de configuration facultatifs).

iv. Si le module complémentaire nécessite une configuration, saisissez-la dans la zone Configuration values (Valeurs de configuration). Pour déterminer si le module complémentaire nécessite des informations de configuration, consultez la documentation du module complémentaire que vous créez. Pour obtenir la liste des modules complémentaires, consultez the section called " AWS modules complémentaires".

v. Choisissez l'une des options disponibles pour la méthode de résolution des conflits. Si vous sélectionnez Remplacer comme méthode de résolution des conflits, un ou plusieurs

paramètres du module complémentaire existant peuvent être remplacés par les paramètres du module complémentaire Amazon EKS. Si vous n'activez pas cette option et qu'il y a un conflit avec vos paramètres existants, l'opération échoue. Vous pouvez utiliser le message d'erreur qui en résulte pour résoudre le conflit. Avant de sélectionner cette option, assurez-vous que le module complémentaire Amazon EKS ne gère pas les paramètres que vous devez gérer vous-même.

vi. Si vous souhaitez installer le module complémentaire dans un espace de noms spécifique, saisissez-le dans le champ Espace de noms. Pour les modules complémentaires communautaires AWS et ceux de la communauté, vous pouvez définir un espace de noms Kubernetes personnalisé dans lequel installer le module complémentaire. Pour de plus amples informations, veuillez consulter the section called "Espace de noms personnalisé pour les modules complémentaires".

vii. Choisissez Suivant.

8. Sur la page Vérifier et ajouter, choisissez Créer. Une fois l'installation des modules complémentaire terminée, vous pouvez voir les modules complémentaires installés.
9. Si l'un des modules complémentaires que vous avez installés nécessite un abonnement, procédez comme suit :
   a. Cliquez sur le bouton Subscribe (S'abonner) dans le coin inférieur droit du module complémentaire. Vous êtes redirigé vers la page du module complémentaire AWS sur le Marketplace. Lisez les informations relatives au module complémentaire, telles que Product Overview (Présentation du produit) et Pricing Information (Informations sur la tarification).
   b. Cliquez sur le bouton Continue to Subscribe (Continuer pour s'abonner) en haut à droite de la page du module complémentaire.
   c. Lisez les Conditions générales. Si vous les acceptez, sélectionnez Accept Terms (Accepter les conditions). Le traitement de l'abonnement peut prendre plusieurs minutes. Pendant le traitement de l'abonnement, le bouton Return to Amazon EKS Console (Retourner à la console Amazon EKS) est grisé.
   d. Une fois le traitement de l'abonnement terminé, le bouton Return to Amazon EKS Console (Retourner à la console Amazon EKS) n'est plus grisé. Cliquez sur le bouton pour revenir à l'onglet Add-ons (Modules complémentaires) de la console Amazon EKS pour votre cluster.
   e. Pour le module complémentaire auquel vous êtes abonné, sélectionnez Remove and reinstall (Supprimer et réinstaller), puis Reinstall add-on (Réinstaller le module complémentaire). L'installation du module complémentaire peut prendre plusieurs minutes. Une fois l'installation terminée, vous pouvez configurer le module complémentaire.

Créer un module complémentaire (AWS CLI)

1. Vous avez besoin d'`2.12.3`une version ou d'une version ultérieure `1.27.160` ou d'une version ultérieure de l'interface de ligne de AWS commande (AWS CLI) installée et configurée sur votre appareil ou AWS CloudShell. Pour vérifier votre version actuelle, utilisez `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Les gestionnaires de packages tels que `yum` Homebrew pour macOS ont souvent plusieurs versions de retard sur la dernière version de la AWS CLI. `apt-get` Pour installer la dernière version, consultez la section Installation et configuration rapide avec aws configure dans le Guide de l'utilisateur de l'interface de ligne de AWS commande. La version de la AWS CLI installée AWS CloudShell peut également avoir plusieurs versions de retard par rapport à la dernière version. Pour le mettre à jour, consultez la section Installation de la AWS CLI dans votre répertoire de base dans le guide de AWS CloudShell l'utilisateur.
2. Déterminez quels modules complémentaires sont disponibles. Vous pouvez voir tous les modules complémentaires disponibles, leur type et leur éditeur. Vous pouvez également consulter l'URL des modules complémentaires disponibles AWS sur le Marketplace. Remplacez *`1.33`* par la version de votre cluster.

```
aws eks describe-addon-versions --kubernetes-version 1.33 \
    --query 'addons[].{MarketplaceProductUrl: marketplaceInformation.productUrl,
 Name: addonName, Owner: owner Publisher: publisher, Type: type}' --output table
```

L'exemple qui suit illustre un résultat.

```
-------------------------------------------------------------------------------------------
|
 DescribeAddonVersions
  |
+-----------------------------------------------------------
+------------------------------+-----------------+-------------
+---------------------+
|                     MarketplaceProductUrl                     |            Name
          |      Owner       |  Publisher   |        Type         |
+-----------------------------------------------------------
+------------------------------+-----------------+-------------
+---------------------+
|  None                                                         |  aws-ebs-csi-driver
          |  aws             |  eks         |  storage            |
|  None                                                         |  coredns
          |  aws             |  eks         |  networking         |
```

```
|  None                                                                |  kube-proxy
           |  aws             |  eks         |  networking          |
|  None                                                                |  vpc-cni
           |  aws             |  eks         |  networking          |
|  None                                                                |  adot
           |  aws             |  eks         |  observability       |
| https://aws.amazon.com/marketplace/pp/prodview-brb73nceicv7u |
 dynatrace_dynatrace-operator |  aws-marketplace |  dynatrace   |  monitoring
 |
| https://aws.amazon.com/marketplace/pp/prodview-uhc2iwi5xysoc |  upbound_universal-
crossplane |  aws-marketplace |  upbound     |  infra-management   |
| https://aws.amazon.com/marketplace/pp/prodview-hd2ydsrgqy4li |  teleport_teleport
          |  aws-marketplace |  teleport    |  policy-management  |
| https://aws.amazon.com/marketplace/pp/prodview-vgghgqdsplhvc |  factorhouse_kpow
          |  aws-marketplace |  factorhouse |  monitoring         |
|  [...]                                                               |  [...]
           |  [...]           |  [...]       |  [...]               |
+------------------------------------------------------------------
+-------------------------------+------------------+--------------
+---------------------+
```

Votre sortie peut être différente. Dans cet exemple de sortie, trois modules complémentaires différents sont disponibles par type `networking` et cinq modules complémentaires avec un éditeur de type `eks`. Les modules complémentaires `aws-marketplace` figurant dans la colonne `0wner` peuvent nécessiter un abonnement avant de pouvoir être installés. Vous pouvez consulter l'URL pour en savoir plus sur le module complémentaire et vous y abonner.

3. Vous pouvez voir quelles versions sont disponibles pour chaque module complémentaire. Remplacez *`1.33`* par la version de votre cluster et remplacez *`vpc-cni`* par le nom d'un module complémentaire renvoyé à l'étape précédente.

```
aws eks describe-addon-versions --kubernetes-version 1.33 --addon-name vpc-cni \
    --query 'addons[].addonVersions[].{Version: addonVersion, Defaultversion:
 compatibilities[0].defaultVersion}' --output table
```

L'exemple qui suit illustre un résultat.

```
-----------------------------------------
|          DescribeAddonVersions         |
+-----------------+----------------------+
| Defaultversion  |       Version        |
+-----------------+----------------------+
```

```
|  False          |  v1.12.0-eksbuild.1  |
|  True           |  v1.11.4-eksbuild.1  |
|  False          |  v1.10.4-eksbuild.1  |
|  False          |  v1.9.3-eksbuild.1   |
+-----------------+----------------------+
```

La version avec `True` dans la colonne `Defaultversion` correspond à la version avec laquelle le module complémentaire est créé, par défaut.

4. (Facultatif) Recherchez les options de configuration du module complémentaire de votre choix en exécutant la commande suivante :

```
aws eks describe-addon-configuration --addon-name vpc-cni --addon-version v1.12.0-
eksbuild.1
```

```
{
    "addonName": "vpc-cni",
    "addonVersion": "v1.12.0-eksbuild.1",
    "configurationSchema": "{\"$ref\":\"#/definitions/VpcCni\",\"$schema\":\"http://
json-schema.org/draft-06/schema#\",\"definitions\":{\"Cri\":{\"additionalProperties
\":false,\"properties\":{\"hostPath\":{\"$ref\":\"#/definitions/HostPath\"}},\"title
\":\"Cri\",\"type\":\"object\"},\"Env\":{\"additionalProperties\":false,\"properties
\":{\"ADDITIONAL_ENI_TAGS\":{\"type\":\"string\"},\"AWS_VPC_CNI_NODE_PORT_SUPPORT
\":{\"format\":\"boolean\",\"type\":\"string\"},\"AWS_VPC_ENI_MTU\":{\"format\":
\"integer\",\"type\":\"string\"},\"AWS_VPC_K8S_CNI_CONFIGURE_RPFILTER\":{\"format
\":\"boolean\",\"type\":\"string\"},\"AWS_VPC_K8S_CNI_CUSTOM_NETWORK_CFG\":{\"format
\":\"boolean\",\"type\":\"string\"},\"AWS_VPC_K8S_CNI_EXTERNALSNAT\":{\"format\":
\"boolean\",\"type\":\"string\"},\"AWS_VPC_K8S_CNI_LOGLEVEL\":{\"type\":\"string\"},
\"AWS_VPC_K8S_CNI_LOG_FILE\":{\"type\":\"string\"},\"AWS_VPC_K8S_CNI_RANDOMIZESNAT
\":{\"type\":\"string\"},\"AWS_VPC_K8S_CNI_VETHPREFIX\":{\"type\":\"string\"},
\"AWS_VPC_K8S_PLUGIN_LOG_FILE\":{\"type\":\"string\"},\"AWS_VPC_K8S_PLUGIN_LOG_LEVEL
\":{\"type\":\"string\"},\"DISABLE_INTROSPECTION\":{\"format\":\"boolean\",\"type
\":\"string\"},\"DISABLE_METRICS\":{\"format\":\"boolean\",\"type\":\"string
\"},\"DISABLE_NETWORK_RESOURCE_PROVISIONING\":{\"format\":\"boolean\",\"type
\":\"string\"},\"ENABLE_POD_ENI\":{\"format\":\"boolean\",\"type\":\"string
\"},\"ENABLE_PREFIX_DELEGATION\":{\"format\":\"boolean\",\"type\":\"string\"},
\"WARM_ENI_TARGET\":{\"format\":\"integer\",\"type\":\"string\"},\"WARM_PREFIX_TARGET
\":{\"format\":\"integer\",\"type\":\"string\"}},\"title\":\"Env\",\"type\":
\"object\"},\"HostPath\":{\"additionalProperties\":false,\"properties\":{\"path
\":{\"type\":\"string\"}},\"title\":\"HostPath\",\"type\":\"object\"},\"Limits
\":{\"additionalProperties\":false,\"properties\":{\"cpu\":{\"type\":\"string
\"},\"memory\":{\"type\":\"string\"}},\"title\":\"Limits\",\"type\":\"object\"},
```

```
\"Resources\":{\"additionalProperties\":false,\"properties\":{\"limits\":{\"$ref\":
\"#/definitions/Limits\"},\"requests\":{\"$ref\":\"#/definitions/Limits\"}},\"title
\":\"Resources\",\"type\":\"object\"},\"VpcCni\":{\"additionalProperties\":false,
\"properties\":{\"cri\":{\"$ref\":\"#/definitions/Cri\"},\"env\":{\"$ref\":\"#/
definitions/Env\"},\"resources\":{\"$ref\":\"#/definitions/Resources\"}},\"title\":
\"VpcCni\",\"type\":\"object\"}}}"
}
```

La sortie correspond à un schéma JSON standard.

Voici un exemple de valeurs de configuration valides, au format JSON, fonctionnant avec le schéma ci-dessus.

```
{
  "resources": {
    "limits": {
      "cpu": "100m"
    }
  }
}
```

Voici un exemple de valeurs de configuration valides, au format YAML, fonctionnant avec le schéma ci-dessus.

```
  resources:
    limits:
      cpu: 100m
```

5. Déterminez si le module complémentaire nécessite des autorisations IAM. Si tel est le cas, vous devez (1) déterminer si vous souhaitez utiliser les identités de pod EKS ou les rôles IAM pour les comptes de service (IRSA), (2) déterminer l'ARN du rôle IAM à utiliser avec le module complémentaire et (3) déterminer le nom du compte de service Kubernetes utilisé par le module complémentaire. Pour de plus amples informations, veuillez consulter the section called "Récupérer les informations IAM".
   - Amazon EKS recommande d'utiliser les identités de pod EKS si le module complémentaire le prend en charge. Pour cela, l'agent d'identité du pod doit être installé sur votre cluster. Pour plus d'informations sur l'utilisation des identités de pod avec les modules complémentaires, consultez the section called "Rôles IAM".

- Si le module complémentaire ou votre cluster n'est pas configuré pour les identités de pod EKS, utilisez IRSA. Vérifiez que l'IRSA est configuré sur votre cluster.
- Consultez la documentation relative aux modules complémentaires Amazon EKS afin de déterminer si le module complémentaire nécessite des autorisations IAM et le nom du compte de service Kubernetes associé.

  a. Créez un module complémentaire Amazon EKS. Copiez la commande qui suit sur votre appareil. Si nécessaire, apportez les modifications suivantes à la commande, puis exécutez la commande modifiée :

- Remplacez *my-cluster* par le nom de votre cluster.
- Remplacez *vpc-cni* par un nom de module complémentaire renvoyé dans le résultat de l'étape précédente que vous souhaitez créer.
- Remplacez *version-number* par la version renvoyée dans le résultat de l'étape précédente que vous souhaitez utiliser.
- Si vous souhaitez installer le module complémentaire dans un espace de noms Kubernetes personnalisé, ajoutez l'option `--namespace-config 'namespace=<my-namespace>`. Cette option n'est disponible que pour AWS les modules complémentaires communautaires. Pour de plus amples informations, consultez the section called "Espace de noms personnalisé pour les modules complémentaires".
- Si le module complémentaire ne nécessite pas d'autorisations IAM, *<service-account-configuration>* supprimez-le.
- Effectuez l'une des actions suivantes :
  - Si le module complémentaire (1) nécessite des autorisations IAM et (2) votre cluster utilise EKS Pod Identities, remplacez-le par *<service-account-configuration>* l'association d'identité de pod suivante. *<service-account-name>*Remplacez-le par le nom du compte de service utilisé par le module complémentaire. Remplacez *<role-arn>* par l'ARN d'un rôle IAM. Le rôle doit disposer de la politique de confiance requise par les identités EKS Pod.

    ```
    --pod-identity-associations 'serviceAccount=<service-account-name>,roleArn=<role-arn>'
    ```

  - Si le module complémentaire (1) nécessite des autorisations IAM et (2) votre cluster utilise IRSA, remplacez-le *<service-account-configuration>* par la configuration IRSA suivante. *111122223333*Remplacez-le par votre identifiant de compte et *role-name* par le nom d'un rôle IAM existant que vous avez créé. Pour obtenir des instructions sur la création d'un rôle, consultez la documentation du module complémentaire que vous créez. Pour

obtenir la liste des modules complémentaires, consultez the section called " AWS modules complémentaires". Pour spécifier un rôle de compte de service, vous devez disposer d'un fournisseur IAM OpenID Connect (OIDC) pour votre cluster. Pour déterminer si vous en avez un pour votre cluster ou pour en créer un, consultez the section called "Fournisseur IAM OIDC".

```
--service-account-role-arn arn:aws::iam::111122223333:role/role-name
```

- Cet exemple de commande remplace l'option `--configuration-values` de toute version autogérée existante du module complémentaire, le cas échéant. Remplacez cela par les valeurs de configuration souhaitées, telles qu'une chaîne ou une entrée de fichier. Si vous ne voulez pas fournir de valeurs de configuration, supprimez l'option `--configuration-values`. Si vous ne souhaitez pas que la AWS CLI remplace la configuration d'un module complémentaire autogéré existant, supprimez l'*`--resolve-conflicts OVERWRITE`*option. Si vous supprimez cette option et que le module complémentaire doit remplacer la configuration d'un module complémentaire autogéré existant, la création du module complémentaire Amazon EKS échoue et vous recevez un message d'erreur pour vous aider à résoudre le conflit. Avant de spécifier cette option, assurez-vous que le module complémentaire Amazon EKS ne gère pas les paramètres que vous devez gérer, car ces paramètres sont remplacés par cette option.

```
aws eks create-addon --cluster-name my-cluster --addon-name vpc-cni --addon-version
 version-number \
    <service-account-configuration> --configuration-values '{"resources":
{"limits":{"cpu":"100m"}}}' --resolve-conflicts OVERWRITE
```

```
aws eks create-addon --cluster-name my-cluster --addon-name vpc-cni --addon-version
 version-number \
   <service-account-configuration> --configuration-values 'file://example.yaml' --
resolve-conflicts OVERWRITE
```

Pour la liste complète des options disponibles, consultez `create-addon` dans la référence des lignes de commande Amazon EKS. Si le module complémentaire que vous avez créé possède `aws-marketplace` listé dans la colonne `Owner` d'une étape précédente, la création peut échouer et vous pouvez recevoir un message d'erreur similaire à l'erreur suivante.

```
{
    "addon": {
        "addonName": "addon-name",
```

```
        "clusterName": "my-cluster",
        "status": "CREATE_FAILED",
        "addonVersion": "version",
        "health": {
            "issues": [
                {
                    "code": "AddonSubscriptionNeeded",
                    "message": "You are currently not subscribed to this add-on. To
 subscribe, visit the AWS Marketplace console, agree to the seller EULA, select the
 pricing type if required, then re-install the add-on"
                }
            ]
        }
    }
}
```

Si vous recevez une erreur similaire à celle de la sortie précédente, consultez l'URL figurant dans la sortie d'une étape précédente pour vous abonner au module complémentaire. Une fois abonné, exécutez à nouveau la commande `create-addon`.

Mettre à jour un module complémentaire Amazon EKS

Amazon EKS ne met pas automatiquement à jour un module complémentaire lorsque de nouvelles versions sont publiées ou après la mise à jour de votre cluster vers une nouvelle version mineure de Kubernetes. Pour mettre à jour un module complémentaire pour un cluster existant, vous devez lancer la mise à jour. Ensuite, Amazon EKS met à jour le module complémentaire Amazon EKS pour vous. Avant de mettre à jour un module complémentaire, consultez la documentation relative au module complémentaire. Pour obtenir la liste des modules complémentaires disponibles, consultez the section called " AWS modules complémentaires". Si le module complémentaire nécessite un rôle IAM, consultez les détails du module complémentaire spécifique dans Modules complémentaires Amazon EKS disponibles AWS pour plus d'informations sur la création du rôle.

Conditions préalables

Avant de créer un module complémentaire, procédez comme suit :

- Vérifiez si votre module complémentaire nécessite un rôle IAM. Pour de plus amples informations, veuillez consulter the section called "Modules complémentaires Amazon EKS".

- Vérifiez que la version du module complémentaire Amazon EKS est compatible avec votre cluster. Pour de plus amples informations, veuillez consulter the section called "Vérification de la compatibilité".

Procédure

Vous pouvez mettre à jour un module complémentaire Amazon EKS à l'aide `eksctl` de l'interface de AWS Management Console ligne de commande ou de la AWS CLI.

Mettre à jour le module complémentaire (eksctl)

1. Déterminez les modules complémentaires et les versions de modules complémentaires actuellement installés sur votre cluster. Remplacez *`my-cluster`* par le nom de votre cluster.

   ```
   eksctl get addon --cluster my-cluster
   ```

   L'exemple qui suit illustre un résultat.

   ```
   NAME        VERSION              STATUS  ISSUES  IAMROLE  UPDATE AVAILABLE
   coredns     v1.8.7-eksbuild.2    ACTIVE  0
   kube-proxy  v1.23.7-eksbuild.1   ACTIVE  0                v1.23.8-eksbuild.2
   vpc-cni     v1.10.4-eksbuild.1   ACTIVE  0                v1.12.0-eksbuild.1,v1.11.4-
   eksbuild.1,v1.11.3-eksbuild.1,v1.11.2-eksbuild.1,v1.11.0-eksbuild.1
   ```

   Votre sortie peut être différente, en fonction des modules complémentaires et des versions dont vous disposez sur votre cluster. Vous pouvez voir que dans l'exemple de sortie précédent, deux modules complémentaires existants sur le cluster disposent de versions plus récentes disponibles dans la colonne `UPDATE AVAILABLE`.
2. Mettez à jour le module complémentaire.
   a. Copiez la commande qui suit sur votre appareil. Si nécessaire, apportez les modifications suivantes à la commande :
      - Remplacez *`my-cluster`* par le nom de votre cluster.
      - Remplacez *`region-code`* par la AWS région dans laquelle se trouve votre cluster.
      - Remplacez *`vpc-cni`* par le nom d'un module complémentaire renvoyé dans la sortie de l'étape précédente que vous souhaitez mettre à jour.
      - Si vous souhaitez procéder à une mise à jour vers une version antérieure à la dernière version disponible, remplacez *`latest`* par le numéro de version renvoyé dans la sortie de

l'étape précédente que vous souhaitez utiliser. Certains modules complémentaires présentent des versions recommandées. Pour plus d'informations, consultez la documentation relative au module complémentaire que vous mettez à jour. Pour obtenir la liste des modules complémentaires, consultezthe section called " AWS modules complémentaires". * Si le module complémentaire utilise un compte de service Kubernetes et un rôle IAM, remplacez-les par *111122223333* votre identifiant de compte et *role-name* par le nom d'un rôle IAM existant que vous avez créé. Pour obtenir des instructions sur la création d'un rôle, consultez la documentation du module complémentaire que vous créez. Pour obtenir la liste des modules complémentaires, consultez the section called " AWS modules complémentaires". Pour spécifier un rôle de compte de service, vous devez disposer d'un fournisseur IAM OpenID Connect (OIDC) pour votre cluster. Pour déterminer si vous en avez un pour votre cluster ou pour en créer un, consultez the section called "Fournisseur IAM OIDC".

Si le module complémentaire n'utilise pas de compte de service Kubernetes ni de rôle IAM, supprimez la ligne `serviceAccountRoleARN: arn:aws:iam::`*111122223333*`:role/`*role-name* .

- L'option *preserve* conserve les valeurs existantes pour le module complémentaire. Si vous avez défini des valeurs personnalisées pour les paramètres du module complémentaire et que vous n'utilisez pas cette option, Amazon EKS remplace vos valeurs par ses valeurs par défaut. Si vous utilisez cette option, nous vous recommandons de tester les modifications de champ et de valeur sur un cluster hors production avant de mettre à jour le module complémentaire sur votre cluster de production. Si vous remplacez cette valeur par `overwrite`, tous les paramètres sont remplacés par les valeurs par défaut d'Amazon EKS. Si vous avez défini des valeurs personnalisées pour certains paramètres, il est possible qu'elles soient remplacées par les valeurs par défaut d'Amazon EKS. Si vous remplacez cette valeur par `none`, Amazon EKS ne modifie la valeur d'aucun paramètre, mais la mise à jour risque d'échouer. Si la mise à jour échoue, vous recevez un message d'erreur pour vous aider à résoudre le conflit.

```
cat >update-addon.yaml <<EOF
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
metadata:
  name: my-cluster
  region: region-code

addons:
- name: vpc-cni
```

```
  version: latest
  serviceAccountRoleARN: arn:aws: iam::111122223333:role/role-name
  resolveConflicts: preserve
EOF
```

b. Exécutez la commande modifiée pour créer le fichier `update-addon.yaml`.

c. Appliquez le fichier de configuration à votre cluster.

```
eksctl update addon -f update-addon.yaml
```

Pour plus d'informations sur la mise à jour des modules complémentaires, consultez Mise à jour des modules complémentaires dans la documentation `eksctl`.

Mettre à jour le module complémentaire (AWS console)

1. Ouvrez la console Amazon EKS.
2. Dans le volet de navigation de gauche, choisissez Clusters.
3. Sélectionnez le nom du cluster que vous voulez mettre à jour le module complémentaire.
4. Choisissez l'onglet Modules complémentaires.
5. Sélectionnez le module complémentaire que vous voulez mettre à jour.
6. Choisissez Modifier.
7. Sur la *name of addon* page Configurer, procédez comme suit :
   a. Sélectionnez la Version que vous voulez utiliser. Le module complémentaire peut présenter une version recommandée. Pour plus d'informations, consultez la documentation relative au module complémentaire que vous mettez à jour. Pour obtenir la liste des modules complémentaires, consultez the section called " AWS modules complémentaires".
   b. Vous disposez de deux options pour configurer les rôles des modules complémentaires : le rôle IAM EKS Pod Identities et les rôles IAM pour les comptes de service (IRSA). Suivez l'étape appropriée ci-dessous pour l'option que vous préférez. Si tous les modules complémentaires que vous avez sélectionnés ont la mention Nécessite un abonnement sous Statut, sélectionnez Suivant. Pour les modules complémentaires qui n'ont pas la mention Abonnement requis sous Statut, procédez comme suit :
      i. Pour le rôle IAM d'identité du pod pour le compte de service, vous pouvez soit utiliser un rôle IAM d'identité du pod EKS existant, soit en créer un à l'aide du bouton Créer un rôle recommandé. Ce champ ne proposera que les options correspondant à la politique de

confiance appropriée. Si aucun rôle n'est sélectionnable, cela signifie qu'aucun rôle existant ne correspond à la politique de confiance. Pour configurer un rôle IAM d'identité du pod EKS pour les comptes de service du module complémentaire sélectionné, choisissez Créer un rôle recommandé. L'assistant de création de rôle s'ouvre dans une fenêtre séparée. L'assistant remplira automatiquement les informations relatives au rôle comme suit. Pour chaque module complémentaire pour lequel vous souhaitez créer le rôle IAM d'identité du pod EKS, suivez les étapes de l'assistant IAM comme suit.

- À l'étape Sélectionner une entité de confiance, l'option de AWS service pour EKS et le cas d'utilisation pour EKS - Pod Identity sont présélectionnés, et la politique de confiance appropriée sera automatiquement renseignée pour le module complémentaire. Par exemple, le rôle sera créé avec la politique de confiance appropriée contenant le principal IAM pods.eks.amazonaws.com, comme détaillé dans the section called "Avantages des identités du pod EKS". Choisissez Suivant.
- À l'étape Ajouter des autorisations, la stratégie gérée appropriée pour la stratégie de rôle est présélectionnée pour le module complémentaire. Par exemple, pour le module complémentaire Amazon VPC CNI, le rôle sera créé avec la politique gérée `AmazonEKS_CNI_Policy` telle que détaillée dans the section called "Plug-in CNI Amazon VPC pour Kubernetes". Choisissez Suivant.
- À l'étape Nom, révision et création, dans Nom du rôle, le nom de rôle par défaut est automatiquement renseigné pour le module complémentaire. Par exemple, pour le module complémentaire Amazon VPC CNI, le rôle sera créé sous le nom Amazon. EKSPod IdentityAmazon VPCCNIRole Dans Description, la description par défaut est automatiquement remplie avec la description appropriée pour le module complémentaire. Par exemple, pour le module complémentaire Amazon VPC CNI, le rôle sera créé avec la description Autorise les pods exécutés dans le cluster Amazon EKS à accéder aux ressources. AWS Dans la stratégie de confiance, affichez la stratégie de confiance renseignée pour le module complémentaire. Choisissez Créer un rôle.

> **Note**
>
> En conservant le nom de rôle par défaut, EKS peut présélectionner le rôle pour les modules complémentaires dans les nouveaux clusters ou lors de l'ajout de modules complémentaires à des clusters existants. Vous pouvez toujours remplacer ce nom et le rôle sera disponible pour le module complémentaire dans tous vos clusters, mais il devra être sélectionné manuellement dans le menu déroulant.

ii. Pour les modules complémentaires qui ne comportent pas la mention Abonnement requis sous Statut et pour lesquels vous souhaitez configurer des rôles à l'aide d'IRSA, consultez la documentation relative au module complémentaire que vous créez afin de créer une politique IAM et de l'associer à un rôle. Pour obtenir la liste des modules complémentaires, consultez the section called " AWS modules complémentaires". Pour sélectionner un rôle IAM, vous devez disposer d'un fournisseur IAM OpenID Connect (OIDC) pour votre cluster. Pour déterminer si vous en avez un pour votre cluster ou pour en créer un, consultez the section called "Fournisseur IAM OIDC".

c. Sélectionnez Paramètres de configuration facultatifs.

d. Dans Valeurs de configuration, saisissez toutes les informations de configuration spécifiques au module complémentaire. Pour plus d'informations, consultez la documentation relative au module complémentaire que vous mettez à jour. Pour obtenir la liste des modules complémentaires, consultez the section called " AWS modules complémentaires"… Pour Méthode de résolution des conflits, sélectionnez l'une des options. Si vous avez défini des valeurs personnalisées pour les paramètres des modules complémentaires, nous vous recommandons l'option Preserve (Conserver). Si vous ne sélectionnez pas cette option, Amazon EKS remplace vos valeurs par ses valeurs par défaut. Si vous utilisez cette option, nous vous recommandons de tester les modifications de champ et de valeur sur un cluster hors production avant de mettre à jour le module complémentaire sur votre cluster de production. Si vous modifiez cette valeur pour la remplacer, tous les paramètres sont remplacés par les valeurs par défaut d'Amazon EKS. Si vous avez défini des valeurs personnalisées pour certains paramètres, il est possible qu'elles soient remplacées par les valeurs par défaut d'Amazon EKS. Si vous modifiez cette valeur pour « none », Amazon EKS ne modifie la valeur d'aucun paramètre, mais la mise à jour peut échouer. Si la mise à jour échoue, vous recevez un message d'erreur pour vous aider à résoudre le conflit.

8. Sélectionnez Enregistrer les modifications.

Module complémentaire de mise à jour (AWS CLI)

1. Vous avez besoin d'`2.12.3`une version ou d'une version ultérieure `1.27.160` ou d'une version ultérieure de l'interface de ligne de AWS commande (AWS CLI) installée et configurée sur votre appareil ou AWS CloudShell. Pour vérifier votre version actuelle, utilisez `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Les gestionnaires de packages tels que `yum` Homebrew pour macOS ont souvent plusieurs versions de retard sur la dernière version de la AWS CLI. `apt-get` Pour installer la dernière version, consultez la section Installation et configuration rapide avec aws

configure dans le Guide de l'utilisateur de l'interface de ligne de AWS commande. La version de la AWS CLI installée AWS CloudShell peut également avoir plusieurs versions de retard par rapport à la dernière version. Pour le mettre à jour, consultez la section Installation de la AWS CLI dans votre répertoire de base dans le guide de AWS CloudShell l'utilisateur.

2. Consultez la liste des modules complémentaires installés. Remplacez *my-cluster* par le nom de votre cluster.

```
aws eks list-addons --cluster-name my-cluster
```

L'exemple qui suit illustre un résultat.

```
{
    "addons": [
        "coredns",
        "kube-proxy",
        "vpc-cni"
    ]
}
```

3. Affichez la version actuelle du module complémentaire que vous souhaitez mettre à jour. Remplacez *my-cluster* par le nom de votre cluster et *vpc-cni* par le nom du module complémentaire que vous souhaitez mettre à jour.

```
aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni --query
 "addon.addonVersion" --output text
```

L'exemple qui suit illustre un résultat.

```
v1.10.4-eksbuild.1
```

4. Déterminez quelles versions du module complémentaire sont disponibles pour la version de votre cluster. *1.33*Remplacez-le par la version de votre cluster et *vpc-cni* par le nom du module complémentaire que vous souhaitez mettre à jour.

```
aws eks describe-addon-versions --kubernetes-version 1.33 --addon-name vpc-cni \
    --query 'addons[].addonVersions[].{Version: addonVersion, Defaultversion:
 compatibilities[0].defaultVersion}' --output table
```

L'exemple qui suit illustre un résultat.

```
-----------------------------------------
|          DescribeAddonVersions         |
+-----------------+----------------------+
| Defaultversion  |       Version        |
+-----------------+----------------------+
|  False          |  v1.12.0-eksbuild.1  |
|  True           |  v1.11.4-eksbuild.1  |
|  False          |  v1.10.4-eksbuild.1  |
|  False          |  v1.9.3-eksbuild.1   |
+-----------------+----------------------+
```

La version avec `True` dans la colonne `Defaultversion` correspond à la version avec laquelle le module complémentaire est créé, par défaut.

5. Mettez à jour votre module complémentaire. Copiez la commande qui suit sur votre appareil. Si nécessaire, apportez les modifications suivantes à la commande, puis exécutez la commande modifiée. Pour plus d'informations sur cette commande, consultez update-addon dans la référence des commandes en ligne Amazon EKS.
   - Remplacez *`my-cluster`* par le nom de votre cluster.
   - Remplacez *`vpc-cni`* par le nom du module complémentaire que vous souhaitez mettre à jour renvoyé dans la sortie d'une étape précédente.
   - Remplacez *`version-number`* par la version renvoyée dans la sortie de l'étape précédente vers laquelle vous souhaitez effectuer la mise à jour. Certains modules complémentaires présentent des versions recommandées. Pour plus d'informations, consultez la documentation relative au module complémentaire que vous mettez à jour. Pour obtenir la liste des modules complémentaires, consultezthe section called " AWS modules complémentaires". * Si le module complémentaire utilise un compte de service Kubernetes et un rôle IAM, remplacez-les par *`111122223333`* votre identifiant de compte et *`role-name`* par le nom d'un rôle IAM existant que vous avez créé. Pour obtenir des instructions sur la création d'un rôle, consultez la documentation du module complémentaire que vous créez. Pour obtenir la liste des modules complémentaires, consultez the section called " AWS modules complémentaires". Pour spécifier un rôle de compte de service, vous devez disposer d'un fournisseur IAM OpenID Connect (OIDC) pour votre cluster. Pour déterminer si vous en avez un pour votre cluster ou pour en créer un, consultez the section called "Fournisseur IAM OIDC".

Si le module complémentaire n'utilise pas de compte de service Kubernetes ni de rôle IAM, supprimez la ligne `serviceAccountRoleARN: arn:aws: iam::`*`111122223333`*`:role/`*`role-name`* .

- L'option `--resolve-conflicts PRESERVE` conserve les valeurs existantes pour le module complémentaire. Si vous avez défini des valeurs personnalisées pour les paramètres du module complémentaire et que vous n'utilisez pas cette option, Amazon EKS remplace vos valeurs par ses valeurs par défaut. Si vous utilisez cette option, nous vous recommandons de tester les modifications de champ et de valeur sur un cluster hors production avant de mettre à jour le module complémentaire sur votre cluster de production. Si vous remplacez cette valeur par `OVERWRITE`, tous les paramètres sont remplacés par les valeurs par défaut d'Amazon EKS. Si vous avez défini des valeurs personnalisées pour certains paramètres, il est possible qu'elles soient remplacées par les valeurs par défaut d'Amazon EKS. Si vous remplacez cette valeur par `NONE`, Amazon EKS ne modifie la valeur d'aucun paramètre, mais la mise à jour risque d'échouer. Si la mise à jour échoue, vous recevez un message d'erreur pour vous aider à résoudre le conflit.
- Si vous voulez supprimer toutes les configurations personnalisées, effectuez la mise à jour à l'aide de l'option *`--configuration-values '{}'`*. Cela ramène toutes les configurations personnalisées aux valeurs par défaut. Si vous ne souhaitez pas modifier votre configuration personnalisée, ne fournissez pas le *`--configuration-values`* drapeau. Si vous voulez ajuster une configuration personnalisée, remplacez *`{}`* par les nouveaux paramètres.

```
aws eks update-addon --cluster-name my-cluster --addon-name vpc-cni --addon-version
 version-number \
    --service-account-role-arn arn:aws: iam::111122223333:role/role-name --
configuration-values '{}' --resolve-conflicts PRESERVE
```

6. Vérifiez l'état de la mise à jour. *`my-cluster`*Remplacez-le par le nom de votre cluster et *`vpc-cni`* par le nom du module complémentaire que vous mettez à jour.

```
aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni
```

L'exemple qui suit illustre un résultat.

```
{
    "addon": {
        "addonName": "vpc-cni",
        "clusterName": "my-cluster",
```

```
        "status": "UPDATING",
    }
}
```

Lorsque la mise à jour est terminé, l'état indique `ACTIVE`.

Vérification de la compatibilité de la version du module complémentaire Amazon EKS avec un cluster

Avant de créer un module complémentaire Amazon EKS, vous devez vérifier que sa version est compatible avec votre cluster.

Utilisez l'API describe-addon-versions pour répertorier les versions disponibles des modules complémentaires EKS et connaître les versions de Kubernetes prises en charge par chaque version du module complémentaire.

1. Vérifiez que la CLI AWS est installée et fonctionne avec `aws sts get-caller-identity`. Si cette commande ne fonctionne pas, consultez la section Démarrer avec la CLI AWS.
2. Déterminez le nom du module complémentaire pour lequel vous souhaitez obtenir des informations de compatibilité, par exemple `amazon-cloudwatch-observability`.
3. Déterminez la version Kubernetes de votre cluster, par exemple `1.33`.
4. Utilisez la CLI AWS pour extraire les versions des modules complémentaires compatibles avec la version Kubernetes de votre cluster.

```
aws eks describe-addon-versions --addon-name amazon-cloudwatch-observability --
kubernetes-version 1.33
```

L'exemple qui suit illustre un résultat.

```
{
    "addons": [
        {
            "addonName": "amazon-cloudwatch-observability",
            "type": "observability",
            "addonVersions": [
                {
                    "addonVersion": "vX.X.X-eksbuild.X",
                    "architecture": [
```

```
                        "amd64",
                        "arm64"
                    ],
                    "computeTypes": [
                        "ec2",
                        "auto",
                        "hybrid"
                    ],
                    "compatibilities": [
                        {
                            "clusterVersion": "1.33",
                            "platformVersions": [
                                "*"
                            ],
                            "defaultVersion": true
                        }
                    ],
                }
            ]
        }
    ]
}
```

Ce résultat indique que la version `vX.X.X-eksbuild.X` du module complémentaire est compatible avec la version `1.33` du cluster Kubernetes.

Compatibilité des modules complémentaires avec les types de calcul

Le champ `computeTypes` du résultat `describe-addon-versions` indique la compatibilité du module complémentaire avec les nœuds gérés par le mode automatique EKS ou les nœuds hybrides. Les modules complémentaires marqués `auto` fonctionnent avec l'infrastructure cloud en mode automatique EKS gérée par AWS. Ceux marqués `hybrid` peuvent fonctionner sur des nœuds sur site connectés au plan de contrôle cloud d'EKS.

Pour de plus amples informations, consultez the section called "Considérations relatives au mode automatique Amazon EKS".

Supprimer un module complémentaire Amazon EKS d'un cluster

Vous pouvez supprimer un module complémentaire Amazon EKS de votre cluster à l'aide `eksctl` de la AWS Management Console CLI ou de la AWS CLI.

Lorsque vous supprimez un module complémentaire Amazon EKS d'un cluster :

- Il n'y a pas de temps d'arrêt pour les fonctionnalités fournies par le module complémentaire.
- Si vous utilisez des rôles IAM pour les comptes de service (IRSA) et que le module complémentaire est associé à un rôle IAM, ce dernier n'est pas supprimé.
- Si vous utilisez des identités du pod, toutes les associations d'identités du pod appartenant au module complémentaire sont supprimées. Si vous spécifiez l'`--preserve`option dans la AWS CLI, les associations sont préservées.
- Amazon EKS cesse de gérer les paramètres du module complémentaire.
- La console cesse de vous avertir lorsque de nouvelles versions sont disponibles.
- Vous ne pouvez pas mettre à jour le module complémentaire à l'aide d' AWS outils ou APIs.
- Vous pouvez choisir de laisser le module complémentaire sur votre cluster afin de pouvoir le gérer automatiquement, ou le supprimer de votre cluster. Vous ne devez supprimer le module complémentaire de votre cluster que si aucune ressource sur votre cluster ne dépend de la fonctionnalité fournie par ce module complémentaire.

Conditions préalables

Avant de créer un module complémentaire, procédez comme suit :

- Un cluster Amazon EKS existant. Pour en déployer un, consultez Démarrer.
- Vérifiez si votre module complémentaire nécessite un rôle IAM. Pour plus d'informations, veuillez consulter la rubrique
- Version `0.215.0` ou ultérieure de l'outil de ligne de commande `eksctl` installée sur votre appareil ou AWS CloudShell. Pour installer ou mettre à jour `eksctl`, consultez Installation dans la documentation `eksctl`.

Procédure

Vous disposez de deux options pour supprimer un module complémentaire Amazon EKS.

- Conserver le logiciel complémentaire sur votre cluster : cette option supprime la gestion de tous les paramètres par Amazon EKS. Elle supprime également la possibilité pour Amazon EKS de vous informer des mises à jour et de mettre automatiquement à jour le module complémentaire Amazon EKS après le lancement d'une mise à jour. Toutefois, elle préserve le logiciel complémentaire sur votre cluster. Cette option fait du module complémentaire une installation autogérée plutôt qu'un

module complémentaire Amazon EKS. Avec cette option, il n'y a pas de durée d'indisponibilité pour le module complémentaire.

- Supprimer entièrement le logiciel complémentaire de votre cluster : nous vous recommandons de supprimer le module complémentaire Amazon EKS de votre cluster uniquement si aucune ressource de votre cluster n'en dépend.

Vous pouvez supprimer un module complémentaire Amazon EKS à l'aide `eksctl` de l'interface de AWS Management Console ligne de commande ou de la AWS CLI.

Supprimer le module complémentaire (eksctl)

1. Déterminez les modules complémentaires actuellement installés sur votre cluster. Remplacez *my-cluster* par le nom de votre cluster.

   ```
   eksctl get addon --cluster my-cluster
   ```

   L'exemple qui suit illustre un résultat.

   ```
   NAME        VERSION              STATUS  ISSUES  IAMROLE  UPDATE AVAILABLE
   coredns     v1.8.7-eksbuild.2    ACTIVE  0
   kube-proxy  v1.23.7-eksbuild.1   ACTIVE  0
   vpc-cni     v1.10.4-eksbuild.1   ACTIVE  0
   [...]
   ```

   Votre sortie peut être différente, en fonction des modules complémentaires et des versions dont vous disposez sur votre cluster.

2. Supprimer le module complémentaire. Remplacez *my-cluster* par le nom de votre cluster et *name-of-add-on* par le nom du module complémentaire renvoyé dans la sortie de l'étape précédente que vous souhaitez supprimer. Si vous supprimez *--preserve* cette option, en plus du fait qu'Amazon EKS ne gère plus le module complémentaire, le logiciel complémentaire est supprimé de votre cluster.

   ```
   eksctl delete addon --cluster my-cluster --name name-of-addon --preserve
   ```

   Pour plus d'informations sur la suppression des modules complémentaires, consultez Suppression de modules complémentaires dans la documentation `eksctl`.

Supprimer le module complémentaire (AWS console)

1. Ouvrez la console Amazon EKS.
2. Dans le volet de navigation de gauche, choisissez Clusters.
3. Sélectionnez le nom du cluster pour lequel vous voulez supprimer le module complémentaire Amazon EKS.
4. Choisissez l'onglet Modules complémentaires.
5. Sélectionnez le module complémentaire que vous voulez supprimer.
6. Cliquez sur Supprimer.
7. Dans la boîte de dialogue Supprimer : ***name of addon*** confirmation, procédez comme suit :
   a. Si vous voulez qu'Amazon EKS cesse de gérer les paramètres du module complémentaire, sélectionnez Conserver sur le cluster. Faites ceci si vous voulez retenir le logiciel du module complémentaire sur votre cluster. Ceci afin que vous puissiez gérer vous-même tous les paramètres du module complémentaire.
   b. Saisissez le nom du module complémentaire.
   c. Cliquez sur Supprimer.

Supprimer le module complémentaire (AWS CLI)

1. Vous avez besoin d'`0.215.0`une version ou d'une version ultérieure de l'outil de ligne de `eksctl` commande installée sur votre appareil ou AWS CloudShell. Pour installer ou mettre à jour `eksctl`, veuillez consulter Installation dans la documentation de `eksctl`.
2. Consultez la liste des modules complémentaires installés. Remplacez *`my-cluster`* par le nom de votre cluster.

   ```
   aws eks list-addons --cluster-name my-cluster
   ```

   L'exemple qui suit illustre un résultat.

   ```
   {
       "addons": [
           "coredns",
           "kube-proxy",
           "vpc-cni",
           "name-of-addon"
       ]
   ```

```
}
```

3. Supprimez le module complémentaire installé. Remplacez *my-cluster* par le nom de votre cluster et *name-of-add-on* par le nom du module complémentaire que vous souhaitez supprimer. La suppression *--preserve* supprime le logiciel complémentaire de votre cluster.

```
aws eks delete-addon --cluster-name my-cluster --addon-name name-of-addon --preserve
```

L'exemple suivant illustre le résultat abrégé.

```
{
    "addon": {
        "addonName": "name-of-add-on",
        "clusterName": "my-cluster",
        "status": "DELETING",
    }
}
```

4. Vérifiez l'état de la suppression. *my-cluster*Remplacez-le par le nom de votre cluster et *name-of-addon* par le nom du module complémentaire que vous supprimez.

```
aws eks describe-addon --cluster-name my-cluster --addon-name name-of-addon
```

Une fois le module complémentaire supprimé, l'exemple de sortie est le suivant.

```
An error occurred (ResourceNotFoundException) when calling the DescribeAddon
 operation: No addon: name-of-addon found in cluster: my-cluster
```

Rôles IAM pour les modules complémentaires Amazon EKS

Certains modules complémentaires Amazon EKS nécessitent des rôles IAM et des autorisations pour appeler des API AWS. Par exemple, le module complémentaire Amazon VPC CNI appelle certaines API AWS afin de configurer des ressources réseau dans votre compte. Ces modules complémentaires doivent obtenir des autorisations via IAM. Plus précisément, le compte de service du pod exécutant le module complémentaire doit être associé à un rôle IAM doté d'une politique IAM spécifique.

La méthode recommandée pour accorder des autorisations AWS aux charges de travail du cluster consiste à utiliser la fonctionnalité Amazon EKS d'identité du pod. Vous pouvez utiliser une

association d'identité du pod pour mapper le compte de service d'un module complémentaire à un rôle IAM. Si un pod utilise un compte de service qui a une association, Amazon EKS définit des variables d'environnement dans les conteneurs du pod. Les variables d'environnement configurent les kits SDK AWS, y compris la CLI AWS, pour utiliser les informations d'identification d'identité du pod EKS. Pour plus d'informations, consultez the section called "Pod Identity".

Les modules complémentaires Amazon EKS peuvent aider à gérer le cycle de vie des associations d'identité de pod liées à un module complémentaire. Par exemple, vous pouvez créer ou mettre à jour un module complémentaire Amazon EKS et l'association d'identité de pod correspondante en un seul appel d'API. Amazon EKS fournit aussi une API permettant d'extraire les politiques IAM recommandées.

1. Vérifiez que l'agent d'identité du pod Amazon EKS est configuré sur votre cluster.
2. Déterminez si le module complémentaire que vous souhaitez installer nécessite des autorisations IAM en utilisant l'opération `describe-addon-versions` de la CLI AWS. Si le paramètre `requiresIamPermissions` est défini sur `true`, utilisez l'opération `describe-addon-configurations` pour connaître les autorisations nécessaires au module complémentaire. La réponse inclut une liste de politiques IAM gérées recommandées.
3. Extrayez le nom du compte de service Kubernetes et de la politique IAM à l'aide de l'opération `describe-addon-configuration` de la CLI. Évaluez la portée de la politique recommandée au regard de vos exigences de sécurité.
4. Créez un rôle IAM utilisant la politique d'autorisation recommandée ainsi que la politique d'approbation requise par l'identité du pod. Pour de plus amples informations, consultez the section called "Création d'une association Pod Identity (AWS console)".
5. Créez ou mettez à jour un module complémentaire Amazon EKS à l'aide de la CLI. Spécifiez au moins une association d'identité de pod. Une association d'identité de pod correspond au nom d'un compte de service Kubernetes et à l'ARN du rôle IAM.
   - Les associations d'identité de pod créées via les API des modules complémentaires sont gérées par le module complémentaire concerné. Si vous supprimez le module complémentaire, l'association d'identité de pod est également supprimée. Vous pouvez éviter cette suppression en cascade en utilisant l'option `preserve` lors de la suppression d'un module complémentaire via la CLI AWS ou l'API. Vous pouvez également mettre à jour ou supprimer directement l'association d'identité de pod si nécessaire. Les modules complémentaires ne peuvent pas prendre la propriété d'associations d'identité de pod déjà existantes. Vous devez supprimer l'association existante, puis la recréer via une opération de création ou de mise à jour du module complémentaire.

- Amazon EKS recommande d'utiliser les associations d'identité de pod pour gérer les autorisations IAM des modules complémentaires. La méthode précédente, les rôles IAM pour les comptes de service (IRSA), est toujours prise en charge. Vous pouvez spécifier à la fois un `serviceAccountRoleArn` IRSA et une association d'identité de pod pour un module complémentaire. Si l'agent d'identité de pod EKS est installé sur le cluster, le `serviceAccountRoleArn` est ignoré et EKS utilise l'association d'identité de pod fournie. Si l'identité du pod n'est pas activée, le `serviceAccountRoleArn` est utilisé.
- Si vous mettez à jour les associations d'identité de pod d'un module complémentaire existant, Amazon EKS lance un redémarrage progressif des pods du module complémentaire.

Récupérer les informations IAM sur un module complémentaire Amazon EKS

Avant de créer un module complémentaire, utilisez l'AWS CLI pour déterminer :

- Si le module complémentaire nécessite des autorisations IAM
- La politique IAM suggérée à utiliser

Procédure

1. Déterminez le nom du module complémentaire que vous voulez installer et la version Kubernetes de votre cluster. Pour plus d'informations sur les modules complémentaires, voir the section called "Modules complémentaires Amazon EKS".
2. Utilisez l'interface AWS CLI pour déterminer si le module complémentaire nécessite des autorisations IAM.

```
aws eks describe-addon-versions \
--addon-name <addon-name> \
--kubernetes-version <kubernetes-version>
```

Par exemple :

```
aws eks describe-addon-versions \
--addon-name aws-ebs-csi-driver \
--kubernetes-version 1.30
```

Veuillez examiner l'exemple de résultat suivant. Notez que `requiresIamPermissions` est `true` et qu'il s'agit de la version par défaut du module complémentaire. Vous devez spécifier la version du module complémentaire lorsque vous récupérez la politique IAM recommandée.

```
{
    "addons": [
        {
            "addonName": "aws-ebs-csi-driver",
            "type": "storage",
            "addonVersions": [
                {
                    "addonVersion": "v1.31.0-eksbuild.1",
                    "architecture": [
                        "amd64",
                        "arm64"
                    ],
                    "compatibilities": [
                        {
                            "clusterVersion": "1.30",
                            "platformVersions": [
                                "*"
                            ],
                            "defaultVersion": true
                        }
                    ],
                    "requiresConfiguration": false,
                    "requiresIamPermissions": true
                },
[...]
```

3. Si le module complémentaire nécessite des autorisations IAM, utilisez l'AWS CLI pour récupérer une stratégie IAM recommandée.

```
aws eks describe-addon-configuration \
--query podIdentityConfiguration \
--addon-name <addon-name> \
--addon-version <addon-version>
```

Par exemple :

```
aws eks describe-addon-configuration \
```

```
--query podIdentityConfiguration \
--addon-name aws-ebs-csi-driver \
--addon-version v1.31.0-eksbuild.1
```

Veuillez examiner l'exemple suivant. Notez le `recommendedManagedPolicies`.

```
[
    {
        "serviceAccount": "ebs-csi-controller-sa",
        "recommendedManagedPolicies": [
            "arn:aws:iam::aws:policy/service-role/AmazonEBSCSIDriverPolicy"
        ]
    }
]
```

4. Créez un rôle IAM et associez-lui la stratégie gérée recommandée. Vous pouvez également examiner la stratégie gérée et réduire la portée des autorisations si nécessaire. Pour plus d'informations, consultez the section called "Création d'une association Pod Identity (AWS console)".

Référence relative à la prise en charge de l'identité du pod

Le tableau suivant indique si certains modules complémentaires Amazon EKS prennent en charge l'identité du pod EKS.

Nom du module complémentaire	Prise en charge de l'identité du pod	Version minimale requise
Pilote CSI Amazon EBS	Oui	v1.26.0-eksbuild.1
CNI Amazon VPC	Oui	v1.15.5-eksbuild.1
Pilote CSI Amazon EFS	Oui	v2.0.5-eksbuild.1
Distro pour Open Telemetry AWS	Oui	v0.94.1-eksbuild.1
Mountpoint pour Amazon S3 CSI Driver	Non	N/A

Nom du module complémentaire	Prise en charge de l'identité du pod	Version minimale requise
Agent d'observabilité Amazon CloudWatch	Oui	v3.1.0-eksbuild.1

Ce tableau a été mis à jour pour la dernière fois le 28 octobre 2024.

Utiliser les identités du pod pour attribuer un rôle IAM à un module complémentaire Amazon EKS

Certains modules complémentaires Amazon EKS nécessitent des rôles et des autorisations IAM. Avant d'ajouter ou de mettre à jour un module complémentaire Amazon EKS pour utiliser une association d'identité du pod, vérifiez le rôle et la politique à utiliser. Pour de plus amples informations, consultez the section called "Récupérer les informations IAM".

1. Déterminez :
   - `cluster-name` : le nom du cluster sur lequel installer le module complémentaire.
   - `addon-name` : le nom du module complémentaire à installer.
   - `service-account-name` : le nom du compte de service Kubernetes utilisé par le module complémentaire.
   - `iam-role-arn` : l'ARN d'un rôle IAM disposant des autorisations suffisantes pour le module complémentaire. Le rôle doit disposer de la politique d'approbation requise pour l'identité du pod EKS. Pour plus d'informations, consultez the section called "Création d'une association Pod Identity (AWS console)".
2. Mettez à jour le module complémentaire à l'aide de l'interface AWS CLI. Vous pouvez également spécifier des associations d'identité du pod lors de la création d'un module complémentaire, en utilisant la même syntaxe `--pod-identity-assocations`. Notez que lorsque vous spécifiez des associations d'identité du pod lors de la mise à jour d'un module complémentaire, toutes les associations d'identité du pod précédentes sont écrasées.

```
aws eks update-addon --cluster-name <cluster-name> \
--addon-name <addon-name> \
--pod-identity-associations 'serviceAccount=<service-account-name>,roleArn=<role-
arn>'
```

Par exemple :

```
aws eks update-addon --cluster-name mycluster \
--addon-name aws-ebs-csi-driver \
--pod-identity-associations 'serviceAccount=ebs-csi-controller-
sa,roleArn=arn:aws:iam::123456789012:role/StorageDriver'
```

3. Vérifiez que l'association d'identité du pod a été créée :

```
aws eks list-pod-identity-associations --cluster-name <cluster-name>
```

Si la commande aboutit, vous devriez obtenir une sortie similaire à ce qui suit. Notez le OwnerARN du module complémentaire EKS.

```
{
    "associations": [
        {
            "clusterName": "mycluster",
            "namespace": "kube-system",
            "serviceAccount": "ebs-csi-controller-sa",
            "associationArn": "arn:aws:eks:us-
west-2:123456789012:podidentityassociation/mycluster/a-4wvljrezsukshq1bv",
            "associationId": "a-4wvljrezsukshq1bv",
            "ownerArn": "arn:aws:eks:us-west-2:123456789012:addon/mycluster/aws-ebs-
csi-driver/9cc7ce8c-2e15-b0a7-f311-426691cd8546"
        }
    ]
}
```

Supprimer les associations d'identité du pod d'un module complémentaire Amazon EKS

Supprimez les associations d'identité du pod EKS d'un module complémentaire Amazon EKS.

1. Déterminez :
   - `cluster-name` : le nom du cluster EKS sur lequel installer le module complémentaire.
   - `addon-name` : le nom du module complémentaire Amazon EKS à installer.

2. Mettez à jour le module complémentaire pour spécifier un tableau vide d'associations d'identités de pod.

```
aws eks update-addon --cluster-name <cluster-name> \
--addon-name <addon-name> \
--pod-identity-associations "[]"
```

Résolution des problèmes relatifs aux identités des pods pour les modules complémentaires EKS

Si vos modules complémentaires rencontrent des erreurs lors des opérations des API, des kits SDK ou de la CLI AWS, vérifiez les points suivants :

- L'agent d'identité du pod est installé dans votre cluster.
  - Pour plus d'informations sur l'installation de l'agent d'identité du pod, consultez the section called "Configurer l'agent".
- Le module complémentaire dispose d'une association d'identité de pod valide.
  - Utilisez la CLI AWS pour extraire les associations créées pour le nom du compte de service utilisé par le module complémentaire.

```
aws eks list-pod-identity-associations --cluster-name <cluster-name>
```

- Le rôle IAM contient la politique d'approbation requise pour les identités des pods.
  - Utilisez la CLI AWS pour extraire la politique d'approbation d'un module complémentaire.

```
aws iam get-role --role-name <role-name> --query Role.AssumeRolePolicyDocument
```

- Le rôle IAM dispose des autorisations nécessaires pour le module complémentaire.
  - Utilisez AWS CloudTrail pour passer en revue les événements `AccessDenied` ou `UnauthorizedOperation`.
- Le nom du compte de service indiqué dans l'association d'identité de pod correspond au nom du compte de service utilisé par le module complémentaire.
  - Pour plus d'informations sur les modules complémentaires disponibles, consultez the section called " AWS modules complémentaires".
- Vérifiez la configuration de MutatingWebHookConfiguration nommé `pod-identity-webhook`
  - `admissionReviewVersions` du webhook doit être `v1beta1` et ne fonctionne pas avec `v1`.

Déterminez les champs que vous pouvez personnaliser pour les modules complémentaires Amazon EKS

Les modules complémentaires Amazon EKS sont installés sur votre cluster à l'aide des configurations standard de bonnes pratiques. Pour plus d'informations sur l'ajout d'un module complémentaire Amazon EKS à votre cluster, consultez the section called “Modules complémentaires Amazon EKS”.

Vous voudrez peut-être personnaliser la configuration d'un module complémentaire Amazon EKS pour activer les fonctions avancées. Amazon EKS utilise la fonction apply côté serveur de Kubernetes pour permettre la gestion d'un module complémentaire par Amazon EKS sans écraser votre configuration pour les paramètres qui ne sont pas gérés par Amazon EKS. Pour plus d'informations, consultez apply côté serveur dans la documentation Kubernetes. Pour ce faire, Amazon EKS gère un ensemble minimal de champs pour chaque module complémentaire qu'il installe. Vous pouvez, sans problème, modifier tous les champs qui ne sont pas gérés par Amazon EKS ou un autre processus de plan de contrôle Kubernetes tel que `kube-controller-manager`.

> **Important**
>
> La modification d'un champ géré par Amazon EKS empêche Amazon EKS de gérer le module complémentaire. Ce processus peut entraîner l'écrasement de vos modifications lors de la mise à jour d'un module complémentaire.

Syntaxe de gestion de champs

Lorsque vous affichez les détails d'un objet Kubernetes, les champs gérés et non gérés sont renvoyés en sortie. Les champs gérés peuvent être l'un des types suivants :

- Entièrement géré : toutes les clés du champ sont gérées par Amazon EKS. Les modifications apportées à une valeur provoquent un conflit.
- Partiellement géré : certaines clés du champ sont gérées par Amazon EKS. Seules les modifications apportées aux clés explicitement gérées par Amazon EKS provoquent un conflit.

Les deux types de champs sont étiquetés avec `manager: eks`.

Chaque clé est soit un `.` représentant le champ lui-même, qui correspond toujours à un ensemble vide, soit une chaîne qui représente un sous-champ ou un élément. La sortie pour la gestion des champs comprend les types de déclarations suivants :

- `f:`*`name`* , où *`name`* est le nom d'un champ d'une liste.
- `k:`*`keys`* , où se *`keys`* trouve une carte des champs d'un élément de liste.
- `v:`*`value`* , où *`value`* est la valeur exacte au format JSON formaté de l'élément d'une liste.
- `i:`*`index`* , où *`index`* est la position d'un élément dans la liste.

Les parties de sortie suivantes pour le module complémentaire CoreDNS illustrent les déclarations précédentes :

- Champs entièrement gérés : si un champ géré possède un (champ) `f:` spécifié, mais aucune (clé) `k:`, alors le champ entier est géré. Les modifications apportées aux valeurs de ce champ provoquent un conflit.

  Dans la sortie suivante, vous pouvez voir que le conteneur nommé `coredns` est géré par `eks`. Les sous-champs `args`, `image` et `imagePullPolicy` sont également gérés par `eks`. Les modifications apportées aux valeurs de ces champs provoquent un conflit.

  ```
  [...]
  f:containers:
    k:{"name":"coredns"}:
    .: {}
    f:args: {}
    f:image: {}
    f:imagePullPolicy: {}
  [...]
  manager: eks
  [...]
  ```

- Champs partiellement gérés : si une clé gérée a une valeur spécifiée, les clés déclarées sont gérées pour ce champ. La modification des clés spécifiées provoque un conflit.

  Dans la sortie suivante, vous pouvez voir que `eks` gère les volumes `config-volume` et `tmp` définis à l'aide de la clé `name`.

  ```
  [...]
  f:volumes:
  ```

```
  k:{"name":"config-volume"}:
    .: {}
    f:configMap:
      f:items: {}
      f:name: {}
    f:name: {}
  k:{"name":"tmp"}:
    .: {}
    f:name: {}
[...]
manager: eks
[...]
```

- Ajout de clés à des champs partiellement gérés : si seule une valeur de clé spécifique est gérée, vous pouvez ajouter en toute sécurité des clés supplémentaires, telles que des arguments, à un champ sans provoquer de conflit. Si vous ajoutez des clés supplémentaires, assurez-vous d'abord que le champ n'est pas géré. L'ajout ou la modification d'une valeur gérée provoque un conflit.

  Dans la sortie suivante, vous pouvez voir que la clé `name` et le champ `name` sont gérés. L'ajout ou la modification d'un nom de conteneur provoque un conflit avec cette clé gérée.

```
[...]
f:containers:
  k:{"name":"coredns"}:
[...]
    f:name: {}
[...]
manager: eks
[...]
```

Procédure

Vous pouvez utiliser `kubectl` pour voir quels champs sont gérés par Amazon EKS pour tout module complémentaire Amazon EKS.

Vous pouvez, sans problème, modifier tous les champs qui ne sont pas gérés par Amazon EKS ou un autre processus de plan de contrôle Kubernetes tel que `kube-controller-manager`.

1. Déterminez le module complémentaire que vous souhaitez examiner. Pour voir l'ensemble du `deployments` terrain DaemonSets déployé sur votre cluster, consultezthe section called "Accéder aux ressources du cluster".

2. Pour afficher les champs gérés d'un module complémentaire, exécutez la commande suivante :

```
kubectl get type/add-on-name -n add-on-namespace -o yaml
```

Par exemple, vous pouvez voir les champs gérés pour le module complémentaire CoreDNS avec la commande suivante.

```
kubectl get deployment/coredns -n kube-system -o yaml
```

La gestion des champs est répertoriée dans la section suivante dans la sortie renvoyée.

```
[...]
managedFields:
  - apiVersion: apps/v1
    fieldsType: FieldsV1
    fieldsV1:
[...]
```

> **Note**
>
> Si vous ne voyez pas `managedFields` en sortie, ajoutez `--show-managed-fields` à la commande et exécutez-la à nouveau. La version de `kubectl` que vous utilisez détermine si les champs gérés sont retournés par défaut.

Étapes suivantes

Personnalisez les champs qui ne sont pas détenus par AWS le module complémentaire for you.

Valider les signatures des images de conteneur pendant le déploiement

Si vous utilisez Signer AWS et souhaitez vérifier les images de conteneur signées au moment du déploiement, vous pouvez utiliser l'une des solutions suivantes :

- Gatekeeper et Ratify : utilisez Gatekeeper comme contrôleur d'admission et Ratify est configuré avec un plug-in AWS Signer en tant que hook Web pour valider les signatures.

- Kyverno : moteur de politiques Kubernetes configuré avec un plug-in AWS Signer pour valider les signatures.

> **Note**
>
> Avant de vérifier les signatures des images des conteneurs, configurez le référentiel de confiance Notation et la politique de confiance, comme l'exige le contrôleur d'admission que vous avez sélectionné.

Fonctionnalités d'EKS

> **Tip**
>
> Pour commencer : créer une fonctionnalité ACK | Créer une fonctionnalité Argo CD | Créer une fonctionnalité Kro

Amazon EKS Capabilities est un ensemble en couches de fonctionnalités de cluster entièrement gérées qui aident à accélérer la rapidité des développeurs et à se libérer de la complexité liée à la création et à la mise à l'échelle avec Kubernetes. Les fonctionnalités EKS sont des fonctionnalités natives de Kubernetes destinées au déploiement continu déclaratif, à la gestion des AWS ressources, ainsi qu'à la création et à l'orchestration des ressources Kubernetes, le tout entièrement géré par. AWS Grâce aux fonctionnalités d'EKS, vous pouvez vous concentrer davantage sur la création et le dimensionnement de vos charges de travail, en vous déchargeant de la charge opérationnelle liée à ces services de plateforme fondamentaux. AWS Ces fonctionnalités s'exécutent au sein d'EKS plutôt que dans vos clusters, ce qui élimine le besoin d'installer, de maintenir et de dimensionner les composants critiques de la plate-forme sur vos nœuds de travail.

Pour commencer, vous pouvez créer une ou plusieurs fonctionnalités EKS sur un cluster EKS nouveau ou existant. Pour ce faire, vous pouvez utiliser la AWS CLI, EKS AWS Management Console APIs, eksctl ou vos outils préférés infrastructure-as-code. Bien que les fonctionnalités EKS soient conçues pour fonctionner ensemble, il s'agit de ressources cloud indépendantes que vous pouvez sélectionner en fonction de votre cas d'utilisation et de vos exigences.

Toutes les versions de Kubernetes prises en charge par EKS sont prises en charge pour les fonctionnalités EKS.

> **Note**
>
> Les fonctionnalités EKS sont disponibles dans toutes les régions AWS commerciales où Amazon EKS est disponible. Pour obtenir la liste des régions prises en charge, consultez la section Points de terminaison et quotas Amazon EKS dans le manuel de référence AWS général.

Fonctionnalités disponibles

AWS Contrôleurs pour Kubernetes (ACK)

ACK permet de gérer les AWS ressources à l'aide de Kubernetes APIs, ce qui vous permet de créer et de gérer des compartiments S3, des bases de données RDS, des rôles IAM et d'autres AWS ressources à l'aide des ressources personnalisées de Kubernetes. ACK concilie en permanence l'état souhaité avec l'état réel AWS, en corrigeant toute dérive au fil du temps afin de maintenir le bon fonctionnement de votre système et la configuration de vos ressources conformément aux spécifications. Vous pouvez gérer les AWS ressources parallèlement à vos charges de travail Kubernetes à l'aide des mêmes outils et flux de travail, avec la prise en charge de plus de 50 AWS services, dont S3, RDS, DynamoDB et Lambda. ACK prend en charge la gestion des ressources entre comptes et entre régions, permettant ainsi des architectures complexes de gestion de systèmes multi-comptes et multi-clusters. ACK prend en charge les ressources en lecture seule et l'adoption en lecture seule, facilitant ainsi la migration depuis d'autres infrastructures sous forme d'outils de code vers vos systèmes basés sur Kubernetes.

En savoir plus sur ACK →

Argo CD

Argo CD implémente un déploiement continu GitOps basé sur le déploiement pour vos applications, en utilisant les référentiels Git comme source fiable pour vos charges de travail et l'état de votre système. Argo CD synchronise automatiquement les ressources des applications avec vos clusters à partir de vos référentiels Git, détectant et corrigeant les dérives pour garantir que les applications déployées correspondent à l'état souhaité. Vous pouvez déployer et gérer des applications sur plusieurs clusters à partir d'une seule instance Argo CD, avec un déploiement automatique à partir des référentiels Git chaque fois que des modifications sont validées. L'utilisation conjointe d'Argo CD et d'ACK fournit un GitOps système de base, simplifiant la gestion des dépendances liées à la charge de travail et prenant en charge les conceptions de systèmes complets, y compris la gestion des clusters et des infrastructures à grande échelle. Argo CD s'intègre à AWS Identity Center pour l'authentification et l'autorisation, et fournit une interface utilisateur Argo hébergée pour visualiser l'état de l'application et l'état de déploiement.

En savoir plus sur Argo CD →

kro (Kube Resource Orchestrator)

kro vous permet de créer des Kubernetes personnalisés APIs qui composent plusieurs ressources en abstractions de haut niveau, permettant aux équipes de la plateforme de définir des modèles réutilisables pour des combinaisons de ressources communes et des éléments de base du cloud. Avec kro, vous pouvez composer Kubernetes et les AWS ressources en abstractions unifiées, en utilisant une syntaxe simple pour permettre des configurations dynamiques et une logique conditionnelle. kro permet aux équipes de plateforme de fournir des fonctionnalités de libre-service avec des garde-fous appropriés, permettant aux développeurs de fournir une infrastructure complexe à l'aide de simples infrastructures spécialement conçues APIs tout en respectant les normes organisationnelles et les meilleures pratiques. les ressources kro sont simplement des ressources Kubernetes et sont spécifiées dans des manifestes Kubernetes qui peuvent être stockés dans Git Git, ou poussé vers OCI- registres compatibles tels qu'Amazon ECR pour une large distribution organisationnelle.

En savoir plus sur Kro →

Avantages des fonctionnalités d'EKS

Les fonctionnalités d'EKS sont entièrement gérées AWS, ce qui élimine le besoin d'installation, de maintenance et de mise à l'échelle des services de cluster de base. AWS gère les correctifs de sécurité, les mises à jour et la gestion opérationnelle, ce qui permet à vos équipes de se concentrer sur le développement AWS plutôt que sur les opérations du cluster. Contrairement aux modules complémentaires Kubernetes traditionnels qui consomment les ressources du cluster, les fonctionnalités s'exécutent dans EKS plutôt que sur vos nœuds de travail. Cela libère de la capacité et des ressources du cluster pour vos charges de travail tout en minimisant la charge opérationnelle liée à la gestion des contrôleurs intégrés au cluster et des autres composants de la plate-forme.

Grâce aux fonctionnalités d'EKS, vous pouvez gérer les déploiements, les AWS ressources, les ressources Kubernetes personnalisées et les compositions à l'aide de Kubernetes natifs et d'outils tels que. APIs `kubectl` Toutes les fonctionnalités fonctionnent dans le contexte de vos clusters, détectant et corrigeant automatiquement les dérives de configuration dans les ressources de l'application et de l'infrastructure cloud. Vous pouvez déployer et gérer des ressources sur plusieurs clusters, AWS comptes et régions à partir d'un point de contrôle unique, ce qui simplifie les opérations dans des environnements complexes et distribués.

Les fonctionnalités EKS sont conçues pour les GitOps flux de travail, fournissant une infrastructure déclarative contrôlée par version et une gestion des applications. Les modifications sont transmises

par Git au système, fournissant des pistes d'audit, des fonctionnalités de restauration et des flux de travail collaboratifs qui s'intègrent à vos pratiques de développement existantes. Cette approche native de Kubernetes signifie que vous n'avez pas besoin d'utiliser plusieurs outils ou de gérer des infrastructure-as-code systèmes externes à vos clusters, et qu'il existe une seule source fiable à laquelle vous pouvez vous référer. L'état souhaité, défini dans la configuration déclarative Kubernetes contrôlée par version, est continuellement appliqué dans l'ensemble de votre environnement.

Tarification

Avec EKS Capabilities, il n'y a aucun engagement initial ni aucun frais minimum. Chaque ressource de fonctionnalité vous est facturée pour chaque heure pendant laquelle elle est active sur votre cluster Amazon EKS. Les ressources Kubernetes spécifiques gérées par EKS Capabilities sont également facturées à un taux horaire.

Pour obtenir les informations tarifaires actuelles, consultez la page de tarification d'Amazon EKS.

> **Tip**
>
> Vous pouvez utiliser AWS Cost Explorer et Cost and Usage Reports pour suivre les coûts de capacité séparément des autres frais EKS. Vous pouvez étiqueter vos capacités avec le nom du cluster, le type de capacité et d'autres informations à des fins de répartition des coûts.

Comment fonctionnent les fonctionnalités d'EKS

Chaque fonctionnalité est une AWS ressource que vous créez sur votre cluster EKS. Une fois créée, la fonctionnalité s'exécute dans EKS et est entièrement gérée par AWS.

> **Note**
>
> Vous pouvez créer une ressource de capacité de chaque type (Argo CD, ACK et kro) pour un cluster donné. Vous ne pouvez pas créer plusieurs ressources de capacité du même type sur le même cluster.

Vous interagissez avec les fonctionnalités de votre cluster à l'aide de Kubernetes APIs et d'outils standard :

- `kubectl`À utiliser pour appliquer des ressources personnalisées Kubernetes
- Utiliser les référentiels Git comme source fiable pour GitOps les flux de travail

Des outils supplémentaires sont pris en charge pour certaines fonctionnalités. Par exemple :

- Utilisez la CLI Argo CD pour configurer et gérer les référentiels et les clusters dans votre fonctionnalité Argo CD
- Utilisez l'interface utilisateur d'Argo CD pour visualiser et gérer les applications gérées par votre fonctionnalité Argo CD

Les fonctionnalités sont conçues pour fonctionner ensemble, mais elles sont indépendantes et entièrement activées. Vous pouvez activer une, deux ou les trois fonctionnalités en fonction de vos besoins, et mettre à jour votre configuration en fonction de l'évolution de vos besoins.

Tous les types de calcul EKS sont compatibles avec les fonctionnalités EKS. Pour de plus amples informations, veuillez consulter Gestion du Calcul.

Pour la configuration de la sécurité et des informations sur les rôles IAM, consultezthe section called "Considérations relatives aux fonctionnalités d'EKS". Pour les modèles d'architecture multi-clusters, voirthe section called "Considérations".

Cas d'utilisation courants

GitOps pour les applications et l'infrastructure

Utilisez Argo CD pour déployer des applications et des composants opérationnels et ACK pour gérer les configurations de clusters et provisionner l'infrastructure, tous deux à partir de référentiels Git. L'ensemble de votre pile (applications, bases de données, stockage et mise en réseau) est défini sous forme de code et automatiquement déployé.

Exemple : une équipe de développement introduit des modifications dans Git. Argo CD déploie l'application mise à jour et ACK approvisionne une nouvelle base de données RDS avec la configuration correcte. Toutes les modifications sont vérifiables, réversibles et cohérentes dans tous les environnements.

Ingénierie de plateforme avec libre-service

Utilisez kro pour créer des ressources personnalisées APIs qui composent les ressources ACK et Kubernetes. Les équipes de la plateforme définissent des modèles approuvés à l'aide de glissières

de sécurité. Les équipes chargées des applications utilisent des outils simples et de haut niveau APIs pour approvisionner des stacks complets.

Exemple : une équipe de plateforme crée une API WebApplication « » qui fournit un compartiment de déploiement, de service, d'entrée et S3. Les développeurs utilisent cette API sans avoir besoin de comprendre la complexité ou AWS les autorisations sous-jacentes.

Gestion des applications multi-clusters

Utilisez Argo CD pour déployer des applications sur plusieurs clusters EKS dans différentes régions ou comptes. Gérez tous les déploiements à partir d'une seule instance Argo CD avec des politiques et des flux de travail cohérents.

Exemple : déployez la même application sur des clusters de développement, de préparation et de production répartis dans plusieurs régions. Argo CD garantit que chaque environnement reste synchronisé avec la branche Git correspondante.

Gestion de plusieurs clusters

Utilisez ACK pour définir et approvisionner des clusters EKS, kro pour personnaliser les configurations de clusters conformément aux normes organisationnelles et Argo CD pour gérer le cycle de vie et la configuration des clusters. Cela permet de gérer les end-to-end clusters depuis leur création jusqu'aux opérations en cours.

Exemple : définissez des clusters EKS à l'aide d'ACK et de kro pour provisionner et gérer l'infrastructure du cluster, en définissant les normes organisationnelles pour le réseau, les politiques de sécurité, les modules complémentaires et autres configurations. Utilisez Argo CD pour créer et gérer en permanence les clusters, les configurations et les mises à jour des versions de Kubernetes au sein de votre flotte en tirant parti de normes cohérentes et d'une gestion automatisée du cycle de vie.

Migrations et modernisation

Simplifiez la migration vers EKS grâce au provisionnement des ressources cloud et à des GitOps flux de travail natifs. Utilisez ACK pour adopter les AWS ressources existantes sans les recréer, et Argo CD pour opérationnaliser les déploiements de charges de travail à partir de Git.

Exemple : une équipe qui migre depuis EKS adopte ses bases de données RDS et ses compartiments S3 existants EC2 à l'aide d'ACK, puis utilise Argo CD pour déployer des applications conteneurisées à partir de Git. La voie de migration est claire et les opérations sont normalisées dès le premier jour.

Démarrage des comptes et des régions

Automatisez le déploiement de l'infrastructure entre les comptes et les régions en utilisant conjointement Argo CD et ACK. Définissez votre infrastructure sous forme de code dans Git et laissez les fonctionnalités s'occuper du déploiement et de la gestion.

Exemple : une équipe de plateforme gère des référentiels Git définissant les configurations de compte standard, les rôles IAMVPCs, les instances RDS et les stacks de surveillance. Argo CD déploie automatiquement ces configurations sur de nouveaux comptes et régions, garantissant ainsi la cohérence et réduisant le temps de configuration manuelle de plusieurs jours à quelques minutes.

Utilisation des ressources en matière de capacités

Cette rubrique décrit les opérations courantes de gestion des ressources de capacité pour tous les types de capacités.

Ressources relatives aux fonctionnalités d'EKS

Les fonctionnalités EKS sont AWS des ressources qui activent les fonctionnalités gérées sur votre cluster Amazon EKS. Les fonctionnalités sont exécutées dans EKS, ce qui élimine le besoin d'installer et de maintenir des contrôleurs et d'autres composants opérationnels sur vos nœuds de travail. Les fonctionnalités sont créées pour un cluster EKS spécifique et restent affiliées à ce cluster pendant tout son cycle de vie.

Chaque ressource de capacité possède :

- Un nom unique au sein de votre cluster
- Un type de capacité (ACK, ARGOCD ou KRO)
- Un Amazon Resource Name (ARN), spécifiant à la fois le nom et le type
- Un rôle IAM axé sur les capacités
- Un statut qui indique son état actuel
- Configuration, à la fois générique et spécifique au type de fonctionnalité

Comprendre l'état des capacités

Les ressources de capacité ont un statut qui indique leur état actuel. Vous pouvez consulter l'état des fonctionnalités et l'état de santé dans la console EKS ou à l'aide de la AWS CLI.

Console :

1. Ouvrez la console Amazon EKS à l'adresse https://console.aws.amazon.com/eks/ home#/clusters.
2. Sélectionnez le nom de votre cluster.
3. Choisissez l'onglet Fonctionnalités pour afficher l'état de toutes les fonctionnalités.
4. Pour obtenir des informations détaillées sur l'état de santé, cliquez sur l'onglet Observabilité, puis sur Surveiller le cluster, puis sur l'onglet Fonctionnalités.

AWS CLI :

```
aws eks describe-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name my-capability-name
```

Statuts des capacités

CRÉATION : La capacité est en cours de configuration. Vous pouvez quitter la console : la fonctionnalité continuera de se créer en arrière-plan.

ACTIF : La fonctionnalité fonctionne et est prête à être utilisée. Si les ressources ne fonctionnent pas comme prévu, vérifiez l'état des ressources et les autorisations IAM. Pour obtenir des conseils, veuillez consulter the section called “Fonctionnalités de résolution des problèmes”.

MISE À JOUR : Les modifications de configuration sont en cours d'application. Attendez que le statut revienne à`ACTIVE`.

SUPPRESSION : la fonctionnalité est supprimée du cluster.

CREATE_FAILED : le programme d'installation a rencontré une erreur. Les causes courantes incluent :

- Politique de confiance des rôles IAM incorrecte ou manquante
- Le rôle IAM n'existe pas ou n'est pas accessible
- Problèmes d'accès au cluster
- Paramètres de configuration non valides

Consultez la section sur l'état des fonctionnalités pour obtenir des informations spécifiques sur les erreurs.

UPDATE_FAILED : La mise à jour de configuration a échoué. Consultez la section sur l'état des fonctionnalités pour plus de détails et vérifiez les autorisations IAM.

> **Tip**
>
> Pour obtenir des conseils de dépannage détaillés, voir :
>
> - the section called "Fonctionnalités de résolution des problèmes"- Résolution des problèmes généraux liés aux fonctionnalités
> - the section called "Résolution des problèmes"- Problèmes spécifiques à ACK
> - the section called "Résolution des problèmes"- Problèmes spécifiques à Argo CD
> - the section called "Résolution des problèmes"- problèmes spécifiques à Kro

Créez des capacités

Pour créer une fonctionnalité sur votre cluster, consultez les rubriques suivantes :

- the section called "Créer une fonctionnalité ACK"— Créez une fonctionnalité ACK pour gérer les AWS ressources à l'aide de Kubernetes APIs
- the section called "Création d'une fonctionnalité Argo CD"— Création d'une fonctionnalité Argo CD pour GitOps une livraison continue
- the section called "Créez une capacité KRO"— Créez une capacité KRO pour la composition et l'orchestration des ressources

Fonctionnalités de la liste

Vous pouvez répertorier toutes les ressources de capacité d'un cluster.

Console

1. Ouvrez la console Amazon EKS à l'adresse https://console.aws.amazon.com/eks/ home#/clusters.
2. Sélectionnez le nom de votre cluster pour ouvrir la page détaillée du cluster.
3. Choisissez l'onglet Fonctionnalités.

4. Consultez les ressources relatives aux capacités sous Fonctionnalités gérées.

AWS CLI

Utilisez la `list-capabilities` commande pour afficher toutes les fonctionnalités de votre cluster. Remplacez *`region-code`* par la AWS région dans laquelle se trouve votre cluster et remplacez *`my-cluster`* par le nom de votre cluster.

```
aws eks list-capabilities \
  --region region-code \
  --cluster-name my-cluster
```


```
{
    "capabilities": [
        {
            "capabilityName": "my-ack",
            "arn": "arn:aws:eks:us-west-2:111122223333:capability/my-cluster/ack/my-
ack/abc123",
            "type": "ACK",
            "status": "ACTIVE",
            "createdAt": "2025-11-02T10:30:00.000000-07:00",
            "modifiedAt": "2025-11-02T10:32:15.000000-07:00",
        },
        {
            "capabilityName": "my-kro",
            "arn": "arn:aws:eks:us-west-2:111122223333:capability/my-cluster/kro/my-
kro/abc123",
            "type": "KRO",
            "status": "ACTIVE",
            "version": "v0.6.3",
            "createdAt": "2025-11-02T10:30:00.000000-07:00",
            "modifiedAt": "2025-11-02T10:32:15.000000-07:00",
        },
        {
            "capabilityName": "my-argocd",
            "arn": "arn:aws:eks:us-west-2:111122223333:capability/my-cluster/argocd/my-
argocd/abc123",
            "type": "ARGOCD",
            "status": "ACTIVE",
            "version": "3.1.8-eks-1",
            "createdAt": "2025-11-21T08:22:28.486000-05:00",
```

```
            "modifiedAt": "2025-11-21T08:22:28.486000-05:00"
        }
    ]
}
```

Décrire une fonctionnalité

Obtenez des informations détaillées sur une fonctionnalité spécifique, notamment sa configuration et son état.

Console

1. Ouvrez la console Amazon EKS à l'adresse https://console.aws.amazon.com/eks/ home#/clusters.
2. Sélectionnez le nom de votre cluster pour ouvrir la page détaillée du cluster.
3. Choisissez l'onglet Fonctionnalités.
4. Choisissez la fonctionnalité que vous souhaitez afficher dans la section Fonctionnalités gérées.
5. Consultez les détails des fonctionnalités, notamment l'état, la configuration et l'heure de création.

AWS CLI

Utilisez la `describe-capability` commande pour afficher des informations détaillées. Remplacez *`region-code`* par la AWS région dans laquelle se trouve votre cluster, remplacez *`my-cluster`* par le nom de votre cluster et remplacez par *`capability-name`* le nom de la fonctionnalité (ack, argocd ou kro).

```
aws eks describe-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name capability-name
```

Exemple de sortie :

```
{
  "capability": {
    "capabilityName": "my-ack",
    "capabilityArn": "arn:aws:eks:us-west-2:111122223333:capability/my-cluster/ack/my-
ack/abc123",
```

```
    "clusterName": "my-cluster",
    "type": "ACK",
    "roleArn": "arn:aws:iam::111122223333:role/AmazonEKSCapabilityACKRole",
    "status": "ACTIVE",
    "configuration": {},
    "tags": {},
    "health": {
      "issues": []
    },
    "createdAt": "2025-11-19T17:11:30.242000-05:00",
    "modifiedAt": "2025-11-19T17:11:30.242000-05:00",
    "deletePropagationPolicy": "RETAIN"
  }
}
```

Mettre à jour la configuration d'une fonctionnalité

Vous pouvez mettre à jour certains aspects de la configuration d'une fonctionnalité après sa création. Les options de configuration spécifiques varient en fonction du type de fonctionnalité.

> **Note**
>
> Les ressources des fonctionnalités d'EKS sont entièrement gérées, y compris les correctifs et les mises à jour de version. La mise à jour d'une fonctionnalité met à jour la configuration des ressources et n'entraîne pas de mise à jour de version des composants de la fonctionnalité gérée.

AWS CLI

Utilisez la `update-capability` commande pour modifier une fonctionnalité :

```
aws eks update-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name capability-name \
  --role-arn arn:aws:iam::[.replaceable]111122223333:role/NewCapabilityRole
```

> **Note**
>
> Les propriétés des fonctionnalités ne peuvent pas toutes être mises à jour après leur création. Reportez-vous à la documentation spécifique aux fonctionnalités pour plus de détails sur ce qui peut être modifié.

Supprimer une fonctionnalité

Lorsque vous n'avez plus besoin d'une fonctionnalité sur votre cluster, vous pouvez supprimer la ressource de capacité.

> **Important**
>
> Supprimez les ressources du cluster avant de supprimer la fonctionnalité.
> La suppression d'une ressource de capacité ne supprime pas automatiquement les ressources créées par le biais de cette fonctionnalité :
>
> - Toutes les définitions de ressources personnalisées Kubernetes (CRDs) restent installées dans votre cluster.
> - Les ressources ACK restent dans votre cluster et les AWS ressources correspondantes restent dans votre compte
> - Les applications Argo CD et leurs ressources Kubernetes restent dans votre cluster
> - kro ResourceGraphDefinitions et les instances restent dans votre cluster
>
> Vous devez supprimer ces ressources avant de supprimer la fonctionnalité afin d'éviter les ressources orphelines.
> Vous pouvez éventuellement choisir de conserver les AWS ressources associées aux ressources ACK Kubernetes. Voir les considérations relatives à l'ACK

Console

1. Ouvrez la console Amazon EKS à l'adresse https://console.aws.amazon.com/eks/ home#/clusters.
2. Sélectionnez le nom de votre cluster pour ouvrir la page détaillée du cluster.
3. Choisissez l'onglet Fonctionnalités.

4. Sélectionnez la fonctionnalité que vous souhaitez supprimer dans la liste des fonctionnalités gérées.
5. Choisissez la fonctionnalité Supprimer.
6. Dans la boîte de dialogue de confirmation, tapez le nom de la fonctionnalité pour confirmer la suppression.
7. Sélectionnez Delete (Supprimer).

AWS CLI

Utilisez la `delete-capability` commande pour supprimer une ressource de capacité :

Remplacez *`region-code`* par la AWS région dans laquelle se trouve votre cluster, remplacez *`my-cluster`* par le nom de votre cluster et remplacez *`capability-name`* par le nom de la fonctionnalité à supprimer.

```
aws eks delete-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name capability-name
```

Étapes suivantes

- the section called "ressources Kubernetes"— Découvrez les ressources Kubernetes fournies par chaque type de fonctionnalité
- the section called "Concepts d'ACK"— Comprendre les concepts ACK et le cycle de vie des ressources
- the section called "Travailler avec Argo CD"— Utilisation des fonctionnalités d'Argo CD pour GitOps les flux de travail
- the section called "concepts kro"— Comprendre les concepts Kro et la composition des ressources

Capacités : ressources Kubernetes

Une fois que vous avez activé une fonctionnalité sur votre cluster, vous interagissez le plus souvent avec elle en créant et en gérant des ressources personnalisées Kubernetes dans votre cluster.

Chaque fonctionnalité fournit son propre ensemble de définitions de ressources personnalisées (CRDs) qui étendent l'API Kubernetes avec des fonctionnalités spécifiques aux fonctionnalités.

Ressources sur Argo CD

Lorsque vous activez la fonctionnalité Argo CD, vous pouvez créer et gérer les ressources Kubernetes suivantes :

Application

Définit un déploiement depuis un dépôt Git vers un cluster cible. `Application`les ressources spécifient le référentiel source, l'espace de noms cible et la politique de synchronisation. Vous pouvez créer jusqu'à 1 000 `Application` ressources par instance de fonctionnalité Argo CD.

ApplicationSet

Génère plusieurs `Application` ressources à partir de modèles, permettant ainsi des déploiements multi-clusters et multi-environnements. `ApplicationSet`les ressources utilisent des générateurs pour créer `Application` des ressources dynamiquement en fonction de listes de clusters, de répertoires Git ou d'autres sources.

AppProject

Fournit un regroupement logique et un contrôle d'accès pour les `Application` ressources. `AppProject`les ressources définissent les référentiels, les clusters et les espaces de noms que les `Application` ressources peuvent utiliser, ce qui permet la mutualisation et les limites de sécurité.

Exemple `Application` de ressource :

```
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: my-app
  namespace: argocd
spec:
  project: default
  source:
    repoURL: https://github.com/org/repo
    targetRevision: main
    path: manifests
  destination:
```

```
    server: https://kubernetes.default.svc
    namespace: production
```

Pour plus d'informations sur les ressources et les concepts d'Argo CD, consultezthe section called "Concepts d'Argo CD".

ressources kro

Lorsque vous activez la fonctionnalité Kro, vous pouvez créer et gérer les ressources Kubernetes suivantes :

ResourceGraphDefinition (RGD)

Définit une API personnalisée qui compose plusieurs Kubernetes et AWS ressources dans une abstraction de niveau supérieur. Les équipes de la plateforme créent `ResourceGraphDefinition` des ressources pour fournir des modèles réutilisables avec des glissières de sécurité.

Instances de ressources personnalisées

Après avoir créé une `ResourceGraphDefinition` ressource, vous pouvez créer des instances de l'API personnalisée définie par le`ResourceGraphDefinition`. kro crée et gère automatiquement les ressources spécifiées dans le`ResourceGraphDefinition`.

Exemple `ResourceGraphDefinition` de ressource :

```
apiVersion: kro.run/v1alpha1
kind: ResourceGraphDefinition
metadata:
  name: web-application
spec:
  schema:
    apiVersion: v1alpha1
    kind: WebApplication
    spec:
      name: string
      replicas: integer
  resources:
    - id: deployment
      template:
        apiVersion: apps/v1
        kind: Deployment
```

```
        # ... deployment spec
    - id: service
      template:
        apiVersion: v1
        kind: Service
        # ... service spec
```

Exemple d'`WebApplication`instance :

```
apiVersion: v1alpha1
kind: WebApplication
metadata:
  name: my-web-app
  namespace: default
spec:
  name: my-web-app
  replicas: 3
```

Lorsque vous appliquez cette instance, kro crée automatiquement les `Service` ressources `Deployment` et définies dans le`ResourceGraphDefinition`.

Pour plus d'informations sur les ressources et les concepts de Kro, consultezthe section called "concepts kro".

Ressources de l'ACK

Lorsque vous activez la fonctionnalité ACK, vous pouvez créer et gérer des ressources à l'aide AWS des ressources personnalisées de Kubernetes. ACK en fournit plus de 200 CRDs pour plus de 50 AWS services, ce qui vous permet de définir AWS des ressources parallèlement à vos charges de travail Kubernetes et de gérer des ressources d' AWS infrastructure dédiées avec Kubernetes.

Exemples de ressources ACK :

S3 Bucket

`Bucket`les ressources créent et gèrent les compartiments Amazon S3 avec des politiques de versionnement, de chiffrement et de cycle de vie.

RDS DBInstance

`DBInstance`provisionner les ressources et gérer les instances de base de données Amazon RDS avec des sauvegardes et des fenêtres de maintenance automatisées.

Tableau DynamoDB

`Table`les ressources créent et gèrent des tables DynamoDB avec une capacité provisionnée ou à la demande.

IAM Role

`Role`les ressources définissent les rôles IAM avec des politiques de confiance et des politiques d'autorisation pour l'accès aux AWS services.

Fonction Lambda

`Function`les ressources créent et gèrent les fonctions Lambda avec la configuration du code, de l'exécution et des rôles d'exécution.

Exemple de spécification d'une `Bucket` ressource :

```
apiVersion: s3.services.k8s.aws/v1alpha1
kind: Bucket
metadata:
  name: my-app-bucket
spec:
  name: my-unique-bucket-name-12345
  versioning:
    status: Enabled
  encryption:
    rules:
      - applyServerSideEncryptionByDefault:
          sseAlgorithm: AES256
```

Pour plus d'informations sur les ressources et les concepts d'ACK, consultezthe section called "Concepts d'ACK".

Limites des ressources

Les capacités EKS sont soumises aux limites de ressources suivantes :

Limites d'utilisation du CD Argo :

- 1 000 `Application` ressources maximum par instance de capacité Argo CD
- Maximum de 100 clusters distants configurés par instance de capacité Argo CD

Limites de configuration des ressources :

- 150 ressources Kubernetes maximum par `Application` ressource dans Argo CD
- 64 ressources Kubernetes maximum par in kro `ResourceGraphDefinition`

> **Note**
>
> Ces limites s'appliquent au nombre de ressources gérées par chaque instance de fonctionnalité. Si vous avez besoin de limites plus élevées, vous pouvez déployer des fonctionnalités sur plusieurs clusters.

Étapes suivantes

Pour les tâches spécifiques aux fonctionnalités et la configuration avancée, consultez les rubriques suivantes :

- the section called “Concepts d'ACK”— Comprendre les concepts ACK et le cycle de vie des ressources
- the section called “Travailler avec Argo CD”— Utilisation des fonctionnalités d'Argo CD pour GitOps les flux de travail
- the section called “concepts kro”— Comprendre les concepts Kro et la composition des ressources

Considérations relatives aux fonctionnalités EKS

Cette rubrique aborde les considérations importantes relatives à l'utilisation des fonctionnalités EKS, notamment la conception du contrôle d'accès, le choix entre les fonctionnalités EKS et les solutions autogérées, les modèles architecturaux pour les déploiements multiclusters et les meilleures pratiques opérationnelles.

Fonctionnalité : rôles IAM et Kubernetes RBAC

Chaque ressource de capacité EKS possède un rôle IAM de fonctionnalité configuré. Le rôle de capacité est utilisé pour accorder des autorisations de AWS service permettant aux fonctionnalités EKS d'agir en votre nom. Par exemple, pour utiliser la fonctionnalité EKS pour ACK afin de gérer les compartiments Amazon S3, vous devez accorder à cette fonctionnalité des autorisations administratives lui permettant de créer et de gérer des compartiments.

Une fois la fonctionnalité configurée, les ressources S3 dans AWS peuvent être créées et gérées avec les ressources personnalisées Kubernetes de votre cluster. Kubernetes RBAC est le mécanisme de contrôle d'accès intégré au cluster qui permet de déterminer quels utilisateurs et groupes peuvent créer et gérer ces ressources personnalisées. Par exemple, accordez à des utilisateurs et à des groupes Kubernetes RBAC spécifiques des autorisations pour créer et gérer des `Bucket` ressources dans les espaces de noms de votre choix.

Ainsi, IAM et Kubernetes RBAC sont les deux moitiés du système de contrôle d' end-to-endaccès qui régit les autorisations liées aux capacités et aux ressources d'EKS. Il est important de concevoir la bonne combinaison d'autorisations IAM et de politiques d'accès RBAC pour votre cas d'utilisation.

Pour plus d'informations sur les fonctionnalités, les rôles IAM et les autorisations Kubernetes, consultez. the section called “Considérations relatives aux fonctionnalités d'EKS”

Modèles d'architecture multi-clusters

Lorsque vous déployez des fonctionnalités sur plusieurs clusters, tenez compte des modèles architecturaux courants suivants :

Hub and Spoke avec gestion centralisée

Exécutez les trois fonctionnalités dans un cluster géré de manière centralisée pour orchestrer les charges de travail et gérer l'infrastructure cloud sur plusieurs clusters de charge de travail.

- Argo CD sur le cluster de gestion déploie des applications sur des clusters de charge de travail situés dans différentes régions ou comptes
- ACK sur le cluster de gestion fournit des AWS ressources (RDS, S3, IAM) pour tous les clusters
- kro sur le cluster de gestion crée des abstractions de plate-forme portables qui fonctionnent sur tous les clusters

Ce modèle centralise la gestion de la charge de travail et de l'infrastructure cloud, et peut simplifier les opérations pour les entreprises qui gèrent de nombreux clusters.

Décentralisé GitOps

Les charges de travail et l'infrastructure cloud sont gérées par des fonctionnalités du même cluster sur lequel les charges de travail sont exécutées.

- Argo CD gère les ressources de l'application sur le cluster local.
- Les ressources ACK sont utilisées pour les besoins des clusters et des charges de travail.

- les abstractions de la plateforme kro sont installées et orchestrent les ressources locales.

Ce modèle décentralise les opérations, les équipes gérant leurs propres services de plateforme dédiés dans un ou plusieurs clusters.

Déploiement de Hub and Spoke avec Hybrid ACK

Combinez des modèles centralisés et décentralisés, avec des déploiements d'applications centralisés et une gestion des ressources en fonction du périmètre et de la propriété.

- Cluster central :
  - Argo CD gère les GitOps déploiements vers le cluster local et tous les clusters de charge de travail distants
  - ACK est utilisé sur le cluster de gestion pour les ressources administrées (bases de données de production, rôles IAM,) VPCs
  - kro est utilisé sur le cluster de gestion pour les abstractions de plateforme réutilisables
- Clusters de rayons :
  - Les charges de travail sont gérées via Argo CD sur le cluster centralisé
  - ACK est utilisé localement pour les ressources limitées à la charge de travail (compartiments S3, instances, ElastiCache files d'attente SQS)
  - kro est utilisé localement pour les compositions de ressources et les modèles de blocs de construction

Ce schéma distingue les préoccupations : les équipes chargées des plateformes gèrent les infrastructures critiques de manière centralisée sur des clusters de gestion, y compris éventuellement des clusters de charge de travail, tandis que les équipes chargées des applications spécifient et gèrent les ressources cloud parallèlement aux charges de travail.

Choisir un motif

Tenez compte des facteurs suivants lors du choix d'une architecture :

- Structure organisationnelle : les équipes centralisées privilégient les modèles de hub ; les équipes décentralisées peuvent préférer les capacités par cluster
- Étendue des ressources : les ressources réservées aux administrateurs (bases de données, IAM) bénéficient souvent d'une gestion centralisée ; les ressources de charge de travail (compartiments, files d'attente) peuvent être gérées localement

- Libre-service : les équipes centralisées de la plateforme peuvent créer et distribuer des ressources personnalisées prescriptives afin de permettre un libre-service sécurisé des ressources cloud pour les besoins de charge de travail courants
- Gestion du parc de clusters : les clusters de gestion centralisée fournissent un plan de contrôle appartenant au client pour la gestion du parc de clusters EKS, ainsi que d'autres ressources destinées aux administrateurs
- Exigences de conformité : certaines organisations ont besoin d'un contrôle centralisé pour l'audit et la gouvernance
- Complexité opérationnelle : la diminution du nombre d'instances de capacité simplifie les opérations mais peut créer des goulots d'étranglement

> **Note**
>
> Vous pouvez commencer avec un modèle et évoluer vers un autre au fur et à mesure que votre plateforme mûrit. Les fonctionnalités sont indépendantes : vous pouvez les déployer différemment entre les clusters en fonction de vos besoins.

Comparaison des fonctionnalités d'EKS avec les solutions autogérées

Les fonctionnalités EKS fournissent des expériences entièrement gérées pour les outils et contrôleurs Kubernetes les plus courants qui s'exécutent dans EKS. Cela diffère des solutions autogérées, que vous installez et exploitez dans votre cluster.

Principales différences

Déploiement et gestion

AWS gère entièrement les fonctionnalités EKS sans qu'aucune installation, configuration ou maintenance des composants logiciels ne soit requise. AWS installe et gère automatiquement toutes les définitions de ressources personnalisées Kubernetes (CRDs) requises dans le cluster.

Avec les solutions autogérées, vous installez et configurez un logiciel de cluster à l'aide de graphiques Helm, de kubectl ou d'autres opérateurs. Vous avez le contrôle total du cycle de vie des logiciels et de la configuration d'exécution de vos solutions autogérées, en fournissant des personnalisations à tous les niveaux de la solution.

Exploitation et maintenance

AWS gère les correctifs et autres opérations du cycle de vie des logiciels pour EKS Capabilities, avec des mises à jour automatiques et des correctifs de sécurité. Les fonctionnalités EKS sont intégrées à des AWS fonctionnalités pour des configurations rationalisées, offrent une haute disponibilité et une tolérance aux pannes intégrées, et éliminent le dépannage des charges de travail des contrôleurs au sein du cluster.

Les solutions autogérées vous obligent à surveiller l'état de santé et les journaux des composants, à appliquer des correctifs de sécurité et des mises à jour de version, à configurer la haute disponibilité avec plusieurs répliques et des budgets d'interruption des pods, à résoudre les problèmes de charge de travail des contrôleurs et à gérer les versions et les versions. Vous avez le contrôle total de vos déploiements, mais cela nécessite souvent des solutions sur mesure pour l'accès aux clusters privés et d'autres intégrations qui doivent être conformes aux normes organisationnelles et aux exigences de conformité en matière de sécurité.

Consommation de ressources

Les fonctionnalités EKS s'exécutent dans EKS et hors de vos clusters, libérant ainsi des ressources de nœud et de cluster. Les fonctionnalités n'utilisent pas les ressources de charge de travail du cluster, ne consomment ni le processeur ni la mémoire de vos nœuds de travail, évoluent automatiquement et ont un impact minimal sur la planification de la capacité du cluster.

Les solutions autogérées exécutent des contrôleurs et d'autres composants sur vos nœuds de travail, consommant ainsi directement les ressources des nœuds de travail IPs, du cluster et d'autres ressources du cluster. La gestion des services de cluster nécessite une planification des capacités pour leurs charges de travail, ainsi que la planification et la configuration des demandes de ressources et des limites afin de gérer les exigences de scalabilité et de haute disponibilité.

Support des fonctionnalités

En tant que fonctionnalités de service entièrement gérées, les fonctionnalités EKS Capabilities sont, par nature, bien ancrées par rapport aux solutions autogérées. Bien que les fonctionnalités soient compatibles avec la plupart des fonctionnalités et des cas d'utilisation, il y aura une différence de couverture par rapport aux solutions autogérées.

Avec les solutions autogérées, vous contrôlez entièrement la configuration, les fonctionnalités optionnelles et les autres aspects des fonctionnalités de votre logiciel. Vous pouvez choisir d'exécuter vos propres images personnalisées, de personnaliser tous les aspects de la configuration et de contrôler entièrement les fonctionnalités de votre solution autogérée.

Considérations relatives aux coûts

Chaque ressource de capacité EKS a un coût horaire associé, qui varie en fonction du type de capacité. Les ressources du cluster gérées par cette fonctionnalité ont également un coût horaire associé à leur propre tarification. Pour plus d'informations, consultez les tarifs Amazon EKS.

Les solutions autogérées n'ont aucun coût direct lié aux AWS frais, mais vous payez pour les ressources de calcul du cluster utilisées par les contrôleurs et les charges de travail associées. Au-delà de la consommation des ressources des nœuds et des clusters, le coût total de possession des solutions autogérées inclut les frais d'exploitation et les dépenses de maintenance, de dépannage et de support.

Choisir entre les fonctionnalités EKS et les solutions autogérées

Capacités d'EKS Envisagez ce choix lorsque vous souhaitez réduire les frais opérationnels et vous concentrer sur la valeur différenciée de vos logiciels et systèmes, plutôt que sur les opérations de plate-forme de cluster pour répondre à des exigences de base. Utilisez les fonctionnalités EKS lorsque vous souhaitez minimiser la charge opérationnelle liée aux correctifs de sécurité et à la gestion du cycle de vie des logiciels, libérer des ressources de nœuds et de clusters pour les charges de travail des applications, simplifier la configuration et la gestion de la sécurité, et bénéficier d'une couverture de AWS support. Les fonctionnalités EKS sont idéales pour la plupart des cas d'utilisation en production et constituent l'approche recommandée pour les nouveaux déploiements.

Solutions autogérées Envisagez ce choix lorsque vous avez besoin de versions spécifiques de l'API de ressources Kubernetes, de versions de contrôleurs personnalisées, de systèmes d'automatisation et d'outils existants basés sur des déploiements autogérés ou lorsque vous avez besoin d'une personnalisation approfondie des configurations d'exécution des contrôleurs. Les solutions autogérées offrent de la flexibilité pour les cas d'utilisation spécialisés et vous permettent de contrôler totalement votre déploiement et votre configuration d'exécution.

> **Note**
>
> Les fonctionnalités EKS peuvent coexister dans votre cluster avec des solutions autogérées, et des migrations par étapes sont possibles.

Comparaisons spécifiques aux capacités

Pour des comparaisons détaillées, notamment les fonctionnalités spécifiques aux capacités, les différences en amont et les chemins de migration, voir :

- the section called “Comparaison avec l'autogestion”
- the section called “Comparaison avec l'autogestion”
- the section called “Comparaison avec l'autogestion”

Déployez AWS des ressources depuis Kubernetes avec AWS Controllers for Kubernetes (ACK)

AWS Controllers for Kubernetes (ACK) vous permet de définir et de gérer les ressources de AWS service directement depuis Kubernetes. Avec AWS Controllers for Kubernetes (ACK), vous pouvez gérer les ressources de charge de travail et l'infrastructure cloud à l'aide de ressources personnalisées Kubernetes, parallèlement aux charges de travail de vos applications à l'aide de Kubernetes et d'outils Kubernetes familiers. APIs

Grâce aux fonctionnalités d'EKS, ACK est entièrement géré par AWS, ce qui élimine le besoin d'installer, de maintenir et de dimensionner les contrôleurs ACK sur vos clusters.

Comment fonctionne ACK

ACK traduit les spécifications de ressources personnalisées de Kubernetes en AWS appels d'API. Lorsque vous créez, mettez à jour ou supprimez une ressource personnalisée Kubernetes représentant une ressource de AWS service, ACK effectue les appels d' AWS API requis pour créer, mettre à jour ou supprimer la ressource. AWS

Chaque AWS ressource prise en charge par ACK possède sa propre définition de ressource personnalisée (CRD) qui définit le schéma de l'API Kubernetes pour spécifier sa configuration. Par exemple, ACK fournit CRDs S3, notamment des compartiments, des politiques de compartiment et d'autres ressources S3.

ACK concilie en permanence l'état de vos AWS ressources avec l'état souhaité défini dans vos ressources personnalisées Kubernetes. Si une ressource s'écarte de son état souhaité, ACK le détecte et prend des mesures correctives pour la remettre dans son alignement. Les modifications apportées aux ressources Kubernetes se répercutent immédiatement sur l'état AWS des ressources, tandis que la détection passive des dérives et la correction des modifications des AWS ressources en amont peuvent prendre jusqu'à 10 heures (période de resynchronisation), mais se produisent généralement bien plus tôt.

Exemple de manifeste de ressources du compartiment S3

```
apiVersion: s3.services.k8s.aws/v1alpha1
kind: Bucket
metadata:
  name: my-ack-bucket
spec:
  name: my-unique-bucket-name
```

Lorsque vous appliquez cette ressource personnalisée à votre cluster, ACK crée un compartiment Amazon S3 dans votre compte s'il n'existe pas encore. Les modifications ultérieures apportées à cette ressource, par exemple en spécifiant un niveau de stockage autre que celui par défaut ou en ajoutant une politique, seront appliquées à la ressource S3 dans AWS. Lorsque cette ressource est supprimée du cluster, le compartiment S3 AWS est supprimé par défaut.

Avantages de l'ACK

ACK fournit une gestion des AWS ressources native de Kubernetes, vous permettant de gérer les AWS ressources en utilisant les mêmes Kubernetes APIs et les mêmes outils que ceux que vous utilisez pour vos applications. Cette approche unifiée simplifie le flux de travail de gestion de votre infrastructure en éliminant le besoin de passer d'un outil à l'autre ou d'apprendre à utiliser infrastructure-as-code des systèmes distincts. Vous définissez vos AWS ressources de manière déclarative dans les manifestes Kubernetes, en activant les GitOps flux de travail et l'infrastructure en tant que pratiques de code qui s'intègrent parfaitement à vos processus de développement existants.

ACK concilie en permanence l'état souhaité de vos AWS ressources avec leur état réel, en corrigeant les dérives et en garantissant la cohérence au sein de votre infrastructure. Ce rapprochement continu signifie que les out-of-band modifications impératives apportées aux AWS ressources sont automatiquement annulées pour correspondre à votre configuration déclarée, préservant ainsi l'intégrité de votre infrastructure sous forme de code. Vous pouvez configurer ACK pour gérer les ressources sur plusieurs AWS comptes et régions, en activant des architectures multicomptes complexes sans outils supplémentaires.

Pour les entreprises qui migrent depuis d'autres outils de gestion d'infrastructure, ACK prend en charge l'adoption des ressources, ce qui vous permet de placer les AWS ressources existantes sous la gestion d'ACK sans les recréer. ACK fournit également des ressources en lecture seule pour l'observation AWS des ressources sans accès aux modifications, ainsi que des annotations pour éventuellement conserver les AWS ressources même lorsque la ressource Kubernetes est supprimée du cluster.

Pour en savoir plus et commencer à utiliser la fonctionnalité EKS pour ACK, consultez the section called “Concepts d'ACK” etthe section called “Considérations”.

AWS Services pris en charge

ACK prend en charge un large éventail de AWS services, y compris, mais sans s'y limiter :

- Amazon EC2
- Amazon S3
- Amazon RDS
- Amazon DynamoDB
- Amazon ElastiCache
- Amazon EKS
- Amazon SQS
- Amazon SNS
- AWS Lambda
- AWS IAM

Tous les AWS services répertoriés comme généralement disponibles en amont sont pris en charge par la fonctionnalité EKS pour ACK. Consultez la liste complète des AWS services pris en charge pour plus de détails.

Intégration avec d'autres fonctionnalités gérées par EKS

ACK s'intègre aux autres fonctionnalités gérées par EKS.

- Argo CD : utilisez Argo CD pour gérer le déploiement des ressources ACK sur plusieurs clusters, en activant les GitOps flux de travail pour votre AWS infrastructure.
  - ACK étend les avantages de l' GitOps association avec ArgoCD, mais ACK ne nécessite pas d'intégration avec git.
- kro (Kube Resource Orchestrator) : utilisez kro pour composer des ressources complexes à partir de ressources ACK, en créant des abstractions de haut niveau qui simplifient la gestion des ressources.
  - Vous pouvez créer des ressources personnalisées composites avec kro qui définissent à la fois les ressources Kubernetes et les ressources. AWS Les membres de l'équipe peuvent utiliser ces ressources personnalisées pour déployer rapidement des applications complexes.

Débuter avec ACK

Pour commencer à utiliser la fonctionnalité EKS pour ACK :

1. Créez et configurez un rôle de capacité IAM avec les autorisations nécessaires pour qu'ACK gère les AWS ressources en votre nom.
2. Créez une ressource de capacité ACK sur votre cluster EKS via la AWS console, la AWS CLI ou votre infrastructure préférée en tant qu'outil de code.
3. Appliquez des ressources personnalisées Kubernetes à votre cluster pour commencer à gérer vos AWS ressources dans Kubernetes.

Création d'une fonctionnalité ACK

Ce chapitre explique comment créer une fonctionnalité ACK sur votre cluster Amazon EKS.

Conditions préalables

Avant de créer une fonctionnalité ACK, assurez-vous d'avoir :

- Un cluster Amazon EKS
- Un rôle de capacité IAM avec des autorisations permettant à ACK de gérer les ressources AWS
- Autorisations IAM suffisantes pour créer des ressources de capacités sur les clusters EKS
- L'outil CLI approprié installé et configuré, ou l'accès à la console EKS

Pour obtenir des instructions sur la création du rôle de capacité IAM, consultezthe section called "Capacité : rôle IAM".

> **Important**
>
> ACK est une fonctionnalité de gestion d'infrastructure qui permet de créer, de modifier et de supprimer AWS des ressources. Il s'agit d'une fonctionnalité réservée aux administrateurs qui doit être soigneusement contrôlée. Toute personne autorisée à créer des ressources Kubernetes dans votre cluster peut créer efficacement des AWS ressources via ACK, sous réserve des autorisations du rôle de capacité IAM. Le rôle de capacité IAM que vous fournissez détermine les AWS ressources qu'ACK peut créer et gérer. Pour obtenir des conseils sur la création d'un rôle approprié avec des autorisations de moindre privilège,

consultez the section called “Capacité : rôle IAM” et. the section called “Considérations relatives aux fonctionnalités d'EKS”

Choisissez votre outil

Vous pouvez créer une fonctionnalité ACK à l'aide de la AWS Management Console AWS CLI ou de eksctl :

- the section called “Console”- Utilisez la console pour une expérience guidée
- the section called “ AWS CLI”- Utilisez la AWS CLI pour la création de scripts et l'automatisation
- the section called “eksctl”- Utilisez eksctl pour une expérience native de Kubernetes

Que se passe-t-il lorsque vous créez une fonctionnalité ACK ?

Lorsque vous créez une fonctionnalité ACK :

1. EKS crée le service de capacité ACK et le configure pour surveiller et gérer les ressources de votre cluster
2. Des définitions de ressources personnalisées (CRDs) sont installées dans votre cluster
3. Une entrée d'accès est automatiquement créée pour votre rôle de capacité IAM avec des politiques d'entrée d'accès spécifiques aux fonctionnalités qui accordent des autorisations Kubernetes de base (voir) the section called “Considérations relatives aux fonctionnalités d'EKS”
4. La fonctionnalité assume le rôle de capacité IAM que vous fournissez.
5. ACK commence à surveiller ses ressources personnalisées dans votre cluster
6. L'état de capacité passe de `CREATING` à `ACTIVE`

Une fois active, vous pouvez créer des ressources personnalisées ACK dans votre cluster pour gérer les AWS ressources.

> **Note**
>
> L'entrée d'accès créée automatiquement inclut celle `AmazonEKSACKPolicy` qui accorde à ACK les autorisations nécessaires pour gérer les AWS ressources. Certaines ressources ACK qui font référence à des secrets Kubernetes (telles que les bases de données RDS avec mots de passe) nécessitent des politiques d'accès supplémentaires. Pour en savoir plus sur

les entrées d'accès et comment configurer des autorisations supplémentaires, consultezthe section called “Considérations relatives aux fonctionnalités d'EKS”.

Étapes suivantes

Après avoir créé la fonctionnalité ACK :

- the section called “Concepts d'ACK”- Comprenez les concepts ACK et commencez à utiliser les AWS ressources
- the section called “Concepts d'ACK”- En savoir plus sur le rapprochement, les exportations sur le terrain et les modèles d'adoption des ressources
- the section called “Configurer les autorisations ”- Configurer les autorisations IAM et les modèles multi-comptes

Création d'une fonctionnalité ACK à l'aide de la console

Cette rubrique décrit comment créer une fonctionnalité AWS Controllers for Kubernetes (ACK) à l'aide du. AWS Management Console

Création de la fonctionnalité ACK

1. Ouvrez la console Amazon EKS à l'adresse https://console.aws.amazon.com/eks/ home#/clusters.
2. Sélectionnez le nom de votre cluster pour ouvrir la page détaillée du cluster.
3. Choisissez l'onglet Fonctionnalités.
4. Dans le menu de navigation de gauche, choisissez AWS Controllers for Kubernetes (ACK).
5. Choisissez Create AWS Controllers pour la fonctionnalité Kubernetes.
6. Pour le rôle de capacité IAM :
   - Si vous possédez déjà un rôle de capacité IAM, sélectionnez-le dans le menu déroulant
   - Si vous devez créer un rôle, choisissez Créer un rôle d'administrateur

     Cela ouvre la console IAM dans un nouvel onglet avec la politique de confiance préremplie et la politique `AdministratorAccess` gérée. Vous pouvez désélectionner cette politique et ajouter d'autres autorisations si vous le souhaitez.

     Après avoir créé le rôle, retournez à la console EKS et le rôle sera automatiquement sélectionné.

> **Important**
>
> La `AdministratorAccess` politique suggérée accorde des autorisations étendues et vise à rationaliser le démarrage. Pour une utilisation en production, remplacez-la par une politique personnalisée qui n'accorde que les autorisations nécessaires pour les AWS services spécifiques que vous prévoyez de gérer avec ACK. Pour obtenir des conseils sur la création de politiques de moindre privilège, consultez the section called “Configurer les autorisations ” et. the section called “Considérations relatives aux fonctionnalités d'EKS”

7. Choisissez Créer.

Le processus de création de capacités commence.

Vérifiez que la fonctionnalité est active

1. Dans l'onglet Fonctionnalités, consultez l'état de la fonctionnalité ACK.
2. Attendez que le statut passe de `CREATING` à`ACTIVE`.
3. Une fois active, la fonctionnalité est prête à être utilisée.

Pour plus d'informations sur l'état des fonctionnalités et la résolution des problèmes, consultezthe section called “Travailler avec les capacités”.

Vérifiez que les ressources personnalisées sont disponibles

Une fois la fonctionnalité activée, vérifiez que les ressources personnalisées ACK sont disponibles dans votre cluster.

Utilisation de la console

1. Accédez à votre cluster dans la console Amazon EKS
2. Choisissez l'onglet Ressources
3. Choisissez les extensions
4. Choisissez CustomResourceDefinitions

Vous devriez voir un certain nombre de AWS ressources CRDs répertoriées.

Utilisation de kubectl

```
kubectl api-resources | grep services.k8s.aws
```

Vous devriez voir un certain nombre de AWS ressources APIs répertoriées.

> **Note**
>
> La fonctionnalité des AWS Controllers for Kubernetes installera un certain nombre de ressources CRDs pour diverses ressources. AWS

Étapes suivantes

- the section called "Concepts d'ACK"- Comprenez les concepts ACK et lancez-vous
- the section called "Configurer les autorisations "- Configurer les autorisations IAM pour les autres services AWS
- the section called "Travailler avec les capacités"- Gérez votre ressource de capacité ACK

Création d'une fonctionnalité ACK à l'aide de la AWS CLI

Cette rubrique décrit comment créer une fonctionnalité AWS Controllers for Kubernetes (ACK) à l'aide de la CLI. AWS

Conditions préalables

- AWS CLI : version `2.12.3` ou ultérieure. Pour vérifier votre version, lancez`aws --version`. Pour plus d'informations, consultez la section Installation dans le guide de l'utilisateur de l'interface de ligne de AWS commande.
- **`kubectl`** : outil de ligne de commande pour travailler avec des clusters Kubernetes. Pour de plus amples informations, veuillez consulter the section called "Configurer kubectl et eksctl".

Étape 1 : Création d'un rôle de capacité IAM

Créez un fichier de politique de confiance :

```
cat > ack-trust-policy.json << 'EOF'
{
  "Version": "2012-10-17",
```

```
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "capabilities.eks.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ]
    }
  ]
}
EOF
```

Créez le rôle IAM :

```
aws iam create-role \
  --role-name ACKCapabilityRole \
  --assume-role-policy-document file://ack-trust-policy.json
```

Associez la politique `AdministratorAccess` gérée au rôle :

```
aws iam attach-role-policy \
  --role-name ACKCapabilityRole \
  --policy-arn arn:aws:iam::aws:policy/AdministratorAccess
```

> **Important**
>
> La `AdministratorAccess` politique suggérée accorde des autorisations étendues et vise à rationaliser le démarrage. Pour une utilisation en production, remplacez-la par une politique personnalisée qui n'accorde que les autorisations nécessaires pour les AWS services spécifiques que vous prévoyez de gérer avec ACK. Pour obtenir des conseils sur la création de politiques de moindre privilège, consultez the section called “Configurer les autorisations ” et. the section called “Considérations relatives aux fonctionnalités d'EKS”

Étape 2 : Création de la fonctionnalité ACK

Créez la ressource de capacité ACK sur votre cluster. Remplacez *`region-code`* par la AWS région dans laquelle se trouve votre cluster et remplacez *`my-cluster`* par le nom de votre cluster.

```
aws eks create-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name my-ack \
  --type ACK \
  --role-arn arn:aws:iam::$(aws sts get-caller-identity --query Account --output
 text):role/ACKCapabilityRole \
  --delete-propagation-policy RETAIN
```

La commande revient immédiatement, mais la fonctionnalité met un certain temps à devenir active car EKS crée l'infrastructure et les composants de capacité requis. EKS installera les définitions de ressources personnalisées Kubernetes associées à cette fonctionnalité dans votre cluster lors de sa création.

> **Note**
>
> Si vous recevez un message d'erreur indiquant que le cluster n'existe pas ou que vous n'êtes pas autorisé, vérifiez :
>
> - Le nom du cluster est correct
> - Votre AWS CLI est configurée pour la bonne région
> - Vous disposez des autorisations IAM requises

Étape 3 : vérifier que la fonctionnalité est active

Attendez que la fonctionnalité soit activée. Remplacez *region-code* par la AWS région dans laquelle se trouve votre cluster et remplacez *my-cluster* par le nom de votre cluster.

```
aws eks describe-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name my-ack \
  --query 'capability.status' \
  --output text
```

La fonctionnalité est prête lorsque l'état s'afficheACTIVE. Ne passez pas à l'étape suivante tant que le statut n'est pas atteintACTIVE.

Vous pouvez également consulter les détails complets des fonctionnalités :

```
aws eks describe-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name my-ack
```

Étape 4 : vérifier que les ressources personnalisées sont disponibles

Une fois la fonctionnalité activée, vérifiez que les ressources personnalisées ACK sont disponibles dans votre cluster :

```
kubectl api-resources | grep services.k8s.aws
```

Vous devriez voir un certain nombre de AWS ressources APIs répertoriées.

> **Note**
>
> La fonctionnalité des AWS Controllers for Kubernetes installera un certain nombre de ressources CRDs pour diverses ressources. AWS

Étapes suivantes

- the section called "Concepts d'ACK"- Comprenez les concepts ACK et lancez-vous
- the section called "Configurer les autorisations "- Configurer les autorisations IAM pour les autres services AWS
- the section called "Travailler avec les capacités"- Gérez votre ressource de capacité ACK

Création d'une capacité ACK à l'aide de eksctl

Cette rubrique décrit comment créer une fonctionnalité AWS Controllers for Kubernetes (ACK) à l'aide de eksctl.

> **Note**
>
> Les étapes suivantes nécessitent la version eksctl `0.220.0` ou une version ultérieure. Pour vérifier votre version, lancez`eksctl version`.

Étape 1 : Création d'un rôle de capacité IAM

Créez un fichier de politique de confiance :

```
cat > ack-trust-policy.json << 'EOF'
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "capabilities.eks.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ]
    }
  ]
}
EOF
```

Créez le rôle IAM :

```
aws iam create-role \
  --role-name ACKCapabilityRole \
  --assume-role-policy-document file://ack-trust-policy.json
```

Associez la politique `AdministratorAccess` gérée au rôle :

```
aws iam attach-role-policy \
  --role-name ACKCapabilityRole \
  --policy-arn arn:aws:iam::aws:policy/AdministratorAccess
```

> **Important**
>
> La `AdministratorAccess` politique suggérée accorde des autorisations étendues et vise à rationaliser le démarrage. Pour une utilisation en production, remplacez-la par une politique personnalisée qui n'accorde que les autorisations nécessaires pour les AWS services spécifiques que vous prévoyez de gérer avec ACK. Pour obtenir des conseils sur la création

de politiques de moindre privilège, consultez the section called “Configurer les autorisations ” et. the section called “Considérations relatives aux fonctionnalités d'EKS”

> **Important**
>
> Cette politique accorde des autorisations pour la gestion des compartiments S3 avec`"Resource": "*"`, ce qui permet d'effectuer des opérations sur tous les compartiments S3.
> Pour une utilisation en production :* Limitez le `Resource` champ à des modèles de bucket ARNs ou de nom spécifiques * Utilisez les clés de condition IAM pour limiter l'accès par balises de ressource * N'accordez que les autorisations minimales nécessaires à votre cas d'utilisation
> Pour les autres AWS services, voirthe section called “Configurer les autorisations ”.

Attachez la stratégie au rôle :

```
aws iam attach-role-policy \
  --role-name ACKCapabilityRole \
  --policy-arn arn:aws:iam::$(aws sts get-caller-identity --query Account --output
 text):policy/ACKS3Policy
```

Étape 2 : Création de la fonctionnalité ACK

Créez la capacité ACK à l'aide de eksctl. Remplacez *`region-code`* par la AWS région dans laquelle se trouve votre cluster et remplacez *`my-cluster`* par le nom de votre cluster.

```
eksctl create capability \
  --cluster [.replaceable]`my-cluster` \
  --region [.replaceable]`region-code` \
  --name ack \
  --type ACK \
  --role-arn arn:aws:iam::$(aws sts get-caller-identity --query Account --output
 text):role/ACKCapabilityRole \
  --ack-service-controllers s3
```

> **Note**
>
> Le `--ack-service-controllers` drapeau est facultatif. En cas d'omission, ACK active tous les contrôleurs disponibles. Pour améliorer les performances et la sécurité, pensez à activer uniquement les contrôleurs dont vous avez besoin. Vous pouvez spécifier plusieurs contrôleurs : `--ack-service-controllers s3,rds,dynamodb`

La commande revient immédiatement, mais la fonctionnalité met un certain temps à devenir active.

Étape 3 : vérifier que la fonctionnalité est active

Vérifiez l'état des capacités :

```
eksctl get capability \
  --cluster [.replaceable]`my-cluster` \
  --region [.replaceable]`region-code` \
  --name ack
```

La fonctionnalité est prête lorsque l'état s'afficheACTIVE.

Étape 4 : vérifier que les ressources personnalisées sont disponibles

Une fois la fonctionnalité activée, vérifiez que les ressources personnalisées ACK sont disponibles dans votre cluster :

```
kubectl api-resources | grep services.k8s.aws
```

Vous devriez voir un certain nombre de AWS ressources APIs répertoriées.

> **Note**
>
> La fonctionnalité des AWS Controllers for Kubernetes installera un certain nombre de ressources CRDs pour diverses ressources. AWS

Étapes suivantes

- the section called “Concepts d'ACK”- Comprenez les concepts ACK et lancez-vous

- the section called “Configurer les autorisations ”- Configurer les autorisations IAM pour les autres services AWS
- the section called “Travailler avec les capacités”- Gérez votre ressource de capacité ACK

Concepts d'ACK

ACK gère les AWS ressources via Kubernetes en APIs conciliant en permanence l'état souhaité dans vos manifestes avec l'état réel dans. AWS Lorsque vous créez ou mettez à jour une ressource personnalisée Kubernetes, ACK effectue les appels d' AWS API nécessaires pour créer ou modifier la AWS ressource correspondante, puis surveille sa dérive et met à jour le statut de Kubernetes pour refléter l'état actuel. Cette approche vous permet de gérer l'infrastructure à l'aide d'outils et de flux de travail Kubernetes familiers tout en maintenant la cohérence entre votre cluster et. AWS

Cette rubrique explique les concepts fondamentaux qui sous-tendent la façon dont ACK gère les AWS ressources via Kubernetes APIs.

Commencer à utiliser ACK

Après avoir créé la fonctionnalité ACK (voirthe section called “Créer une fonctionnalité ACK”), vous pouvez commencer à gérer les AWS ressources à l'aide de manifestes Kubernetes dans votre cluster.

Par exemple, créez ce manifeste de compartiment S3 dans`bucket.yaml`, en choisissant votre propre nom de compartiment unique.

```
apiVersion: s3.services.k8s.aws/v1alpha1
kind: Bucket
metadata:
  name: my-test-bucket
  namespace: default
spec:
  name: my-unique-bucket-name-12345
```

Appliquez le manifeste :

```
kubectl apply -f bucket.yaml
```

Vérifiez le statut :

```
kubectl get bucket my-test-bucket
kubectl describe bucket my-test-bucket
```

Vérifiez que le bucket a été créé dans AWS :

```
aws s3 ls | grep my-unique-bucket-name-12345
```

Supprimez la ressource Kubernetes :

```
kubectl delete bucket my-test-bucket
```

Vérifiez que le bucket a été supprimé de AWS :

```
aws s3 ls | grep my-unique-bucket-name-12345
```

Le bucket ne doit plus apparaître dans la liste, ce qui prouve qu'ACK gère le cycle de vie complet des AWS ressources.

Pour plus d'informations sur la prise en main d'ACK, consultez Getting Started with ACK.

Cycle de vie et réconciliation des ressources

ACK utilise une boucle de réconciliation continue pour garantir que vos AWS ressources correspondent à l'état souhaité défini dans vos manifestes Kubernetes.

Comment fonctionne la réconciliation :

1. Vous créez ou mettez à jour une ressource personnalisée Kubernetes (par exemple, un compartiment S3)
2. ACK détecte le changement et compare l'état souhaité avec l'état réel dans AWS
3. S'ils diffèrent, ACK effectue des appels d' AWS API pour concilier la différence
4. ACK met à jour le statut des ressources dans Kubernetes pour refléter l'état actuel
5. La boucle se répète en continu, généralement toutes les quelques heures

La réconciliation est déclenchée lorsque vous créez une nouvelle ressource Kubernetes, que vous mettez à jour une ressource existante ou lorsque ACK détecte une dérive due à des modifications

manuelles effectuées en AWS dehors d'ACK. `spec` En outre, ACK effectue des rapprochements périodiques avec une période de resynchronisation de 10 heures. Les modifications apportées aux ressources Kubernetes déclenchent une réconciliation immédiate, tandis que la détection passive des modifications des AWS ressources en amont se produit lors de la resynchronisation périodique.

Lorsque vous suivez l'exemple de démarrage ci-dessus, ACK exécute les étapes suivantes :

1. Vérifie si le bucket existe dans AWS
2. Si ce n'est pas le cas, appelez `s3:CreateBucket`
3. Met à jour le statut de Kubernetes avec l'ARN et l'état du bucket
4. Poursuite de la surveillance de la dérive

Pour en savoir plus sur le fonctionnement d'ACK, consultez ACK Reconciliation.

Conditions relatives au statut

Les ressources ACK utilisent des conditions d'état pour communiquer leur état. La compréhension de ces conditions vous aide à résoudre les problèmes et à comprendre l'état des ressources.

- Prêt : indique que la ressource est prête à être consommée (condition Kubernetes standardisée).
- ACK. ResourceSynced: indique que la spécification de la ressource correspond à l'état de la AWS ressource.
- ACK.terminal : indique qu'une erreur irrécupérable s'est produite.
- ACK.adopted : indique que la ressource a été adoptée à partir d'une AWS ressource existante plutôt que créée.
- Ack.Recoverable : indique une erreur récupérable qui peut être résolue sans mettre à jour les spécifications.
- ACK.advisory : fournit des informations consultatives sur la ressource.
- ACK. LateInitialized: indique si l'initialisation tardive des champs est terminée.
- ACK. ReferencesResolved: indique si tous les `AWSResourceReference` champs ont été résolus.
- ACK. IAMRoleSélectionné : indique si un IAMRole sélecteur a été sélectionné pour gérer cette ressource.

Vérifiez l'état de la ressource :

```
# Check if resource is ready
kubectl get bucket my-bucket -o jsonpath='{.status.conditions[?
(@.type=="Ready")].status}'

# Check for terminal errors
kubectl get bucket my-bucket -o jsonpath='{.status.conditions[?
(@.type=="ACK.Terminal")]}'
```

Exemple de statut :

```
status:
  conditions:
  - type: Ready
    status: "True"
    lastTransitionTime: "2024-01-15T10:30:00Z"
  - type: ACK.ResourceSynced
    status: "True"
    lastTransitionTime: "2024-01-15T10:30:00Z"
  - type: ACK.Terminal
    status: "True"
  ackResourceMetadata:
    arn: arn:aws:s3:::my-unique-bucket-name
    ownerAccountID: "111122223333"
    region: us-west-2
```

Pour en savoir plus sur le statut et les conditions de l'ACK, consultez la section Conditions de l'ACK.

Politiques de suppression

La politique de suppression d'ACK contrôle ce qui arrive aux AWS ressources lorsque vous supprimez la ressource Kubernetes.

Supprimer (par défaut)

La AWS ressource est supprimée lorsque vous supprimez la ressource Kubernetes : il s'agit du comportement par défaut.

```
# No annotation needed - this is the default
apiVersion: s3.services.k8s.aws/v1alpha1
kind: Bucket
metadata:
  name: temp-bucket
```

```
spec:
  name: temporary-bucket
```

La suppression de cette ressource entraîne la suppression du compartiment S3 dans AWS.

Conserver

La AWS ressource est conservée lorsque vous supprimez la ressource Kubernetes :

```
apiVersion: s3.services.k8s.aws/v1alpha1
kind: Bucket
metadata:
  name: important-bucket
  annotations:
    services.k8s.aws/deletion-policy: "retain"
spec:
  name: production-data-bucket
```

La suppression de cette ressource la supprime de Kubernetes mais laisse le compartiment S3 dedans. AWS

Cette `retain` politique est utile pour les bases de données de production qui doivent survivre à la ressource Kubernetes, pour les ressources partagées utilisées par plusieurs applications, pour les ressources contenant des données importantes qui ne doivent pas être supprimées accidentellement ou pour la gestion temporaire d'ACK dans le cadre de laquelle vous adoptez une ressource, la configurez, puis la relancez à la gestion manuelle.

Pour en savoir plus sur la politique de suppression d'ACK, consultez la section Politique de suppression d'ACK.

Adoption des ressources

L'adoption vous permet de placer les AWS ressources existantes sous la gestion d'ACK sans les recréer.

Quand utiliser l'adoption :

- Migration de l'infrastructure existante vers la gestion ACK
- Récupération de AWS ressources orphelines en cas de suppression accidentelle de ressources dans Kubernetes

- Importation de ressources créées par d'autres outils (CloudFormation, Terraform)

Comment fonctionne l'adoption :

```
apiVersion: s3.services.k8s.aws/v1alpha1
kind: Bucket
metadata:
  name: existing-bucket
  annotations:
    services.k8s.aws/adoption-policy: "adopt-or-create"
spec:
  name: my-existing-bucket-name
```

Lorsque vous créez cette ressource :

1. ACK vérifie si un bucket portant ce nom existe dans AWS
2. S'il est trouvé, ACK l'adopte (aucun appel d'API à créer)
3. ACK lit la configuration actuelle depuis AWS
4. ACK met à jour le statut de Kubernetes pour refléter l'état réel
5. Les mises à jour futures réconcilient normalement la ressource

Une fois adoptées, les ressources sont gérées comme n'importe quelle autre ressource ACK, et la suppression de la ressource Kubernetes entraîne la suppression de la AWS ressource, sauf si vous utilisez la `retain` politique de suppression.

Lors de l'adoption de ressources, AWS celles-ci doivent déjà exister et ACK a besoin d'autorisations de lecture pour les découvrir. La `adopt-or-create` politique adopte la ressource si elle existe ou la crée si elle n'existe pas. Cela est utile lorsque vous souhaitez un flux de travail déclaratif qui fonctionne, que la ressource existe ou non.

Pour en savoir plus sur l'adoption des ressources ACK, consultez la section Adoption des ressources ACK.

Ressources entre comptes et entre régions

ACK peut gérer les ressources de différents AWS comptes et régions à partir d'un seul cluster.

Annotations de ressources interrégionales

Vous pouvez spécifier la région d'une AWS ressource à l'aide d'une annotation :

```
apiVersion: s3.services.k8s.aws/v1alpha1
kind: Bucket
metadata:
  name: eu-bucket
  annotations:
    services.k8s.aws/region: eu-west-1
spec:
  name: my-eu-bucket
```

Vous pouvez également spécifier la région de toutes les AWS ressources créées dans un espace de noms donné :

Annotations relatives à l'espace de noms

Définissez une région par défaut pour toutes les ressources d'un espace de noms :

```
apiVersion: v1
kind: Namespace
metadata:
  name: production
  annotations:
    services.k8s.aws/default-region: us-west-2
```

Les ressources créées dans cet espace de noms utilisent cette région sauf si elles sont remplacées par une annotation au niveau des ressources.

Compte croisé

Utilisez les sélecteurs de rôles IAM pour associer des rôles IAM spécifiques à des espaces de noms :

```
apiVersion: services.k8s.aws/v1alpha1
kind: IAMRoleSelector
metadata:
  name: target-account-config
spec:
  arn: arn:aws:iam::444455556666:role/ACKTargetAccountRole
  namespaceSelector:
    names:
      - production
```

Les ressources créées dans l'espace de noms mappé utilisent automatiquement le rôle spécifié.

Pour en savoir plus sur les sélecteurs de rôles IAM, consultez la section Gestion des ressources entre comptes ACK. Pour plus de détails sur la configuration entre comptes, voirthe section called "Configurer les autorisations ".

Gestion des erreurs et comportement des nouvelles tentatives

ACK gère automatiquement les erreurs transitoires et réessaie les opérations qui ont échoué.

Réessayez la stratégie :

- Les erreurs transitoires (limitation du débit, problèmes de service temporaires, autorisations insuffisantes) déclenchent de nouvelles tentatives automatiques
- Le recul exponentiel évite le surmenage AWS APIs
- Le nombre maximal de tentatives varie en fonction du type d'erreur
- Erreurs permanentes (paramètres non valides, conflits de noms de ressources), ne réessayez pas

Vérifiez l'état de la ressource pour obtenir des informations détaillées sur les erreurs en utilisant `kubectl describe` :

```
kubectl describe bucket my-bucket
```

Recherchez les conditions d'état avec des messages d'erreur, les événements indiquant les récentes tentatives de rapprochement et le `message` champ des conditions d'état expliquant les échecs. Les erreurs courantes incluent des autorisations IAM insuffisantes, des conflits de noms de ressources AWS, des valeurs de configuration non valides dans le `spec` et des quotas de AWS service dépassés.

Pour résoudre les erreurs courantes, consultezthe section called "Résolution des problèmes".

Composition des ressources avec kro

Pour composer et connecter plusieurs ressources ACK ensemble, utilisez la fonctionnalité EKS pour kro (Kube Resource Orchestrator). kro fournit un moyen déclaratif de définir des groupes de ressources, en transmettant la configuration entre les ressources afin de gérer simplement des modèles d'infrastructure complexes.

Pour des exemples détaillés de création de compositions de ressources personnalisées avec des ressources ACK, voir the section called "concepts kro"

Étapes suivantes

- the section called "Considérations"- Modèles et stratégies d'intégration spécifiques à EKS

Configurer les autorisations ACK

ACK nécessite des autorisations IAM pour créer et gérer AWS des ressources en votre nom. Cette rubrique explique comment IAM fonctionne avec ACK et fournit des conseils sur la configuration des autorisations pour différents cas d'utilisation.

Comment IAM travaille avec ACK

ACK utilise des rôles IAM pour s'authentifier auprès de vos ressources AWS et effectuer des actions sur celles-ci. Il existe deux manières de fournir des autorisations à ACK :

Rôle de capacité : rôle IAM que vous fournissez lors de la création de la fonctionnalité ACK. Ce rôle est utilisé par défaut pour toutes les opérations ACK.

Sélecteurs de rôles IAM : rôles IAM supplémentaires pouvant être mappés à des espaces de noms ou à des ressources spécifiques. Ces rôles remplacent le rôle de capacité pour les ressources relevant de leur champ d'application.

Lorsqu'ACK doit créer ou gérer une ressource, il détermine le rôle IAM à utiliser :

1. Vérifiez si un IAMRole sélecteur correspond à l'espace de noms de la ressource
2. Si une correspondance est trouvée, supposons que le rôle IAM
3. Sinon, utilisez le rôle de capacité

Cette approche permet une gestion flexible des autorisations, qu'il s'agisse de configurations simples à rôle unique ou de configurations complexes impliquant plusieurs comptes et plusieurs équipes.

Mise en route : configuration simple des autorisations

Pour le développement, les tests ou les cas d'utilisation simples, vous pouvez ajouter toutes les autorisations de service nécessaires directement au rôle de capacité.

Cette approche fonctionne bien lorsque :

- Vous commencez à utiliser ACK
- Toutes les ressources se trouvent dans le même AWS compte
- Une seule équipe gère toutes les ressources de l'ACK
- Vous êtes sûr que tous les utilisateurs d'ACK ont les mêmes autorisations

Bonnes pratiques de production : sélecteurs de rôles IAM

Pour les environnements de production, utilisez les sélecteurs de rôle IAM pour implémenter l'accès avec le moindre privilège et l'isolation au niveau de l'espace de noms.

Lorsque vous utilisez des sélecteurs de rôle IAM, le rôle de capacité n'a besoin que d'`sts:TagSession`autorisations pour assumer `sts:AssumeRole` les rôles spécifiques au service. Vous n'avez pas besoin d'ajouter d'autorisations de AWS service (comme S3 ou RDS) au rôle de capacité lui-même. Ces autorisations sont accordées aux rôles IAM individuels assumés par le rôle de capacité.

Choix entre les modèles d'autorisation :

Utilisez des autorisations directes (ajout d'autorisations de service au rôle de capacité) lorsque :

- Vous êtes sur le point de démarrer et vous souhaitez la configuration la plus simple
- Toutes les ressources se trouvent dans le même compte que votre cluster
- Vous avez des exigences en matière d'autorisations administratives à l'échelle du cluster
- Toutes les équipes peuvent partager les mêmes autorisations

Utilisez les sélecteurs de rôles IAM lorsque :

- Gestion des ressources sur plusieurs AWS comptes
- Des équipes ou des espaces de noms différents ont besoin d'autorisations différentes
- Vous avez besoin d'un contrôle d'accès précis par espace de noms
- Vous souhaitez suivre les pratiques de sécurité fondées sur le principe du moindre privilège

Vous pouvez commencer par des autorisations directes et migrer vers les sélecteurs de rôle IAM ultérieurement à mesure que vos besoins augmentent.

Pourquoi utiliser les sélecteurs de rôles IAM en production :

- Privilège minimal : chaque espace de noms ne reçoit que les autorisations dont il a besoin
- Isolement de l'équipe : l'équipe A ne peut pas utiliser accidentellement les autorisations de l'équipe B
- Audit simplifié : mappage clair de l'espace de noms qui utilise quel rôle
- Support multicompte : nécessaire pour gérer les ressources de plusieurs comptes
- Séparation des préoccupations : différents services ou environnements utilisent des rôles différents

Configuration de base du sélecteur de rôle IAM

Étape 1 : créer un rôle IAM spécifique au service

Créez un rôle IAM avec des autorisations pour des AWS services spécifiques :

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:*"
      ],
      "Resource": "*"
    }
  ]
}
```

Configurez la politique de confiance pour permettre au rôle de capacité de l'assumer :

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/ACKCapabilityRole"
      },
      "Action": ["sts:AssumeRole", "sts:TagSession"]
    }
```

```
  ]
}
```

Étape 2 : Accorder AssumeRole l'autorisation à Capability Role

Ajoutez l'autorisation au rôle de capacité pour assumer le rôle spécifique au service :

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["sts:AssumeRole", "sts:TagSession"],
      "Resource": "arn:aws:iam::111122223333:role/ACK-S3-Role"
    }
  ]
}
```

Étape 3 : créer un IAMRole sélecteur

Mappez le rôle IAM à un espace de noms :

```
apiVersion: services.k8s.aws/v1alpha1
kind: IAMRoleSelector
metadata:
  name: s3-namespace-config
spec:
  arn: arn:aws:iam::111122223333:role/ACK-S3-Role
  namespaceSelector:
    names:
      - s3-resources
```

Étape 4 : créer des ressources dans l'espace de noms mappé

Les ressources de l'espace de `s3-resources` noms utilisent automatiquement le rôle spécifié :

```
apiVersion: s3.services.k8s.aws/v1alpha1
kind: Bucket
metadata:
  name: my-bucket
  namespace: s3-resources
spec:
```

```
  name: my-production-bucket
```

Gestion de plusieurs comptes

Utilisez les sélecteurs de rôle IAM pour gérer les ressources de plusieurs AWS comptes.

Étape 1 : créer un rôle IAM entre comptes

Dans le compte cible (444455556666), créez un rôle qui fait confiance au rôle de capacité du compte source :

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/ACKCapabilityRole"
      },
      "Action": ["sts:AssumeRole", "sts:TagSession"]
    }
  ]
}
```

Associez des autorisations spécifiques au service à ce rôle.

Étape 2 : Accorder AssumeRole l'autorisation

Dans le compte source (111122223333), autorisez le rôle de capacité à assumer le rôle de compte cible :

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["sts:AssumeRole", "sts:TagSession"],
      "Resource": "arn:aws:iam::444455556666:role/ACKTargetAccountRole"
    }
  ]
}
```

Étape 3 : créer un IAMRole sélecteur

Associez le rôle multicompte à un espace de noms :

```
apiVersion: services.k8s.aws/v1alpha1
kind: IAMRoleSelector
metadata:
  name: production-account-config
spec:
  arn: arn:aws:iam::444455556666:role/ACKTargetAccountRole
  namespaceSelector:
    names:
      - production
```

Étape 4 : Création de ressources

Les ressources de l'espace de `production` noms sont créées dans le compte cible :

```
apiVersion: s3.services.k8s.aws/v1alpha1
kind: Bucket
metadata:
  name: my-bucket
  namespace: production
spec:
  name: my-cross-account-bucket
```

Balises de session

La fonctionnalité EKS ACK définit automatiquement les balises de session pour toutes les demandes AWS d'API. Ces balises permettent un contrôle d'accès et un audit précis en identifiant la source de chaque demande.

Tags de session disponibles

Les balises de session suivantes sont incluses dans chaque appel AWS d'API effectué par ACK :

Clé de tag	Description
`eks:eks-capability-arn`	L'ARN de la capacité EKS à l'origine de la demande
`eks:kubernetes-namespace`	L'espace de noms Kubernetes de la ressource gérée

Clé de tag	Description
`eks:kubernetes-api-group`	Le groupe d'API Kubernetes de la ressource (par exemple,) `s3.services.k8s.aws`

Utilisation de balises de session pour le contrôle d'accès

Vous pouvez utiliser ces balises de session dans les conditions de politique IAM pour limiter les ressources qu'ACK peut gérer. Cela fournit une couche de sécurité supplémentaire au-delà des sélecteurs de rôles IAM basés sur des espaces de noms.

Exemple : Restreindre par espace de noms

Autorisez ACK à créer des compartiments S3 uniquement lorsque la demande provient de l'`production`espace de noms :

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:CreateBucket",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:PrincipalTag/eks:kubernetes-namespace": "production"
        }
      }
    }
  ]
}
```

Exemple : Restreindre en fonction de la capacité

Autoriser les actions uniquement à partir d'une fonctionnalité ACK spécifique :

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

```
      "Effect": "Allow",
      "Action": "s3:*",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:PrincipalTag/eks:eks-capability-arn": "arn:aws:eks:us-
west-2:111122223333:capability/my-cluster/ack/my-ack"
        }
      }
    }
  ]
}
```

> **Note**
>
> Les balises de session sont différentes de l'ACK autogéré, qui ne définit pas ces balises par défaut. Cela permet un contrôle d'accès plus granulaire grâce à la fonctionnalité gérée.

Modèles de sélection de rôles IAM avancés

Pour une configuration avancée, y compris les sélecteurs d'étiquettes, le mappage des rôles spécifiques aux ressources et des exemples supplémentaires, consultez la documentation ACK IRSA.

Étapes suivantes

- the section called “Concepts d'ACK”- Comprendre les concepts ACK et le cycle de vie des ressources
- the section called “Concepts d'ACK”- En savoir plus sur les politiques d'adoption et de suppression des ressources
- the section called “Considérations relatives aux fonctionnalités d'EKS”- Comprendre les meilleures pratiques en matière de sécurité en termes de fonctionnalités

Considérations relatives à l'ACK pour EKS

Cette rubrique couvre les considérations importantes relatives à l'utilisation de la fonctionnalité EKS pour ACK, notamment la configuration IAM, les modèles multi-comptes et l'intégration avec d'autres fonctionnalités EKS.

Modèles de configuration IAM

La fonctionnalité ACK utilise un rôle de capacité IAM pour s'authentifier. AWS Choisissez le modèle IAM adapté à vos besoins.

Simple : rôle de capacité unique

Pour le développement, les tests ou les cas d'utilisation simples, accordez toutes les autorisations nécessaires directement au rôle de capacité.

Quand utiliser :

- Commencer à utiliser ACK
- Déploiements à compte unique
- Toutes les ressources sont gérées par une seule équipe
- Environnements de développement et de test

Exemple : ajoutez des autorisations S3 et RDS à votre rôle de capacité avec des conditions de balisage des ressources :

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["s3:*"],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:RequestedRegion": ["us-west-2", "us-east-1"]
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": ["rds:*"],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:RequestedRegion": ["us-west-2", "us-east-1"]
        },
```

```
      }
    }
  ]
}
```

Cet exemple limite les opérations S3 et RDS à des régions spécifiques et exige que les ressources RDS soient dotées d'une `ManagedBy: ACK` balise.

Production : Sélecteurs de rôles IAM

Pour les environnements de production, utilisez les sélecteurs de rôle IAM pour implémenter l'accès avec le moindre privilège et l'isolation au niveau de l'espace de noms.

Quand utiliser :

- Environnements de production
- Clusters multi-équipes
- Gestion des ressources multi-comptes
- Exigences de sécurité relatives au moindre privilège
- Les différents services nécessitent des autorisations différentes

Avantages :

- Chaque espace de noms ne reçoit que les autorisations dont il a besoin
- Isolement de l'équipe - L'équipe A ne peut pas utiliser les autorisations de l'équipe B
- Audit et conformité simplifiés
- Nécessaire pour la gestion des ressources entre comptes

Pour une configuration détaillée du sélecteur de rôle IAM, consultez. the section called “Configurer les autorisations ”

Intégration avec d'autres fonctionnalités d'EKS

GitOps avec Argo CD

Utilisez le CD EKS Capability for Argo pour déployer des ressources ACK à partir de référentiels Git, permettant ainsi des GitOps flux de travail pour la gestion de l'infrastructure.

Considérations :

- Stockez les ressources ACK à côté des manifestes d'applications pour end-to-end GitOps
- Organisez par environnement, service ou type de ressource en fonction de la structure de votre équipe
- Utilisez la synchronisation automatique d'Argo CD pour un rapprochement continu
- Activez l'élagage pour supprimer automatiquement les ressources supprimées
- Envisagez hub-and-spoke des modèles de gestion de l'infrastructure multi-clusters

GitOps fournit des pistes d'audit, des fonctionnalités de restauration et une gestion déclarative de l'infrastructure. Pour en savoir plus sur Argo CD, voirthe section called “Travailler avec Argo CD”.

Composition des ressources avec kro

Utilisez la fonctionnalité EKS pour kro (Kube Resource Orchestrator) pour composer plusieurs ressources ACK en abstractions de niveau supérieur et personnalisées. APIs

Quand utiliser Kro avec ACK :

- Créez des modèles réutilisables pour les piles d'infrastructure courantes (base de données, sauvegarde et surveillance)
- Créez des plateformes en libre-service simplifiées APIs pour les équipes d'application
- Gérer les dépendances des ressources et transmettre des valeurs entre les ressources (fonction ARN du compartiment S3 vers Lambda)
- Standardisez les configurations d'infrastructure entre les équipes
- Réduisez la complexité en masquant les détails de mise en œuvre derrière des ressources personnalisées

Exemples de modèles :

- Pile d'applications : compartiment S3, file d'attente SQS et configuration des notifications
- Configuration de la base de données : instance RDS + groupe de paramètres + groupe de sécurité + secrets
- Mise en réseau : VPC + sous-réseaux + tables de routage + groupes de sécurité

kro gère l'ordre des dépendances, la propagation des statuts et la gestion du cycle de vie des ressources composées. Pour en savoir plus sur Kro, voirthe section called “concepts kro”.

Organisation de vos ressources

Organisez les ressources ACK à l'aide des espaces de noms et des balises de AWS ressources Kubernetes pour améliorer la gestion, le contrôle d'accès et le suivi des coûts.

Organisation de l'espace de noms

Utilisez les espaces de noms Kubernetes pour séparer logiquement les ressources ACK par environnement (production, développement, développement), par équipe (plateforme, données, ml) ou par application.

Avantages :

- RBAC limité à l'espace de noms pour le contrôle d'accès
- Définissez les régions par défaut par espace de noms à l'aide d'annotations
- Gestion et nettoyage des ressources simplifiés
- Séparation logique alignée sur la structure organisationnelle

Étiquette des ressources

La fonctionnalité EKS ACK applique automatiquement des balises par défaut à toutes les AWS ressources qu'elle crée. Ces étiquettes diffèrent de l'ACK autogéré et offrent une meilleure traçabilité.

Balises par défaut appliquées par la fonctionnalité :

Clé de tag	Description
`eks:controller-version`	La version du contrôleur ACK
`eks:kubernetes-namespace`	L'espace de noms Kubernetes de la ressource ACK
`eks:kubernetes-resource-name`	Le nom de la ressource Kubernetes
`eks:kubernetes-api-group`	Le groupe d'API Kubernetes (par exemple,) `s3.services.k8s.aws`

Clé de tag	Description
`eks:eks-capability-arn`	L'ARN de la fonctionnalité EKS ACK

> **Note**
>
> L'ACK autogéré utilise différentes balises par défaut : `services.k8s.aws/controller-version` et`services.k8s.aws/namespace`. Les balises de la fonctionnalité utilisent le `eks:` préfixe par souci de cohérence avec les autres fonctionnalités d'EKS.

Tags supplémentaires recommandés :

Ajoutez des balises personnalisées pour la répartition des coûts, le suivi de la propriété et à des fins organisationnelles :

- Environnement (production, mise en scène, développement)
- Propriété d'une équipe ou d'un département
- Centre de coûts pour la répartition de la facturation
- Nom de l'application ou du service

Migration depuis d'autres Infrastructure-as-code outils

De nombreuses entreprises trouvent la valeur de la standardisation sur Kubernetes au-delà de l'orchestration de leur charge de travail. La migration de la gestion de l'infrastructure et AWS des ressources vers ACK vous permet de standardiser la gestion de l'infrastructure à l'aide de Kubernetes APIs parallèlement aux charges de travail de vos applications.

Avantages de la standardisation sur Kubernetes pour l'infrastructure :

- Source unique de vérité : gérez à la fois les applications et l'infrastructure dans Kubernetes, permettant ainsi une pratique end-to-end GitOps
- Outillage unifié : les équipes utilisent les ressources et les outils Kubernetes plutôt que d'apprendre plusieurs outils et frameworks

- Réconciliation cohérente : ACK réconcilie en permanence les AWS ressources, comme le fait Kubernetes pour les charges de travail, en détectant et en corrigeant les dérives par rapport aux outils indispensables
- Compositions natives : avec kro et ACK combinés, référencez les AWS ressources directement dans les manifestes d'applications et de ressources, en passant des chaînes de connexion et ARNs entre les ressources
- Opérations simplifiées : un seul plan de contrôle pour les déploiements, les annulations et l'observabilité sur l'ensemble de votre système

ACK prend en charge l'adoption AWS des ressources existantes sans les recréer, ce qui permet une migration sans interruption depuis CloudFormation Terraform ou des ressources externes au cluster.

Adoptez une ressource existante :

```
apiVersion: s3.services.k8s.aws/v1alpha1
kind: Bucket
metadata:
  name: existing-bucket
  annotations:
    services.k8s.aws/adoption-policy: "adopt-or-create"
spec:
  name: my-existing-bucket-name
```

Une fois adoptée, la ressource est gérée par ACK et peut être mise à jour via les manifestes Kubernetes. Vous pouvez effectuer une migration incrémentielle, en adoptant des ressources selon vos besoins, tout en conservant les outils IaC existants pour d'autres ressources.

ACK prend également en charge les ressources en lecture seule. Pour les ressources gérées par d'autres équipes ou les outils que vous souhaitez référencer mais ne pas modifier, combinez l'adoption avec la politique de `retain` suppression et accordez uniquement des autorisations de lecture IAM. Cela permet aux applications de découvrir une infrastructure partagée (rôles IAMVPCs, clés KMS) via Kubernetes APIs sans risquer de modifications.

Pour en savoir plus sur l'adoption des ressources, voirthe section called “Concepts d'ACK”.

Politiques de suppression

Les politiques de suppression contrôlent ce qu'il advient AWS des ressources lorsque vous supprimez la ressource Kubernetes correspondante. Choisissez la bonne politique en fonction du cycle de vie des ressources et de vos exigences opérationnelles.

Supprimer (par défaut)

La AWS ressource est supprimée lorsque vous supprimez la ressource Kubernetes. Cela permet de maintenir la cohérence entre votre cluster et AWS de garantir que les ressources ne s'accumulent pas.

Quand utiliser la suppression :

- Environnements de développement et de test dans lesquels le nettoyage est important
- Ressources éphémères liées au cycle de vie des applications (bases de données de test, compartiments temporaires)
- Ressources qui ne devraient pas durer plus longtemps que l'application (files d'attente SQS, clusters) ElastiCache
- Optimisation des coûts : nettoyage automatique des ressources inutilisées
- Environnements gérés avec GitOps lesquels la suppression des ressources de Git devrait supprimer l'infrastructure

La politique de suppression par défaut s'aligne sur le modèle déclaratif de Kubernetes : le contenu du cluster correspond à ce qui existe dans. AWS

Conserver

La AWS ressource est conservée lorsque vous supprimez la ressource Kubernetes. Cela protège les données critiques et permet aux ressources de survivre à leur représentation sur Kubernetes.

Quand utiliser Retain :

- Bases de données de production contenant des données critiques qui doivent résister aux modifications des clusters
- Compartiments de stockage à long terme soumis à des exigences de conformité ou d'audit
- Ressources partagées utilisées par plusieurs applications ou équipes
- Migration des ressources vers différents outils de gestion

- Scénarios de reprise après sinistre dans lesquels vous souhaitez préserver l'infrastructure
- Ressources présentant des dépendances complexes qui nécessitent une mise hors service minutieuse

```
apiVersion: rds.services.k8s.aws/v1alpha1
kind: DBInstance
metadata:
  name: production-db
  annotations:
    services.k8s.aws/deletion-policy: "retain"
spec:
  dbInstanceIdentifier: prod-db
  # ... configuration
```

> **Important**
>
> Les ressources conservées continuent d'entraîner des AWS coûts et doivent être supprimées manuellement AWS lorsqu'elles ne sont plus nécessaires. Utilisez le balisage des ressources pour suivre les ressources conservées à des fins de nettoyage.

Pour en savoir plus sur les politiques de suppression, consultezthe section called “Concepts d'ACK”.

Documentation en amont

Pour des informations détaillées sur l'utilisation d'ACK :

- Guide d'utilisation d'ACK - Création et gestion de ressources
- Référence de l'API ACK - Documentation complète sur les API pour tous les services
- Documentation ACK - Documentation utilisateur complète

Étapes suivantes

- the section called “Configurer les autorisations ”- Configurer les autorisations IAM et les modèles multi-comptes
- the section called “Concepts d'ACK”- Comprendre les concepts ACK et le cycle de vie des ressources

- the section called "Résolution des problèmes"- Résoudre les problèmes d'ACK
- the section called "Travailler avec Argo CD"- Déployez les ressources ACK avec GitOps
- the section called "concepts kro"- Composez les ressources ACK en abstractions de niveau supérieur

Résoudre les problèmes liés aux fonctionnalités ACK

Cette rubrique fournit des conseils de dépannage relatifs à la fonctionnalité EKS pour ACK, notamment les vérifications de l'état des fonctionnalités, la vérification de l'état des ressources et les problèmes d'autorisation IAM.

> **Note**
>
> Les fonctionnalités EKS sont entièrement gérées et exécutées en dehors de votre cluster. Vous n'avez pas accès aux journaux ni aux espaces de noms des contrôleurs. Le dépannage se concentre sur l'état des capacités, l'état des ressources et la configuration IAM.

La capacité est ACTIVE mais les ressources ne sont pas créées

Si votre fonctionnalité ACK affiche un `ACTIVE` état mais que les ressources n'y sont pas créées AWS, vérifiez l'état de la fonctionnalité, l'état des ressources et les autorisations IAM.

Vérifiez l'état des capacités :

Vous pouvez consulter les problèmes d'état et d'état des fonctionnalités dans la console EKS ou à l'aide de la AWS CLI.

Console :

1. Ouvrez la console Amazon EKS à l'adresse https://console.aws.amazon.com/eks/ home#/clusters.
2. Sélectionnez le nom de votre cluster.
3. Sélectionnez l'onglet Observabilité.
4. Sélectionnez Surveiller le cluster.
5. Choisissez l'onglet Fonctionnalités pour afficher l'état et l'état de toutes les fonctionnalités.

AWS CLI :

```
# View capability status and health
aws eks describe-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name my-ack

# Look for issues in the health section
```

Causes courantes :

- Autorisations IAM manquantes : le rôle de capacité ne dispose pas d'autorisations pour le service AWS
- Mauvais espace de noms : ressources créées dans l'espace de noms sans sélecteur approprié IAMRole
- Spécification de ressource non valide : vérifiez les conditions d'état des ressources pour détecter les erreurs de validation
- Limitation de l'API : les limites de débit des AWS API sont atteintes
- Webhooks d'admission : webhooks d'admission empêchant le contrôleur de corriger le statut des ressources

Vérifiez l'état des ressources :

```
# Describe the resource to see conditions and events
kubectl describe bucket my-bucket -n default

# Look for status conditions
kubectl get bucket my-bucket -n default -o jsonpath='{.status.conditions}'

# View resource events
kubectl get events --field-selector involvedObject.name=my-bucket -n default
```

Vérifiez les autorisations IAM :

```
# View the Capability Role's policies
aws iam list-attached-role-policies --role-name my-ack-capability-role
aws iam list-role-policies --role-name my-ack-capability-role

# Get specific policy details
aws iam get-role-policy --role-name my-ack-capability-role --policy-name policy-name
```

Ressources créées dans Kubernetes AWS mais non affichées

ACK suit uniquement les ressources qu'il crée via les manifestes Kubernetes. Pour gérer les AWS ressources existantes avec ACK, utilisez la fonctionnalité d'adoption.

```
apiVersion: s3.services.k8s.aws/v1alpha1
kind: Bucket
metadata:
  name: existing-bucket
  annotations:
    services.k8s.aws/adoption-policy: "adopt-or-create"
spec:
  name: my-existing-bucket-name
```

Pour en savoir plus sur l'adoption des ressources, voirthe section called “Concepts d'ACK”.

Les ressources entre comptes ne sont pas créées

Si aucune ressource n'est créée dans un AWS compte cible lors de l'utilisation des sélecteurs de rôle IAM, vérifiez la relation de confiance et la configuration du IAMRole sélecteur.

Vérifiez la relation de confiance :

```
# Check the trust policy in the target account role
aws iam get-role --role-name cross-account-ack-role --query
 'Role.AssumeRolePolicyDocument'
```

La politique de confiance doit permettre au rôle de capacité du compte source de l'assumer.

Confirmez la configuration IAMRole du sélecteur :

```
# List IAMRoleSelectors (cluster-scoped)
kubectl get iamroleselector

# Describe specific selector
kubectl describe iamroleselector my-selector
```

Vérifiez l'alignement de l'espace de noms :

IAMRoleLes sélecteurs sont des ressources limitées à un cluster mais ciblent des espaces de noms spécifiques. Assurez-vous que vos ressources ACK se trouvent dans un espace de noms qui correspond au IAMRole sélecteur d'espace de noms du sélecteur :

```
# Check resource namespace
kubectl get bucket my-cross-account-bucket -n production

# List all IAMRoleSelectors (cluster-scoped)
kubectl get iamroleselector

# Check which namespace the selector targets
kubectl get iamroleselector my-selector -o jsonpath='{.spec.namespaceSelector}'
```

Vérifiez la condition IAMRole sélectionnée :

Vérifiez que le IAMRole sélecteur a bien été mis en correspondance avec votre ressource en vérifiant la `ACK.IAMRoleSelected` condition suivante :

```
# Check if IAMRoleSelector was matched
kubectl get bucket my-cross-account-bucket -n production -o
 jsonpath='{.status.conditions[?(@.type=="ACK.IAMRoleSelected")]}'
```

Si la condition est `False` ou absente, le IAMRole sélecteur d'espace de noms du sélecteur ne correspond pas à l'espace de noms de la ressource. Vérifiez que le sélecteur `namespaceSelector` correspond aux libellés d'espace de noms de votre ressource.

Vérifiez les autorisations relatives aux rôles de capacité :

Le rôle de capacité a besoin `sts:AssumeRole` et `sts:TagSession` autorisations pour le rôle de compte cible :

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["sts:AssumeRole", "sts:TagSession"],
      "Resource": "arn:aws:iam::[.replaceable]`444455556666`:role/[.replaceable]`cross-
account-ack-role`"
    }
  ]
```

```
}
```

Pour une configuration détaillée entre comptes, voirthe section called “Configurer les autorisations ”.

Étapes suivantes

- the section called “Considérations”- Considérations et meilleures pratiques relatives à l'ACK
- the section called “Configurer les autorisations ”- Configurer les autorisations IAM et les modèles multi-comptes
- the section called “Concepts d'ACK”- Comprendre les concepts ACK et le cycle de vie des ressources
- the section called “Fonctionnalités de résolution des problèmes”- Conseils généraux de résolution des problèmes liés aux fonctionnalités

Comparaison de la capacité EKS pour ACK à une solution ACK autogérée

La capacité EKS pour ACK fournit les mêmes fonctionnalités que les contrôleurs ACK autogérés, mais avec des avantages opérationnels significatifs. Pour une comparaison générale des fonctionnalités EKS par rapport aux solutions autogérées, voirthe section called “Considérations”. Cette rubrique se concentre sur les différences spécifiques à ACK.

Différences par rapport à l'ACK en amont

La capacité EKS pour ACK est basée sur les contrôleurs ACK en amont, mais elle diffère en termes d'intégration IAM.

Rôle de capacité IAM : la fonctionnalité utilise un rôle IAM dédié avec une politique de confiance qui autorise le principal du `capabilities.eks.amazonaws.com` service, et non l'IRSA (rôles IAM pour les comptes de service). Vous pouvez associer des politiques IAM directement au rôle de capacité sans avoir à créer ou à annoter des comptes de service Kubernetes ou à configurer des fournisseurs OIDC. Une bonne pratique pour les cas d'utilisation en production consiste à configurer les autorisations de service à l'aide de`IAMRoleSelector`. Pour plus d'informations, consultez the section called “Configurer les autorisations ”.

Balises de session : la fonctionnalité gérée définit automatiquement les balises de session pour toutes les demandes d' AWS API, ce qui permet un contrôle d'accès et un audit précis. Les balises incluent `eks:eks-capability-arn`eks:kubernetes-namespace, et`eks:kubernetes-api-group`. Cela diffère de l'ACK autogéré, qui ne définit pas ces balises par défaut. Consultez

the section called "Configurer les autorisations " pour plus de détails sur l'utilisation des balises de session dans les politiques IAM.

Balises de ressources : la fonctionnalité applique des balises par défaut différentes aux AWS ressources par rapport à l'ACK autogéré. La fonctionnalité utilise des balises `eks:` préfixées (telles que`eks:kubernetes-namespace`,`eks:eks-capability-arn`) au lieu des `services.k8s.aws/` balises utilisées par ACK autogéré. Consultez the section called "Considérations" la liste complète des balises de ressources par défaut.

Compatibilité des ressources : les ressources personnalisées ACK fonctionnent de la même manière que les ressources ACK en amont, sans aucune modification de vos fichiers YAML de ressources ACK. La fonctionnalité utilise les mêmes Kubernetes APIs et CRDs les outils `kubectl` fonctionnent donc de la même manière. Tous les contrôleurs GA et les ressources de l'ACK en amont sont pris en charge.

Pour consulter la documentation complète d'ACK et les guides spécifiques aux services, consultez la documentation ACK.

Voie de migration

Vous pouvez passer d'un ACK autogéré à une fonctionnalité gérée sans aucune interruption de service :

1. Mettez à jour votre contrôleur ACK autogéré pour l'utiliser `kube-system` pour les baux électoraux des dirigeants, par exemple :

   ```
   helm upgrade --install ack-s3-controller \
     oci://public.ecr.aws/aws-controllers-k8s/s3-chart \
     --namespace ack-system \
     --set leaderElection.namespace=kube-system
   ```

   Cela déplace le bail du contrôleur vers`kube-system`, ce qui permet à la capacité gérée de se coordonner avec celui-ci.
2. Créez la fonctionnalité ACK sur votre cluster (voirthe section called "Créer une fonctionnalité ACK")
3. La fonctionnalité gérée reconnaît les AWS ressources existantes gérées par ACK et prend en charge le rapprochement
4. Diminuez ou supprimez progressivement les déploiements de contrôleurs autogérés :

   ```
   helm uninstall ack-s3-controller --namespace ack-system
   ```

Cette approche permet aux deux contrôleurs de coexister en toute sécurité pendant la migration. La fonctionnalité gérée adopte automatiquement les ressources précédemment gérées par des contrôleurs autogérés, garantissant ainsi un rapprochement continu sans conflits.

Étapes suivantes

- the section called “Créer une fonctionnalité ACK”- Création d'une ressource de capacité ACK
- the section called “Concepts d'ACK”- Comprendre les concepts ACK et le cycle de vie des ressources
- the section called “Configurer les autorisations ”- Configurer l'IAM et les autorisations

Déploiement continu avec Argo CD

Argo CD est un outil de diffusion GitOps continue déclaratif pour Kubernetes. Avec Argo CD, vous pouvez automatiser le déploiement et la gestion du cycle de vie de vos applications sur plusieurs clusters et environnements. Argo CD prend en charge plusieurs types de sources, notamment les référentiels Git, les registres Helm (HTTP et OCI) et les images OCI, offrant ainsi de la flexibilité aux entreprises ayant des exigences de sécurité et de conformité différentes.

Grâce aux fonctionnalités d'EKS, Argo CD est entièrement géré par AWS, ce qui élimine le besoin d'installer, de maintenir et de dimensionner les contrôleurs Argo CD et leurs dépendances vis-à-vis de vos clusters.

Comment fonctionne Argo CD

Argo CD suit le GitOps modèle selon lequel la source de votre application (dépôt Git, registre Helm ou image OCI) est la source fiable pour définir l'état souhaité de l'application. Lorsque vous créez une `Application` ressource Argo CD, vous spécifiez la source contenant les manifestes de votre application ainsi que le cluster et l'espace de noms Kubernetes cibles. Argo CD surveille en permanence l'état source et l'état réel du cluster, synchronisant automatiquement les modifications pour s'assurer que l'état du cluster correspond à l'état souhaité.

> **Note**
>
> Grâce à la fonctionnalité EKS pour Argo CD, le logiciel Argo CD s'exécute dans le plan de AWS contrôle, et non sur vos nœuds de travail. Cela signifie que vos nœuds de travail n'ont

pas besoin d'un accès direct aux référentiels Git ou aux registres Helm : la fonctionnalité gère l'accès aux sources depuis le compte. AWS

Argo CD fournit trois types de ressources principaux :

- Application : définit un déploiement depuis un dépôt Git vers un cluster cible
- ApplicationSet: Génère plusieurs applications à partir de modèles pour les déploiements multi-clusters
- AppProject: fournit un regroupement logique et un contrôle d'accès pour les applications

Exemple : création d'une application Argo CD

L'exemple suivant montre comment créer une `Application` ressource Argo CD :

```
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: guestbook
  namespace: argocd
spec:
  project: default
  source:
    repoURL: https://github.com/argoproj/argocd-example-apps.git
    targetRevision: HEAD
    path: guestbook
  destination:
    name: in-cluster
    namespace: guestbook
  syncPolicy:
    automated:
      prune: true
      selfHeal: true
```

Note

`destination.name`À utiliser avec le nom du cluster que vous avez utilisé lors de l'enregistrement du cluster (comme `in-cluster` pour le cluster local). Le

`destination.server` champ fonctionne également avec le cluster EKS ARNs, mais il est recommandé d'utiliser des noms de clusters pour une meilleure lisibilité.

Avantages d'Argo CD

Argo CD implémente un GitOps flux de travail dans lequel vous définissez les configurations de vos applications dans les référentiels Git et Argo CD synchronise automatiquement vos applications pour qu'elles correspondent à l'état souhaité. Cette approche centrée sur le GIT fournit une piste d'audit complète de toutes les modifications, permet des annulations faciles et s'intègre naturellement à vos processus de révision et d'approbation de code existants. Argo CD détecte et réconcilie automatiquement la dérive entre l'état souhaité dans Git et l'état réel dans vos clusters, garantissant ainsi la cohérence de vos déploiements avec votre configuration déclarée.

Avec Argo CD, vous pouvez déployer et gérer des applications sur plusieurs clusters à partir d'une seule instance Argo CD, ce qui simplifie les opérations dans les environnements multiclusters et multirégions. L'interface utilisateur d'Argo CD fournit des fonctionnalités de visualisation et de surveillance, vous permettant de visualiser l'état de déploiement, l'état de santé et l'historique de vos applications. L'interface utilisateur s'intègre à AWS Identity Center (anciennement AWS SSO) pour une authentification et une autorisation fluides, vous permettant de contrôler l'accès à l'aide de votre infrastructure de gestion des identités existante.

Dans le cadre des fonctionnalités gérées par EKS, Argo CD est entièrement géré par AWS, ce qui élimine le besoin d'installer, de configurer et de maintenir l'infrastructure Argo CD. AWS gère le dimensionnement, l'application de correctifs et la gestion opérationnelle, permettant à vos équipes de se concentrer sur la livraison des applications plutôt que sur la maintenance des outils.

Intégration à AWS Identity Center

EKS Managed Capabilities fournit une intégration directe entre Argo CD et AWS Identity Center, permettant ainsi une authentification et une autorisation fluides pour vos utilisateurs. Lorsque vous activez la fonctionnalité Argo CD, vous pouvez configurer l'intégration AWS d'Identity Center pour mapper les groupes et les utilisateurs d'Identity Center aux rôles RBAC d'Argo CD, ce qui vous permet de contrôler qui peut accéder aux applications dans Argo CD et les gérer.

Intégration avec d'autres fonctionnalités gérées par EKS

Argo CD s'intègre aux autres fonctionnalités gérées par EKS.

- AWS Contrôleurs pour Kubernetes (ACK) : utilisez Argo CD pour gérer le déploiement des ressources ACK sur plusieurs clusters, en activant les GitOps flux de travail pour votre infrastructure. AWS
- kro (Kube Resource Orchestrator) : utilisez Argo CD pour déployer des compositions kro sur plusieurs clusters, permettant ainsi une composition cohérente des ressources dans l'ensemble de votre parc Kubernetes.

Commencer à utiliser Argo CD

Pour commencer à utiliser la fonctionnalité EKS pour Argo CD :

1. Créez et configurez un rôle de capacité IAM avec les autorisations nécessaires pour qu'Argo CD puisse accéder à vos sources et gérer les applications.
2. Créez une ressource de capacité Argo CD sur votre cluster EKS via la AWS console, la AWS CLI ou votre infrastructure préférée en tant qu'outil de code.
3. Configurez l'accès au référentiel et enregistrez les clusters pour le déploiement des applications.
4. Créez des ressources d'application pour déployer vos applications à partir de vos sources déclaratives.

Création d'une fonctionnalité Argo CD

Cette rubrique explique comment créer une fonctionnalité Argo CD sur votre cluster Amazon EKS.

Conditions préalables

Avant de créer une fonctionnalité Argo CD, assurez-vous de disposer des éléments suivants :

- Un cluster Amazon EKS existant exécutant une version de Kubernetes prise en charge (toutes les versions du support standard et étendu sont prises en charge)
- AWS Identity Center configuré - Nécessaire pour l'authentification sur Argo CD (les utilisateurs locaux ne sont pas pris en charge)
- Un rôle de capacité IAM avec des autorisations pour Argo CD
- Autorisations IAM suffisantes pour créer des ressources de capacités sur les clusters EKS
- `kubectl`configuré pour communiquer avec votre cluster
- (Facultatif) L'Argo CD CLI est installée pour faciliter la gestion des clusters et des référentiels

- (Pour CLI/ekSCTL) L'outil CLI approprié est installé et configuré

Pour obtenir des instructions sur la création du rôle de capacité IAM, consultezthe section called "Capacité : rôle IAM". Pour la configuration d'Identity Center, consultez Getting Started with AWS Identity Center.

> **Important**
>
> Le rôle de capacité IAM que vous fournissez détermine les AWS ressources auxquelles Argo CD peut accéder. Cela inclut l'accès au dépôt Git via CodeConnections et les secrets dans Secrets Manager. Pour obtenir des conseils sur la création d'un rôle approprié avec des autorisations de moindre privilège, consultez the section called "Capacité : rôle IAM" et. the section called "Considérations relatives aux fonctionnalités d'EKS"

Choisissez votre outil

Vous pouvez créer une fonctionnalité Argo CD à l'aide de la AWS Management Console AWS CLI ou de eksctl :

- the section called "Console"- Utilisez la console pour une expérience guidée
- the section called " AWS CLI"- Utilisez la AWS CLI pour la création de scripts et l'automatisation
- the section called "eksctl"- Utilisez eksctl pour une expérience native de Kubernetes

Que se passe-t-il lorsque vous créez une fonctionnalité Argo CD

Lorsque vous créez une fonctionnalité Argo CD :

1. EKS crée le service de capacité Argo CD dans le plan de AWS contrôle
2. Des définitions de ressources personnalisées (CRDs) sont installées dans votre cluster
3. Une entrée d'accès est automatiquement créée pour votre rôle de capacité IAM avec des politiques d'entrée d'accès spécifiques aux fonctionnalités qui accordent des autorisations Kubernetes de base (voir) the section called "Considérations relatives aux fonctionnalités d'EKS"
4. Argo CD commence à rechercher ses ressources personnalisées (Applications, ApplicationSets, AppProjects)
5. L'état de capacité passe de `CREATING` à `ACTIVE`

6. L'interface utilisateur d'Argo CD devient accessible via son URL

Une fois actives, vous pouvez créer des applications Argo CD dans votre cluster pour les déployer à partir de vos sources déclaratives.

> **Note**
>
> L'entrée d'accès créée automatiquement n'accorde pas l'autorisation de déployer des applications sur des clusters. Pour déployer des applications, vous devez configurer des autorisations Kubernetes RBAC supplémentaires pour chaque cluster cible. Consultez the section called "Enregistrer des clusters" pour plus de détails sur l'enregistrement des clusters et la configuration de l'accès.

Étapes suivantes

Après avoir créé la fonctionnalité Argo CD :

- the section called "Concepts d'Argo CD"- Découvrez GitOps les principes, les politiques de synchronisation et les modèles multi-clusters
- the section called "Travailler avec Argo CD"- Configurer l'accès au référentiel, enregistrer les clusters cibles et créer des applications
- the section called "Considérations relatives à Argo CD"- Explorez les modèles d'architecture multi-clusters et la configuration avancée

Création d'une fonctionnalité Argo CD à l'aide de la console

Cette rubrique décrit comment créer une fonctionnalité Argo CD à l'aide du AWS Management Console.

Conditions préalables

- AWS Identity Center configuré : Argo CD nécessite AWS Identity Center pour l'authentification. Les utilisateurs locaux ne sont pas pris en charge. Si AWS Identity Center n'est pas configuré, consultez Commencer à utiliser AWS Identity Center pour créer une instance d'Identity Center, et Ajouter des utilisateurs et Ajouter des groupes pour créer des utilisateurs et des groupes pour accéder à Argo CD.

Création de la fonctionnalité Argo CD

1. Ouvrez la console Amazon EKS à l'adresse https://console.aws.amazon.com/eks/ home#/clusters.
2. Sélectionnez le nom de votre cluster pour ouvrir la page détaillée du cluster.
3. Choisissez l'onglet Fonctionnalités.
4. Dans le menu de navigation de gauche, choisissez Argo CD.
5. Choisissez la fonctionnalité Create Argo CD.
6. Pour le rôle de capacité IAM :
   - Si vous possédez déjà un rôle de capacité IAM, sélectionnez-le dans le menu déroulant
   - Si vous devez créer un rôle, choisissez Créer un rôle Argo CD

     Cela ouvre la console IAM dans un nouvel onglet avec une politique de confiance préremplie et un accès complet en lecture à Secrets Manager. Aucune autre autorisation n'est ajoutée par défaut, mais vous pouvez les ajouter si nécessaire. Si vous prévoyez d'utiliser CodeCommit des référentiels ou d'autres AWS services, ajoutez les autorisations appropriées avant de créer le rôle.

     Après avoir créé le rôle, retournez à la console EKS et le rôle sera automatiquement sélectionné.

     > **Note**
     >
     > Si vous envisagez d'utiliser les intégrations facultatives avec AWS Secrets Manager or AWS CodeConnections, vous devrez ajouter des autorisations au rôle. Pour des exemples de politique IAM et des conseils de configuration, reportez-vous aux sections the section called " AWS Secrets Manager" etthe section called " AWS CodeConnections".
7. Configurer AWS l'intégration d'Identity Center :
   a. Sélectionnez Activer AWS l'intégration d'Identity Center.
   b. Choisissez votre instance Identity Center dans le menu déroulant.
   c. Configurez les mappages de rôles pour RBAC en affectant des utilisateurs ou des groupes à des rôles Argo CD (ADMIN, EDITOR ou VIEWER)
8. Choisissez Créer.

Le processus de création de capacités commence.

Vérifiez que la fonctionnalité est active

1. Dans l'onglet Fonctionnalités, consultez l'état des capacités de l'Argo CD.
2. Attendez que le statut passe de `CREATING` à`ACTIVE`.
3. Une fois active, la fonctionnalité est prête à être utilisée.

Pour plus d'informations sur l'état des fonctionnalités et la résolution des problèmes, consultezthe section called "Travailler avec les capacités".

Accédez à l'interface utilisateur d'Argo CD

Une fois la fonctionnalité activée, vous pouvez accéder à l'interface utilisateur d'Argo CD :

1. Sur la page des fonctionnalités d'Argo CD, choisissez Ouvrir l'interface utilisateur d'Argo CD.
2. L'interface utilisateur d'Argo CD s'ouvre dans un nouvel onglet du navigateur.
3. Vous pouvez désormais créer des applications et gérer les déploiements via l'interface utilisateur.

Étapes suivantes

- the section called "Travailler avec Argo CD"- Configurer des référentiels, enregistrer des clusters et créer des applications
- the section called "Considérations relatives à Argo CD"- Architecture multi-clusters et configuration avancée
- the section called "Travailler avec les capacités"- Gérez vos ressources de capacités Argo CD

Création d'une fonctionnalité Argo CD à l'aide de la CLI AWS

Cette rubrique décrit comment créer une fonctionnalité Argo CD à l'aide de la AWS CLI.

Conditions préalables

- AWS CLI : version `2.12.3` ou ultérieure. Pour vérifier votre version, lancez`aws --version`. Pour plus d'informations, consultez la section Installation dans le guide de l'utilisateur de l'interface de ligne de AWS commande.
- **`kubectl`** : outil de ligne de commande pour travailler avec des clusters Kubernetes. Pour de plus amples informations, veuillez consulter the section called "Configurer kubectl et eksctl".

- AWS Identity Center configuré : Argo CD nécessite AWS Identity Center pour l'authentification. Les utilisateurs locaux ne sont pas pris en charge. Si AWS Identity Center n'est pas configuré, consultez Commencer à utiliser AWS Identity Center pour créer une instance d'Identity Center, et Ajouter des utilisateurs et Ajouter des groupes pour créer des utilisateurs et des groupes pour accéder à Argo CD.

Étape 1 : Création d'un rôle de capacité IAM

Créez un fichier de politique de confiance :

```
cat > argocd-trust-policy.json << 'EOF'
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "capabilities.eks.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ]
    }
  ]
}
EOF
```

Créez le rôle IAM :

```
aws iam create-role \
  --role-name ArgoCDCapabilityRole \
  --assume-role-policy-document file://argocd-trust-policy.json
```

> **Note**
>
> Si vous envisagez d'utiliser les intégrations facultatives avec AWS Secrets Manager or AWS CodeConnections, vous devrez ajouter des autorisations au rôle. Pour des exemples de

politique IAM et des conseils de configuration, reportez-vous aux sections the section called " AWS Secrets Manager" etthe section called " AWS CodeConnections".

Étape 2 : Création de la fonctionnalité Argo CD

Créez la ressource de capacité Argo CD sur votre cluster.

Définissez d'abord les variables d'environnement pour la configuration de votre Identity Center :

```
# Get your Identity Center instance ARN (replace region if your IDC instance is in a
 different region)
export IDC_INSTANCE_ARN=$(aws sso-admin list-instances --region [.replaceable]`region`
 --query 'Instances[0].InstanceArn' --output text)

# Get a user ID for RBAC mapping (replace with your username and region if needed)
export IDC_USER_ID=$(aws identitystore list-users \
  --region [.replaceable]`region` \
  --identity-store-id $(aws sso-admin list-instances --region [.replaceable]`region` --
query 'Instances[0].IdentityStoreId' --output text) \
  --query 'Users[?UserName==`your-username`].UserId' --output text)

echo "IDC_INSTANCE_ARN=$IDC_INSTANCE_ARN"
echo "IDC_USER_ID=$IDC_USER_ID"
```

Créez la fonctionnalité grâce à l'intégration d'Identity Center. *region-code*Remplacez-le par la AWS région où se trouve votre cluster, *my-cluster* par le nom de votre cluster et *idc-region-code* par le code de région dans lequel votre centre d'identité IAM a été configuré :

```
aws eks create-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name my-argocd \
  --type ARGOCD \
  --role-arn arn:aws:iam::$(aws sts get-caller-identity --query Account --output
 text):role/ArgoCDCapabilityRole \
  --delete-propagation-policy RETAIN \
  --configuration '{
    "argoCd": {
      "awsIdc": {
        "idcInstanceArn": "'$IDC_INSTANCE_ARN'",
        "idcRegion": "'[.replaceable]`idc-region-code`'"
```

```
      },
      "rbacRoleMappings": [{
        "role": "ADMIN",
        "identities": [{
          "id": "'$IDC_USER_ID'",
          "type": "SSO_USER"
        }]
      }]
    }
  }'
```

La commande revient immédiatement, mais la fonctionnalité met un certain temps à devenir active car EKS crée l'infrastructure et les composants de capacité requis. EKS installera les définitions de ressources personnalisées Kubernetes associées à cette fonctionnalité dans votre cluster lors de sa création.

> **Note**
>
> Si vous recevez un message d'erreur indiquant que le cluster n'existe pas ou que vous n'êtes pas autorisé, vérifiez :
>
> - Le nom du cluster est correct
> - Votre AWS CLI est configurée pour la bonne région
> - Vous disposez des autorisations IAM requises

Étape 3 : vérifier que la fonctionnalité est active

Attendez que la fonctionnalité soit activée. Remplacez *region-code* par la AWS région où se trouve votre cluster et *my-cluster* par le nom de votre cluster.

```
aws eks describe-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name my-argocd \
  --query 'capability.status' \
  --output text
```

La fonctionnalité est prête lorsque l'état s'afficheACTIVE. Ne passez pas à l'étape suivante tant que le statut n'est pas atteintACTIVE.

Vous pouvez également consulter les détails complets des fonctionnalités :

```
aws eks describe-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name my-argocd
```

Étape 4 : vérifier que les ressources personnalisées sont disponibles

Une fois la fonctionnalité activée, vérifiez que les ressources personnalisées Argo CD sont disponibles dans votre cluster :

```
kubectl api-resources | grep argoproj.io
```

Vous devriez voir apparaître une liste `Application` `ApplicationSet` des types de ressources.

Étapes suivantes

- the section called "Travailler avec Argo CD"- Configurer des référentiels, enregistrer des clusters et créer des applications
- the section called "Considérations relatives à Argo CD"- Architecture multi-clusters et configuration avancée
- the section called "Travailler avec les capacités"- Gérez vos ressources de capacités Argo CD

Création d'une fonctionnalité Argo CD à l'aide de eksctl

Cette rubrique décrit comment créer une fonctionnalité Argo CD à l'aide de eksctl.

> **Note**
>
> Les étapes suivantes nécessitent la version eksctl `0.220.0` ou une version ultérieure. Pour vérifier votre version, lancez`eksctl version`.

Étape 1 : Création d'un rôle de capacité IAM

Créez un fichier de politique de confiance :

```
cat > argocd-trust-policy.json << 'EOF'
```

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

Créez le rôle IAM :

```
aws iam create-role \
  --role-name ArgoCDCapabilityRole \
  --assume-role-policy-document file://argocd-trust-policy.json
```

> **Note**
>
> Pour cette configuration de base, aucune politique IAM supplémentaire n'est nécessaire. Si vous prévoyez d'utiliser Secrets Manager pour les informations d'identification du référentiel CodeConnections, vous devez également ajouter des autorisations au rôle. Pour des exemples de politique IAM et des conseils de configuration, reportez-vous aux sections the section called " AWS Secrets Manager" etthe section called " AWS CodeConnections".

Étape 2 : obtenir la configuration de votre AWS Identity Center

Obtenez l'ARN et l'ID utilisateur de votre instance Identity Center pour la configuration du RBAC :

```
# Get your Identity Center instance ARN
aws sso-admin list-instances --query 'Instances[0].InstanceArn' --output text

# Get a user ID for admin access (replace 'your-username' with your Identity Center
 username)
```

```
aws identitystore list-users \
  --identity-store-id $(aws sso-admin list-instances --query
 'Instances[0].IdentityStoreId' --output text) \
  --query 'Users[?UserName==`your-username`].UserId' --output text
```

Notez ces valeurs, vous en aurez besoin à l'étape suivante.

Étape 3 : Création d'un fichier de configuration eksctl

Créez un fichier nommé `argocd-capability.yaml` avec le contenu suivant. Remplacez les valeurs de l'espace réservé par le nom de votre cluster, la région du cluster, l'ARN du rôle IAM, l'ARN de l'instance Identity Center, la région du centre d'identité et l'ID utilisateur :

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-cluster
  region: cluster-region-code

capabilities:
  - name: my-argocd
    type: ARGOCD
    roleArn: arn:aws:iam::[.replaceable]111122223333:role/ArgoCDCapabilityRole
    deletePropagationPolicy: RETAIN
    configuration:
      argocd:
        awsIdc:
          idcInstanceArn: arn:aws:sso:::instance/ssoins-123abc
          idcRegion: idc-region-code
        rbacRoleMappings:
          - role: ADMIN
            identities:
              - id: 38414300-1041-708a-01af-5422d6091e34
                type: SSO_USER
```

> **Note**
>
> Vous pouvez ajouter plusieurs utilisateurs ou groupes aux mappages RBAC. Pour les groupes, utilisez `type: SSO_GROUP` et fournissez l'ID du groupe. Les rôles disponibles sont `ADMIN`, `EDITOR`, et `VIEWER`.

Étape 4 : Création de la fonctionnalité Argo CD

Appliquez le fichier de configuration :

```
eksctl create capability -f argocd-capability.yaml
```

La commande revient immédiatement, mais la fonctionnalité met un certain temps à devenir active.

Étape 5 : vérifier que la fonctionnalité est active

Vérifiez l'état des capacités. Remplacez *region-code* par la AWS région dans laquelle se trouve votre cluster et remplacez *my-cluster* par le nom de votre cluster.

```
eksctl get capability \
  --region region-code \
  --cluster my-cluster \
  --name my-argocd
```

La fonctionnalité est prête lorsque l'état s'afficheACTIVE.

Étape 6 : vérifier que les ressources personnalisées sont disponibles

Une fois la fonctionnalité activée, vérifiez que les ressources personnalisées Argo CD sont disponibles dans votre cluster :

```
kubectl api-resources | grep argoproj.io
```

Vous devriez voir apparaître une liste `Application` `ApplicationSet` des types de ressources.

Étapes suivantes

- the section called “Travailler avec Argo CD”- Apprenez à créer et à gérer des applications Argo CD
- the section called “Considérations relatives à Argo CD”- Configurer l'accès SSO et multi-clusters
- the section called “Travailler avec les capacités”- Gérez vos ressources de capacités Argo CD

Concepts d'Argo CD

Argo CD implémente Git GitOps en considérant Git comme la seule source fiable pour vos déploiements d'applications. Cette rubrique présente un exemple pratique, puis explique les concepts de base que vous devez comprendre lorsque vous travaillez avec l'EKS Capability for Argo CD.

Commencer à utiliser Argo CD

Après avoir créé la fonctionnalité Argo CD (voirthe section called "Création d'une fonctionnalité Argo CD"), vous pouvez commencer à déployer des applications. Cet exemple décrit l'enregistrement d'un cluster et la création d'une application.

Étape 1 : Configurer

Enregistrez votre cluster (obligatoire)

Enregistrez le cluster dans lequel vous souhaitez déployer des applications. Dans cet exemple, nous allons enregistrer le même cluster sur lequel Argo CD est exécuté (vous pouvez utiliser le nom `in-cluster` pour des raisons de compatibilité avec la plupart des exemples d'Argo CD) :

```
# Get your cluster ARN
CLUSTER_ARN=$(aws eks describe-cluster \
  --name my-cluster \
  --query 'cluster.arn' \
  --output text)

# Register the cluster using Argo CD CLI
argocd cluster add $CLUSTER_ARN \
  --aws-cluster-name $CLUSTER_ARN \
  --name in-cluster \
  --project default
```

> **Note**
>
> Pour plus d'informations sur la configuration de l'Argo CD CLI pour qu'elle fonctionne avec la fonctionnalité Argo CD dans EKS, consultez. the section called "Utilisation de l'Argo CD CLI avec la fonctionnalité gérée"

Vous pouvez également enregistrer le cluster à l'aide d'un secret Kubernetes (voir the section called "Enregistrer des clusters" pour plus de détails).

Configurer l'accès au référentiel (facultatif)

Cet exemple utilise un GitHub référentiel public, aucune configuration de référentiel n'est donc requise. Pour les référentiels privés, configurez l'accès à l'aide de AWS Secrets Manager ou de

Kubernetes Secrets (voir the section called "Configuration des référentiels" pour plus de détails). CodeConnections

Pour les AWS services (ECR pour les graphiques Helm CodeConnections, et CodeCommit), vous pouvez les référencer directement dans les ressources de l'application sans créer de référentiel. Le rôle de capacité doit disposer des autorisations IAM requises. Consultez the section called "Configuration des référentiels" pour plus de détails.

Étape 2 : Créer une application

Créez ce manifeste d'application dans `my-app.yaml` :

```
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: guestbook
  namespace: argocd
spec:
  project: default
  source:
    repoURL: https://github.com/argoproj/argocd-example-apps.git
    targetRevision: HEAD
    path: guestbook
  destination:
    name: in-cluster
    namespace: guestbook
  syncPolicy:
    automated:
      prune: true
      selfHeal: true
    syncOptions:
    - CreateNamespace=true
```

Appliquez l'application :

```
kubectl apply -f my-app.yaml
```

Après avoir appliqué cette application, Argo CD : 1. Synchronise l'application de Git avec votre cluster (déploiement initial) 2. Surveille les modifications apportées au dépôt Git 3. Synchronise automatiquement les modifications ultérieures sur votre cluster 4. Détecte et corrige toute dérive par rapport à l'état souhaité 5. Fournit l'état de santé et l'historique de synchronisation dans l'interface utilisateur

Consultez le statut de la demande :

```
kubectl get application guestbook -n argocd
```

Vous pouvez également afficher l'application à l'aide de l'interface de ligne de commande Argo CD ou de l'interface utilisateur du CD Argo (accessible depuis la console EKS sous l'onglet Capabilities de votre cluster).

> **Note**
>
> Lorsque vous utilisez l'Argo CD CLI avec la fonctionnalité gérée, spécifiez les applications avec le préfixe d'espace de noms :. `argocd app get argocd/guestbook`

> **Note**
>
> Utilisez le nom du cluster dans `destination.name` (le nom que vous avez utilisé lors de l'enregistrement du cluster). La fonctionnalité gérée ne prend pas en charge la valeur par défaut locale au sein du cluster (`kubernetes.default.svc`).

Concepts de base

GitOps principes et types de sources

Argo CD implémente GitOps, où la source de votre application est la seule source fiable pour les déploiements :

- Déclaratif - L'état souhaité est déclaré à l'aide de manifestes YAML, de graphiques Helm ou de superpositions Kustomize
- Versionné : chaque modification est suivie grâce à une piste d'audit complète
- Automatisé : Argo CD surveille en permanence les sources et synchronise automatiquement les modifications
- Autoréparation : détecte et corrige le décalage entre l'état souhaité et l'état réel du cluster

Types de sources pris en charge :

- Référentiels Git - GitHub GitLab, Bitbucket, CodeCommit (HTTPS, SSH ou) CodeConnections

- Registres Helm : registres HTTP (similaireshttps://aws.github.io/eks-charts) et registres OCI (similaires) public.ecr.aws
- Images OCI : images de conteneurs contenant des manifestes ou des diagrammes Helm (similairesoci://registry-1.docker.io/user/my-app)

Cette flexibilité permet aux entreprises de choisir les sources qui répondent à leurs exigences de sécurité et de conformité. Par exemple, les organisations qui limitent l'accès à Git depuis les clusters peuvent utiliser l'ECR pour les diagrammes Helm ou les images OCI.

Pour plus d'informations, consultez la section Sources d'applications dans la documentation Argo CD.

Synchronisation et réconciliation

Argo CD surveille en permanence vos sources et vos clusters afin de détecter et de corriger les différences :

1. Interroge les sources pour connaître les modifications (par défaut : toutes les 6 minutes)
2. Compare l'état souhaité avec l'état du cluster
3. Marque les applications comme Synced ou OutOfSync
4. Synchronise les modifications automatiquement (si elles sont configurées) ou attend une approbation manuelle
5. Surveille l'état des ressources après la synchronisation

Les ondes de synchronisation contrôlent l'ordre de création des ressources à l'aide d'annotations :

```
metadata:
  annotations:
    argocd.argoproj.io/sync-wave: "0"  # Default if not specified
```

Les ressources sont appliquées par ordre de vague (les nombres les plus bas d'abord, y compris les nombres négatifs comme-1). Wave 0 est la valeur par défaut si elle n'est pas spécifiée. Cela vous permet de créer des dépendances telles que des espaces de noms (wave-1) avant les déploiements (wave0) avant les services (wave1).

L'autoréparation annule automatiquement les modifications manuelles :

```
spec:
  syncPolicy:
```

```
    automated:
      selfHeal: true
```

> **Note**
>
> La fonctionnalité gérée utilise le suivi des ressources basé sur les annotations (et non sur les étiquettes) pour une meilleure compatibilité avec les conventions Kubernetes et les autres outils.

Pour des informations détaillées sur les phases de synchronisation, les hooks et les modèles avancés, consultez la documentation de synchronisation Argo CD.

État de santé des applications

Argo CD surveille l'état de toutes les ressources de votre application :

État de santé :* Sain - Toutes les ressources fonctionnent comme prévu * Progression - Ressources en cours de création ou de mise à jour * Dégradée - Certaines ressources ne sont pas saines (pods tombent en panne, tâches échouent) * Suspendue - Application interrompue intentionnellement * Manquante - Les ressources définies dans Git ne sont pas présentes dans le cluster

Argo CD intègre des contrôles de santé pour les ressources Kubernetes courantes (déploiements, tâches StatefulSets, etc.) et prend en charge les contrôles de santé personnalisés pour. CRDs

L'état de santé de l'application est déterminé par toutes ses ressources. Si une ressource l'est`Degraded`, c'est bien l'application qui l'est`Degraded`.

Pour plus d'informations, consultez Resource Health dans la documentation d'Argo CD.

Modèles multi-clusters

Argo CD prend en charge deux modèles de déploiement principaux :

H ub-and-spoke - Exécutez Argo CD sur un cluster de gestion dédié qui se déploie sur plusieurs clusters de charge de travail :* Contrôle et visibilité centralisés * Politiques cohérentes sur tous les clusters * Une instance Argo CD à gérer * Séparation claire entre le plan de contrôle et les charges de travail

Par cluster : exécutez Argo CD sur chaque cluster, en gérant uniquement les applications de ce cluster :* Séparation des clusters (une défaillance n'affecte pas les autres) * Simplification du réseau

(aucune communication entre clusters) * Configuration initiale plus facile (pas d'enregistrement du cluster)

Choisissez hub-and-spoke pour les équipes de plateforme qui gèrent de nombreux clusters, ou par cluster pour les équipes indépendantes ou lorsque les clusters doivent être complètement isolés.

Pour une configuration multicluster détaillée, voirthe section called “Considérations relatives à Argo CD”.

Projets

Les projets fournissent un regroupement logique et un contrôle d'accès pour les applications :

- Restrictions relatives aux sources - Limitez les référentiels Git pouvant être utilisés
- Restrictions de destination : limitez les clusters et les espaces de noms pouvant être ciblés
- Restrictions de ressources : limitez les types de ressources Kubernetes pouvant être déployés
- Intégration RBAC : associez les projets à l'utilisateur et au groupe AWS Identity Center IDs

Les candidatures appartiennent à un seul projet. S'il n'est pas spécifié, ils utilisent le `default` projet, qui n'est soumis à aucune restriction par défaut. Pour une utilisation en production, modifiez le `default` projet pour en restreindre l'accès et créez de nouveaux projets avec les restrictions appropriées.

Pour la configuration du projet et les modèles RBAC, voirthe section called “Configurer les autorisations ”.

Options de synchronisation

Ajustez le comportement de synchronisation à l'aide des options courantes :

- `CreateNamespace=true`- Création automatique d'un espace de noms de destination
- `ServerSideApply=true`- Utilisez l'application côté serveur pour une meilleure résolution des conflits
- `SkipDryRunOnMissingResource=true`- Ignorez l'exécution à sec lorsque cela CRDs n'existe pas encore (utile pour les instances Kro)

```
spec:
  syncPolicy:
```

```
    syncOptions:
    - CreateNamespace=true
    - ServerSideApply=true
    - SkipDryRunOnMissingResource=true
```

Pour une liste complète des options de synchronisation, consultez la documentation des options de synchronisation Argo CD.

Étapes suivantes

- the section called "Configuration des référentiels"- Configurer l'accès au dépôt Git
- the section called "Enregistrer des clusters"- Enregistrez les clusters cibles pour le déploiement
- the section called "Création d'applications"- Créez votre première application
- the section called "Considérations relatives à Argo CD"- Modèles spécifiques à EKS, intégration d'Identity Center et configuration multi-clusters
- Documentation Argo CD - Documentation complète sur Argo CD, y compris les crochets de synchronisation, les contrôles de santé et les modèles avancés

Configurer les autorisations d'Argo CD

La fonctionnalité gérée par Argo CD s'intègre à AWS Identity Center pour l'authentification et utilise des rôles RBAC intégrés pour l'autorisation. Cette rubrique explique comment configurer les autorisations pour les utilisateurs et les équipes.

Comment fonctionnent les autorisations avec Argo CD

La fonctionnalité Argo CD utilise AWS Identity Center pour l'authentification et fournit trois rôles RBAC intégrés pour l'autorisation.

Lorsqu'un utilisateur accède à Argo CD :

1. Ils s'authentifient à l'aide AWS d'Identity Center (qui peut être fédéré auprès de votre fournisseur d'identité d'entreprise)
2. AWS Identity Center fournit des informations sur les utilisateurs et les groupes à Argo CD
3. Argo CD associe les utilisateurs et les groupes aux rôles RBAC en fonction de votre configuration
4. Les utilisateurs ne voient que les applications et les ressources auxquelles ils sont autorisés à accéder

Rôles RBAC intégrés

La fonctionnalité Argo CD fournit trois rôles intégrés que vous pouvez associer aux utilisateurs et aux groupes AWS d'Identity Center. Il s'agit de rôles de portée mondiale qui contrôlent l'accès aux ressources Argo CD telles que les projets, les clusters et les référentiels.

> **Important**
>
> Les rôles globaux contrôlent l'accès à Argo CD lui-même, et non aux ressources liées au projet, telles que les applications. Les utilisateurs de EDITOR et VIEWER ne peuvent pas voir ou gérer les applications par défaut. Ils ont besoin de rôles de projet pour accéder aux ressources spécifiques au projet. Consultez the section called "Rôles du projet et accès délimité par le projet" pour plus de détails sur l'octroi de l'accès aux applications et à d'autres ressources liées au projet.

ADMINISTRATEUR

Accès complet à toutes les ressources et à tous les paramètres d'Argo CD :

- Créez, mettez à jour et supprimez des applications et ApplicationSets dans n'importe quel projet
- Gérer la configuration d'Argo CD
- Enregistrez et gérez les clusters cibles de déploiement
- Configuration de l'accès au référentiel
- Création et gestion de projets
- Afficher l'état et l'historique de toutes les demandes
- Répertorier et accéder à tous les clusters et référentiels

RÉDACTEUR

Peut mettre à jour des projets et configurer les rôles des projets, mais ne peut pas modifier les paramètres globaux d'Argo CD :

- Mettre à jour les projets existants (impossible de créer ou de supprimer des projets)
- Configuration des rôles et des autorisations du projet
- Afficher les clés et certificats GPG

- Impossible de modifier la configuration globale d'Argo CD
- Impossible de gérer directement les clusters ou les référentiels
- Impossible de voir ou de gérer les applications sans rôles dans le projet

SPECTATEUR

Accès en lecture seule aux ressources Argo CD :

- Afficher les configurations de projet
- Répertorier tous les projets (y compris les projets auxquels l'utilisateur n'est pas affecté)
- Afficher les clés et certificats GPG
- Impossible de répertorier les clusters ou les référentiels
- Impossible d'apporter des modifications
- Impossible de voir ou de gérer les applications sans rôles dans le projet

> **Note**
>
> Pour accorder aux utilisateurs EDITOR ou VIEWER l'accès aux applications, un ADMINISTRATEUR ou un ÉDITEUR doit créer des rôles de projet qui associent les groupes Identity Center à des autorisations spécifiques au sein d'un projet.

Rôles du projet et accès délimité par le projet

Les rôles globaux (ADMIN, EDITOR, VIEWER) contrôlent l'accès à Argo CD lui-même. Les rôles de projet contrôlent l'accès aux ressources et aux capacités au sein d'un projet spécifique, notamment :

- Ressources : applications, informations d'identification du référentiel ApplicationSets, informations d'identification du cluster
- Fonctionnalités : accès aux journaux, accès exec aux modules de l'application

Comprendre le modèle d'autorisation à deux niveaux :

- Portée globale : les rôles intégrés déterminent ce que les utilisateurs peuvent faire avec les projets, les clusters, les référentiels et les paramètres d'Argo CD

- Portée du projet : les rôles du projet déterminent ce que les utilisateurs peuvent faire avec les ressources et les capacités d'un projet spécifique

Autrement dit :

- Les utilisateurs ADMIN peuvent accéder à toutes les ressources et fonctionnalités du projet sans configuration supplémentaire
- Les utilisateurs EDITOR et VIEWER doivent se voir attribuer des rôles de projet pour accéder aux ressources et aux capacités du projet
- Les utilisateurs de l'ÉDITEUR peuvent créer des rôles de projet pour s'accorder, ainsi qu'à d'autres, un accès au sein de projets qu'ils peuvent mettre à jour

Exemple de flux de travail :

1. Un ADMINISTRATEUR associe un groupe Identity Center au rôle EDITOR à l'échelle mondiale
2. Un ADMIN crée un projet pour une équipe
3. L'ÉDITEUR configure les rôles du projet au sein de ce projet pour permettre aux membres de l'équipe d'accéder aux ressources définies dans le cadre du projet.
4. Les membres de l'équipe (qui peuvent avoir un rôle global VIEWER) peuvent désormais voir et gérer les applications de ce projet en fonction de leurs autorisations de rôle dans le projet

Pour plus de détails sur la configuration des rôles du projet, consultezthe section called “Contrôle d'accès basé sur le projet”.

Configuration des mappages de rôles

Associez les utilisateurs et les groupes AWS d'Identity Center aux rôles Argo CD lors de la création ou de la mise à jour de la fonctionnalité.

Exemple de mappage des rôles :

```
{
  "rbacRoleMapping": {
    "ADMIN": ["AdminGroup", "alice@example.com"],
    "EDITOR": ["DeveloperGroup", "DevOpsTeam"],
    "VIEWER": ["ReadOnlyGroup", "bob@example.com"]
  }
```

```
}
```

> **Note**
>
> Les noms de rôles distinguent les majuscules des minuscules et doivent être en majuscules (ADMIN, EDITOR, VIEWER).

> **Important**
>
> L'intégration d'EKS Capabilities à AWS Identity Center prend en charge jusqu'à 1 000 identités par fonctionnalité Argo CD. Une identité peut être un utilisateur ou un groupe.

Mettre à jour les mappages de rôles :

```
aws eks update-capability \
  --region us-east-1 \
  --cluster-name cluster \
  --capability-name capname \
  --endpoint "https://eks.ap-northeast-2.amazonaws.com" \
  --role-arn "arn:aws:iam::[.replaceable]111122223333:role/
[.replaceable]`EKSCapabilityRole`" \
  --configuration '{
    "argoCd": {
      "rbacRoleMappings": {
        "addOrUpdateRoleMappings": [
          {
            "role": "ADMIN",
            "identities": [
              { "id": "686103e0-f051-7068-b225-e6392b959d9e", "type": "SSO_USER" }
            ]
          }
        ]
      }
    }
  }'
```

Utilisation du compte administrateur

Le compte administrateur est conçu pour la configuration initiale et les tâches administratives telles que l'enregistrement de clusters et la configuration de référentiels.

Lorsque le compte administrateur est approprié :

- Configuration et configuration initiales des fonctionnalités
- Développement en solo ou démonstrations rapides
- Tâches administratives (enregistrement du cluster, configuration du référentiel, création de projets)

Bonnes pratiques pour le compte administrateur :

- Ne confiez pas les jetons de compte au contrôle de version
- Faites pivoter les jetons immédiatement s'ils sont exposés
- Limitez l'utilisation des jetons de compte aux tâches de configuration et d'administration
- Définissez des délais d'expiration courts (maximum 12 heures)
- Seuls 5 jetons de compte peuvent être créés à un moment donné

Quand utiliser plutôt l'accès basé sur le projet :

- Environnements de développement partagés avec plusieurs utilisateurs
- Tout environnement similaire à celui de la production
- Lorsque vous avez besoin de pistes d'audit pour savoir qui a effectué des actions
- Lorsque vous devez appliquer des restrictions de ressources ou des limites d'accès

Pour les environnements de production et les scénarios multi-utilisateurs, utilisez le contrôle d'accès basé sur le projet avec des rôles RBAC dédiés mappés aux groupes Identity Center. AWS

Contrôle d'accès basé sur le projet

Utilisez Argo CD Projects (AppProject) pour fournir un contrôle d'accès précis et une isolation des ressources aux équipes.

> **Important**
>
> Avant d'affecter des utilisateurs ou des groupes à des rôles spécifiques au projet, vous devez d'abord les associer à un rôle Argo CD global (ADMIN, EDITOR ou VIEWER) dans la configuration des fonctionnalités. Les utilisateurs ne peuvent pas accéder à Argo CD sans mappage global des rôles, même s'ils sont affectés à des rôles de projet.
> Envisagez de mapper les utilisateurs au rôle VIEWER de manière globale, puis d'accorder des autorisations supplémentaires via des rôles spécifiques au projet. Cela fournit un accès de base tout en permettant un contrôle précis au niveau du projet.

Les projets fournissent :

- Restrictions relatives aux sources : limite les référentiels Git pouvant être utilisés
- Restrictions de destination : limitez les clusters et les espaces de noms pouvant être ciblés
- Restrictions de ressources : limitez les types de ressources Kubernetes pouvant être déployés
- Intégration RBAC : associez les projets à des groupes AWS Identity Center ou à des rôles Argo CD

Exemple de projet pour isoler une équipe :

```
apiVersion: argoproj.io/v1alpha1
kind: AppProject
metadata:
  name: team-a
  namespace: argocd
spec:
  description: Team A applications

  # Required: Specify which namespaces this project watches for Applications
  sourceNamespaces:
  - argocd

  # Source restrictions
  sourceRepos:
  - https://github.com/myorg/team-a-apps

  # Destination restrictions
  destinations:
  - namespace: team-a-*
```

```
    server: arn:aws:eks:us-west-2:111122223333:cluster/production

  # Resource restrictions
  clusterResourceWhitelist:
  - group: ''
    kind: Namespace
  namespaceResourceWhitelist:
  - group: 'apps'
    kind: Deployment
  - group: ''
    kind: Service
  - group: ''
    kind: ConfigMap
```

Espaces de noms sources

Lorsque vous utilisez la fonctionnalité EKS Argo CD, le `spec.sourceNamespaces` champ est obligatoire dans AppProject les définitions. Ce champ indique quel espace de noms peut contenir des applications ou ApplicationSets qui font référence à ce projet.

> **Important**
>
> La fonctionnalité EKS Argo CD ne prend en charge qu'un seul espace de noms pour les applications, à savoir l'espace de ApplicationSets noms que vous avez spécifié lors de la création de la fonctionnalité (généralement). `argocd` Cela diffère de l'Argo CD open source qui prend en charge plusieurs espaces de noms.

AppProject configuration

Tous AppProjects doivent inclure l'espace de noms configuré pour la fonctionnalité dans `sourceNamespaces` :

```
apiVersion: argoproj.io/v1alpha1
kind: AppProject
metadata:
  name: team-a-project
  namespace: argocd
spec:
  description: Applications for Team A
```

```
  # Required: Specify the capability's configured namespace
 (configuration.argoCd.namespace)
  sourceNamespaces:
    - argocd  # Must match your capability's namespace configuration

  # Source repositories this project can deploy from
  sourceRepos:
    - 'https://github.com/my-org/team-a-*'

  # Destination restrictions
  destinations:
    - namespace: 'team-a-*'
      server: arn:aws:eks:us-west-2:111122223333:cluster/my-cluster
```

> **Note**
>
> Si vous omettez l'espace de noms de la fonctionnalité dans`sourceNamespaces`, les applications ou ApplicationSets dans cet espace de noms ne peuvent pas faire référence à ce projet, ce qui entraîne des échecs de déploiement.

Affecter des utilisateurs à des projets :

Les rôles de projet permettent aux utilisateurs de EDITOR et VIEWER d'accéder aux ressources du projet (applications ApplicationSets, informations d'identification du référentiel et du cluster) et aux fonctionnalités (journaux, exec) du projet. Sans rôles de projet, ces utilisateurs ne peuvent pas accéder à ces ressources même s'ils disposent d'un accès global aux rôles.

Les utilisateurs ADMIN ont accès à toutes les applications sans avoir besoin de rôles de projet.

Exemple : accorder l'accès à l'application aux membres de l'équipe

```
apiVersion: argoproj.io/v1alpha1
kind: AppProject
metadata:
  name: team-a
  namespace: argocd
spec:
  # ... project configuration ...

  sourceNamespaces:
  - argocd
```

```
  # Project roles grant Application-level access
  roles:
  - name: developer
    description: Team A developers - can manage Applications
    policies:
    - p, proj:team-a:developer, applications, *, team-a/*, allow
    - p, proj:team-a:developer, clusters, get, *, allow  # See cluster names in UI
    groups:
    - 686103e0-f051-7068-b225-e6392b959d9e  # Identity Center group ID

  - name: viewer
    description: Team A viewers - read-only Application access
    policies:
    - p, proj:team-a:viewer, applications, get, team-a/*, allow
    - p, proj:team-a:viewer, clusters, get, *, allow  # See cluster names in UI
    groups:
    - 786203e0-f051-7068-b225-e6392b959d9f  # Identity Center group ID
```

> **Note**
>
> Incluez `clusters, get, *, allow` les rôles dans le projet pour permettre aux utilisateurs de voir les noms des clusters dans l'interface utilisateur. Sans cette autorisation, le cluster de destination s'affiche comme « inconnu ».

Comprendre les politiques relatives aux rôles du projet :

Le format de la politique est le suivant : `p, proj:<project>:<role>, <resource>, <action>, <object>, <allow/deny>`

Politiques relatives aux ressources :

- `applications, , team-a/, allow`- Accès complet à toutes les applications de l'équipe-un projet
- `applications, get, team-a/*, allow`- Accès en lecture seule aux applications
- `applications, sync, team-a/*, allow`- Peut synchroniser les applications mais pas les créer/supprimer
- `applications, delete, team-a/*, allow`- Peut supprimer des applications (à utiliser avec prudence)

- `applicationsets, , team-a/, allow`- Accès complet à ApplicationSets
- `repositories, *, *, allow`- Accès aux informations d'identification du référentiel
- `clusters, *, *, allow`- Accès aux informations d'identification du cluster

Politiques de capacité :

- `logs, , team-a/, allow`- Accès aux journaux des applications
- `exec, , team-a/, allow`- Accès exécutif aux modules d'applications

> **Note**
>
> Les utilisateurs de l'ÉDITEUR peuvent créer des rôles de projet pour s'octroyer, ainsi qu'à d'autres, des autorisations au sein de projets qu'ils peuvent mettre à jour. Cela permet aux chefs d'équipe de contrôler l'accès aux ressources définies dans le cadre du projet pour leur équipe sans nécessiter l'intervention de l'ADMINISTRATEUR.

> **Note**
>
> Utilisez le groupe Identity Center IDs (et non les noms de groupe) dans le `groups` champ. Vous pouvez également utiliser Identity Center User IDs pour un accès utilisateur individuel. Retrouvez-les IDs dans la console AWS Identity Center ou à l'aide de la AWS CLI.

Modèles d'autorisation courants

Modèle 1 : équipe d'administration avec accès complet

```
{
  "rbacRoleMapping": {
    "ADMIN": ["PlatformTeam", "SRETeam"]
  }
}
```

Les utilisateurs ADMIN peuvent voir et gérer toutes les ressources liées au projet sans configuration supplémentaire.

Schéma 2 : les chefs d'équipe gèrent les projets, les développeurs y accèdent via les rôles du projet

```
{
  "rbacRoleMapping": {
    "ADMIN": ["PlatformTeam"],
    "EDITOR": ["TeamLeads"],
    "VIEWER": ["AllDevelopers"]
  }
}
```

1. ADMIN crée des projets pour chaque équipe
2. Les chefs d'équipe (EDITOR) configurent les rôles du projet pour permettre à leurs développeurs d'accéder aux ressources du projet (applications ApplicationSets, informations d'identification) et aux fonctionnalités (journaux, exécutifs)
3. Les développeurs (VIEWER) ne peuvent accéder qu'aux ressources et fonctionnalités autorisées par leurs rôles dans le projet

Modèle 3 : accès basé sur l'équipe avec des rôles de projet

1. L'ADMINISTRATEUR crée des projets et cartographie l'équipe mène au rôle d'ÉDITEUR à l'échelle mondiale
2. Les chefs d'équipe (ÉDITEUR) attribuent aux membres de l'équipe des rôles de projet au sein de leurs projets
3. Les membres de l'équipe n'ont besoin que du rôle global VIEWER : les rôles de projet donnent accès aux ressources et aux capacités du projet

```
{
  "rbacRoleMapping": {
    "ADMIN": ["PlatformTeam"],
    "EDITOR": ["TeamLeads"],
    "VIEWER": ["AllDevelopers"]
  }
}
```

Bonnes pratiques

Utilisez des groupes plutôt que des utilisateurs individuels : associez les groupes AWS Identity Center aux rôles Argo CD plutôt qu'aux utilisateurs individuels pour faciliter la gestion.

Commencez avec le moins de privilèges : commencez par l'accès VIEWER et accordez EDITOR ou ADMIN selon les besoins.

Utilisez des projets pour isoler les équipes : créez des projets distincts AppProjects pour les différentes équipes ou environnements afin de renforcer les limites.

Tirez parti de la fédération Identity Center : configurez AWS Identity Center pour qu'il soit fédéré avec votre fournisseur d'identité d'entreprise afin de centraliser la gestion des utilisateurs.

Révisions d'accès régulières : passez régulièrement en revue les mappages de rôles et les attributions de projets pour garantir des niveaux d'accès appropriés.

Limiter l'accès au cluster : n'oubliez pas qu'Argo CD RBAC contrôle l'accès aux ressources et aux opérations Argo CD, mais ne correspond pas au RBAC Kubernetes. Les utilisateurs disposant d'un accès à Argo CD peuvent déployer des applications sur des clusters auxquels Argo CD a accès. Limitez les clusters auxquels Argo CD peut accéder et utilisez les restrictions de destination du projet pour contrôler où les applications peuvent être déployées.

AWS autorisations de service

Pour utiliser les AWS services directement dans les ressources de l'application (sans créer de ressources de référentiel), associez les autorisations IAM requises au rôle de capacité.

ECR pour les cartes Helm :

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ecr:GetAuthorizationToken",
        "ecr:BatchCheckLayerAvailability",
        "ecr:GetDownloadUrlForLayer",
        "ecr:BatchGetImage"
      ],
```

```
      "Resource": "*"
    }
  ]
}
```

CodeCommit référentiels :

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codecommit:GitPull"
      ],
      "Resource": "arn:aws:codecommit:region:account-id:repository-name"
    }
  ]
}
```

CodeConnections (GitHub GitLab, Bitbucket) :

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codeconnections:UseConnection"
      ],
      "Resource": "arn:aws:codeconnections:region:account-id:connection/connection-id"
    }
  ]
}
```

Consultez the section called "Configuration des référentiels" pour plus de détails sur l'utilisation de ces intégrations.

Étapes suivantes

- the section called "Travailler avec Argo CD"- Apprenez à créer des applications et à gérer les déploiements

- the section called "Concepts d'Argo CD"- Comprendre les concepts d'Argo CD, y compris les projets
- the section called "Considérations relatives aux fonctionnalités d'EKS"- Passez en revue les meilleures pratiques de sécurité en termes de fonctionnalités

Travailler avec Argo CD

Avec Argo CD, vous définissez les applications dans les référentiels Git et Argo CD les synchronise automatiquement avec vos clusters Kubernetes. Cela permet un déploiement d'applications déclaratif et contrôlé par version avec détection automatique des dérives.

Conditions préalables

Avant de travailler avec Argo CD, vous devez :

- Création d'un cluster EKS doté de la fonctionnalité Argo CD (voirthe section called "Création d'une fonctionnalité Argo CD")
- Un dépôt Git contenant des manifestes Kubernetes
- `kubectl`configuré pour communiquer avec votre cluster

Tâches courantes

Les rubriques suivantes vous guident dans les tâches courantes d'Argo CD :

the section called "Configuration des référentiels"- Configurez Argo CD pour accéder à vos référentiels Git à l'aide de AWS Secrets Manager ou de AWS CodeConnections Kubernetes Secrets.

the section called "Enregistrer des clusters"- Enregistrez les clusters cibles sur lesquels Argo CD déploiera des applications.

the section called "Projets"- Organisez les applications et renforcez les limites de sécurité à l'aide de projets pour les environnements à locataires multiples.

the section called "Création d'applications"- Créez des applications déployées à partir de référentiels Git avec des politiques de synchronisation automatisées ou manuelles.

the section called "ApplicationSets"- ApplicationSets À utiliser pour déployer des applications dans plusieurs environnements ou clusters à l'aide de modèles et de générateurs.

Accédez à l'interface utilisateur d'Argo CD

Accédez à l'interface utilisateur du CD Argo via la console EKS :

1. Ouvrez la console Amazon EKS
2. Sélectionnez votre cluster
3. Choisissez l'onglet Fonctionnalités
4. Choisissez Argo CD
5. Choisissez Open Argo CD UI

L'interface utilisateur fournit une topologie visuelle des applications, l'état et l'historique de synchronisation, l'état des ressources et les événements, les contrôles de synchronisation manuels et la gestion des applications.

Documentation en amont

Pour des informations détaillées sur les fonctionnalités d'Argo CD :

- Documentation Argo CD - Guide de l'utilisateur complet
- Spécifications de l'application - Référence complète de l'API de l'application
- ApplicationSet Guide - ApplicationSet modèles et exemples
- Argo CD GitHub - Code source et exemples

Configuration de l'accès au référentiel

Avant de déployer des applications, configurez Argo CD pour accéder à vos référentiels Git et à vos registres de diagrammes Helm. Argo CD prend en charge plusieurs méthodes d'authentification pour GitHub GitLab, Bitbucket et AWS CodeCommit AWS ECR.

> **Note**
>
> Pour les intégrations de AWS services directes (graphiques ECR Helm, CodeCommit référentiels, etc. CodeConnections), vous pouvez les référencer directement dans les ressources de l'application sans créer de configurations de référentiel. Le rôle de capacité doit disposer des autorisations IAM requises. Consultez the section called “Configurer les autorisations ” pour plus de détails.

Conditions préalables

- Création d'un cluster EKS doté de la fonctionnalité Argo CD
- Référentiels Git contenant des manifestes Kubernetes
- `kubectl`configuré pour communiquer avec votre cluster

> **Note**
>
> AWS CodeConnections peut se connecter à des serveurs Git situés dans AWS le cloud ou sur site. Pour de plus amples informations, veuillez consulter AWS CodeConnections.

Méthodes d'authentification

Method	Cas d'utilisation	Autorisations IAM nécessaires
Intégration directe aux AWS services		
CodeCommit	Intégration directe avec les référentiels AWS CodeCommit Git. Aucune configuration du référentiel n'est nécessaire.	`codecommit:GitPull`
CodeConnections	Connectez-vous à GitHub Bitbucket ou connectez-vous à Bitbucket grâce à l'authentification gérée. GitLab Nécessite la configuration de la connexion.	`codeconnections:UseConnection`
Artefacts OCI ECR	Intégration directe avec AWS ECR pour les graphiques OCI Helm et les images de manifeste. Aucune configuration du référentiel n'est nécessaire.	`arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryPullOnly`
Configuration du référentiel avec informations d'identification		

Method	Cas d'utilisation	Autorisations IAM nécessaires
AWS Secrets Manager (nom d'utilisateur/jeton)	Stockez des jetons d'accès ou des mots de passe personnels. Permet la rotation des informations d'identification sans accès à Kubernetes.	`arn:aws:iam::aws:policy/AWSSecretsManagerClientReadOnlyAccess`
AWS Gestionnaire de Secrets (clé SSH)	Utilisez l'authentification par clé SSH. Permet la rotation des informations d'identification sans accès à Kubernetes.	`arn:aws:iam::aws:policy/AWSSecretsManagerClientReadOnlyAccess`
AWS Secrets Manager (GitHub App)	GitHub Authentification de l'application avec clé privée. Permet la rotation des informations d'identification sans accès à Kubernetes.	`arn:aws:iam::aws:policy/AWSSecretsManagerClientReadOnlyAccess`
Le secret de Kubernetes	Méthode Argo CD standard utilisant des secrets intégrés au cluster	Aucune (autorisations gérées par EKS Access Entry avec Kubernetes RBAC)

Accès direct aux AWS services

Pour les AWS services, vous pouvez les référencer directement dans les ressources de l'application sans créer de configurations de référentiel. Le rôle de capacité doit disposer des autorisations IAM requises.

CodeCommit référentiels

CodeCommit Référencez les référentiels directement dans les applications :

```
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: my-app
  namespace: argocd
spec:
  source:
    repoURL: https://git-codecommit.region.amazonaws.com/v1/repos/repository-name
    targetRevision: main
```

```
    path: kubernetes/manifests
```

Autorisations de rôle de capacité requises :

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "codecommit:GitPull",
      "Resource": "arn:aws:codecommit:region:account-id:repository-name"
    }
  ]
}
```

CodeConnections

Reference GitHub GitLab, ou référentiels Bitbucket via. CodeConnections Le format de l'URL du référentiel est dérivé de l'ARN de CodeConnections connexion.

Le format de l'URL du référentiel est le suivant :

```
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: my-app
  namespace: argocd
spec:
  source:
    repoURL: https://codeconnections.region.amazonaws.com/git-http/account-
id/region/connection-id/owner/repository.git
    targetRevision: main
    path: kubernetes/manifests
```

Autorisations de rôle de capacité requises :

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

```
      "Action": "codeconnections:UseConnection",
      "Resource": "arn:aws:codeconnections:region:account-id:connection/connection-id"
    }
  ]
}
```

Graphiques ECR Helm

L'ECR stocke les diagrammes Helm sous forme d'artefacts OCI. Argo CD permet de les référencer de deux manières :

Format Helm (recommandé pour les diagrammes Helm) :

```
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: my-app-helm
  namespace: argocd
spec:
  source:
    repoURL: account-id.dkr.ecr.region.amazonaws.com/repository-name
    targetRevision: chart-version
    chart: chart-name
    helm:
      valueFiles:
        - values.yaml
```

Remarque : N'incluez pas le `oci://` préfixe lorsque vous utilisez le format Helm. Utilisez le `chart` champ pour spécifier le nom du graphique.

Format OCI (pour les artefacts OCI avec des manifestes Kubernetes) :

```
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: my-app-oci
  namespace: argocd
spec:
  source:
    repoURL: oci://account-id.dkr.ecr.region.amazonaws.com/repository-name
    targetRevision: artifact-version
    path: path-to-manifests
```

Remarque : Incluez le `oci://` préfixe lorsque vous utilisez le format OCI. Utilisez le `path` champ au lieu de`chart`.

Autorisations de rôle de capacité requises : attachez la politique gérée :

```
arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryPullOnly
```

Cette politique inclut les autorisations ECR nécessaires : `ecr:GetAuthorizationToken`, `ecr:BatchGetImage`, et`ecr:GetDownloadUrlForLayer`.

Utilisation de AWS Secrets Manager

Stockez les informations d'identification du référentiel dans Secrets Manager et référencez-les dans les configurations du référentiel Argo CD. L'utilisation de Secrets Manager permet une rotation automatique des informations d'identification sans nécessiter l'accès à Kubernetes RBAC. Les informations d'identification peuvent être modifiées à l'aide des autorisations IAM vers Secrets Manager, et Argo CD lit automatiquement les valeurs mises à jour.

> **Note**
>
> Pour la réutilisation des informations d'identification dans plusieurs référentiels (par exemple, tous les référentiels d'une GitHub organisation), utilisez des modèles d'informations d'identification de référentiel avec. `argocd.argoproj.io/secret-type: repo-creds` Cela fournit une meilleure expérience utilisateur que la création de secrets de dépôt individuels. Pour plus d'informations, consultez la section Informations d'identification du référentiel dans la documentation Argo CD.

Authentification par nom d'utilisateur et token

Pour les référentiels HTTPS dotés de jetons d'accès ou de mots de passe personnels :

Créez le secret dans Secrets Manager :

```
aws secretsmanager create-secret \
  --name argocd/my-repo \
  --description "GitHub credentials for Argo CD" \
  --secret-string '{"username":"your-username","token":"your-personal-access-token"}'
```

Champs de certificat client TLS facultatifs (pour les serveurs Git privés) :

```
aws secretsmanager create-secret \
  --name argocd/my-private-repo \
  --secret-string '{
    "username":"your-username",
    "token":"your-token",

 "tlsClientCertData":"LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCi4uLgotLS0tLUVORCBDRVJUSUZJQ0FURS0tL

 "tlsClientCertKey":"LS0tLS1CRUdJTiBQUklWQVRFIEtFWS0tLS0tCi4uLgotLS0tLUVORCBQUklWQVRFIEtFWS0tLS
  }'
```

> **Note**
>
> Les `tlsClientCertKey` valeurs `tlsClientCertData` et doivent être codées en base64.

Créez un secret de dépôt faisant référence à Secrets Manager :

```
apiVersion: v1
kind: Secret
metadata:
  name: my-repo
  namespace: argocd
  labels:
    argocd.argoproj.io/secret-type: repository
stringData:
  type: git
  url: https://github.com/your-org/your-repo
  secretArn: arn:aws:secretsmanager:us-west-2:111122223333:secret:argocd/my-repo-AbCdEf
  project: default
```

Authentification par clé SSH

Pour un accès Git basé sur SSH, stockez la clé privée en texte brut (et non en JSON) :

Créez le secret avec la clé privée SSH :

```
aws secretsmanager create-secret \
  --name argocd/my-repo-ssh \
```

```
  --description "SSH key for Argo CD" \
  --secret-string "-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAABlwAAAAdzc2gtcn
...
-----END OPENSSH PRIVATE KEY-----"
```

Créez un secret de dépôt pour SSH :

```
apiVersion: v1
kind: Secret
metadata:
  name: my-repo-ssh
  namespace: argocd
  labels:
    argocd.argoproj.io/secret-type: repository
stringData:
  type: git
  url: git@github.com:your-org/your-repo.git
  secretArn: arn:aws:secretsmanager:us-west-2:111122223333:secret:argocd/my-repo-ssh-
AbCdEf
  project: default
```

GitHub Authentification par application

Pour GitHub l'authentification des applications à l'aide d'une clé privée :

Créez le secret à l'aide des informations d'identification de GitHub l'application :

```
aws secretsmanager create-secret \
  --name argocd/github-app \
  --description "GitHub App credentials for Argo CD" \
  --secret-string '{

 "githubAppPrivateKeySecret":"LS0tLS1CRUdJTiBSU0EgUFJJVkFURSBLRVktLS0tLQouLi4KLS0tLS1FTkQgUlNBI
    "githubAppID":"123456",
    "githubAppInstallationID":"12345678"
  }'
```

> **Note**
>
> La `githubAppPrivateKeySecret` valeur doit être codée en base64.

Champ facultatif pour GitHub Enterprise :

```
aws secretsmanager create-secret \
  --name argocd/github-enterprise-app \
  --secret-string '{

 "githubAppPrivateKeySecret":"LS0tLS1CRUdJTiBSU0EgUFJJVkFURSBLRVktLS0tLQouLi4KLS0tLS1FTkQgUlNBI
    "githubAppID":"123456",
    "githubAppInstallationID":"12345678",
    "githubAppEnterpriseBaseUrl":"https://github.example.com/api/v3"
  }'
```

Créez un secret de dépôt pour GitHub l'application :

```
apiVersion: v1
kind: Secret
metadata:
  name: my-repo-github-app
  namespace: argocd
  labels:
    argocd.argoproj.io/secret-type: repository
stringData:
  type: git
  url: https://github.com/your-org/your-repo
  secretArn: arn:aws:secretsmanager:us-west-2:111122223333:secret:argocd/github-app-
AbCdEf
  project: default
```

Modèles d'informations d'identification du référentiel

Pour la réutilisation des informations d'identification dans plusieurs référentiels (par exemple, tous les référentiels d'une GitHub organisation ou d'un utilisateur), utilisez des modèles d'informations d'identification de référentiel avec. `argocd.argoproj.io/secret-type: repo-creds` Cela fournit une meilleure expérience utilisateur que la création de secrets de dépôt individuels pour chaque référentiel.

Créez un modèle d'identification de référentiel :

```
apiVersion: v1
kind: Secret
metadata:
  name: github-org-creds
```

```
  namespace: argocd
  labels:
    argocd.argoproj.io/secret-type: repo-creds
stringData:
  type: git
  url: https://github.com/your-org
  secretArn: arn:aws:secretsmanager:us-west-2:111122223333:secret:argocd/github-org-
AbCdEf
```

Ce modèle d'identification s'applique à tous les référentiels correspondant au préfixe d'URL. `https://github.com/your-org` Vous pouvez ensuite référencer n'importe quel référentiel relevant de cette organisation dans Applications sans créer de secrets supplémentaires.

Pour plus d'informations, consultez la section Informations d'identification du référentiel dans la documentation Argo CD.

> **Important**
>
> Assurez-vous que votre rôle de capacité IAM est `arn:aws:iam::aws:policy/AWSSecretsManagerClientReadOnlyAccess` associé à la politique gérée ou à des autorisations équivalentes, y compris `secretsmanager:GetSecretValue` des autorisations de déchiffrement KMS. Voir the section called "Considérations relatives à Argo CD" pour la configuration de la politique IAM.

En utilisant AWS CodeConnections

Pour CodeConnections l'intégration, voirthe section called " AWS CodeConnections".

CodeConnections fournit une authentification gérée pour GitHub GitLab, et Bitbucket sans stocker les informations d'identification.

Utilisation des secrets de Kubernetes

Stockez les informations d'identification directement dans Kubernetes à l'aide de la méthode Argo CD standard.

Pour le protocole HTTPS avec jeton d'accès personnel :

```
apiVersion: v1
```

```
kind: Secret
metadata:
  name: my-repo
  namespace: argocd
  labels:
    argocd.argoproj.io/secret-type: repository
stringData:
  type: git
  url: https://github.com/your-org/your-repo
  username: your-username
  password: your-personal-access-token
```

Pour SSH :

```
apiVersion: v1
kind: Secret
metadata:
  name: my-repo-ssh
  namespace: argocd
  labels:
    argocd.argoproj.io/secret-type: repository
stringData:
  type: git
  url: git@github.com:your-org/your-repo.git
  sshPrivateKey: |
    -----BEGIN OPENSSH PRIVATE KEY-----
    ... your private key ...
    -----END OPENSSH PRIVATE KEY-----
```

CodeCommit référentiels

Pour AWS CodeCommit, accordez les CodeCommit autorisations de votre rôle de capacité IAM (`codecommit:GitPull`).

Configurez le référentiel :

```
apiVersion: v1
kind: Secret
metadata:
  name: codecommit-repo
  namespace: argocd
  labels:
```

```
    argocd.argoproj.io/secret-type: repository
stringData:
  type: git
  url: https://git-codecommit.us-west-2.amazonaws.com/v1/repos/my-repo
  project: default
```

Pour une configuration détaillée de la politique IAM, consultezthe section called "Considérations relatives à Argo CD".

Vérifier la connexion au référentiel

Vérifiez l'état de la connexion via l'interface utilisateur d'Argo CD sous Paramètres → Référentiels. L'interface utilisateur indique l'état de la connexion et les éventuelles erreurs d'authentification.

Les secrets du référentiel n'incluent pas d'informations sur le statut.

Ressources supplémentaires

- the section called "Enregistrer des clusters"- Enregistrez les clusters cibles pour les déploiements
- the section called "Création d'applications"- Créez votre première application
- the section called "Considérations relatives à Argo CD"- Autorisations IAM et configuration de sécurité
- Référentiels privés - Référence de configuration des référentiels en amont

Enregistrer les clusters cibles

Enregistrez les clusters pour permettre à Argo CD d'y déployer des applications. Vous pouvez enregistrer le même cluster sur lequel Argo CD est en cours d'exécution (cluster local) ou des clusters distants dans différents comptes ou régions. Une fois qu'un cluster est enregistré, il reste dans un état de connexion inconnu jusqu'à ce que vous créiez une application au sein de ce cluster. Pour créer une application Argo CD après l'enregistrement de votre cluster, consultezthe section called "Création d'applications".

Conditions préalables

- Création d'un cluster EKS doté de la fonctionnalité Argo CD
- `kubectl`configuré pour communiquer avec votre cluster
- Pour les clusters distants : autorisations IAM et entrées d'accès appropriées

Enregistrez le cluster local

Pour déployer des applications sur le même cluster que celui sur lequel Argo CD est exécuté, enregistrez-le en tant que cible de déploiement.

> **Important**
>
> La fonctionnalité Argo CD n'enregistre pas automatiquement le cluster local. Vous devez l'enregistrer explicitement pour déployer des applications sur le même cluster. Vous pouvez utiliser le nom du cluster `in-cluster` pour assurer la compatibilité avec la plupart des exemples de CD Argo en ligne.

> **Note**
>
> Une entrée d'accès EKS est automatiquement créée pour le cluster local avec le rôle Argo CD Capability, mais aucune autorisation RBAC Kubernetes n'est accordée par défaut. Cela suit le principe du moindre privilège : vous devez configurer explicitement les autorisations dont Argo CD a besoin en fonction de votre cas d'utilisation. Par exemple, si vous utilisez ce cluster uniquement comme hub Argo CD pour gérer des clusters distants, aucune autorisation de déploiement local n'est requise. Consultez la section relative aux exigences RBAC relatives aux entrées d'accès ci-dessous pour connaître les options de configuration.

À l'aide de l'Argo CD CLI :

```
argocd cluster add <cluster-context-name> \
  --aws-cluster-name arn:aws:eks:us-west-2:111122223333:cluster/my-cluster \
  --name local-cluster
```

Utilisation d'un secret Kubernetes :

```
apiVersion: v1
kind: Secret
metadata:
  name: local-cluster
  namespace: argocd
  labels:
    argocd.argoproj.io/secret-type: cluster
```

```
stringData:
  name: local-cluster
  server: arn:aws:eks:us-west-2:111122223333:cluster/my-cluster
  project: default
```

Appliquez la configuration :

```
kubectl apply -f local-cluster.yaml
```

> **Note**
>
> Utilisez l'ARN du cluster EKS `server` sur le terrain, et non l'URL du serveur d'API Kubernetes. La fonctionnalité gérée nécessite ARNs d'identifier les clusters. La valeur par défaut n'`kubernetes.default.svc`est pas prise en charge.

Enregistrer des clusters distants

Pour déployer sur des clusters distants :

Étape 1 : créer l'entrée d'accès sur le cluster distant

Remplacez *`region-code`* par la AWS région dans laquelle se trouve votre cluster distant, remplacez *`remote-cluster`* par le nom de votre cluster distant et remplacez l'ARN par l'ARN de votre rôle de fonctionnalité Argo CD.

```
aws eks create-access-entry \
  --region region-code \
  --cluster-name remote-cluster \
  --principal-arn arn:aws:iam::[.replaceable]111122223333:role/ArgoCDCapabilityRole \
  --type STANDARD
```

Étape 2 : associer une politique d'accès aux autorisations Kubernetes RBAC

L'entrée d'accès nécessite des autorisations RBAC Kubernetes pour qu'Argo CD puisse déployer des applications. Pour démarrer rapidement, vous pouvez utiliser `AmazonEKSClusterAdminPolicy` :

```
aws eks associate-access-policy \
  --region region-code \
  --cluster-name remote-cluster \
```

```
  --principal-arn arn:aws:iam::[.replaceable]111122223333:role/ArgoCDCapabilityRole \
  --policy-arn arn:aws:eks::aws:cluster-access-policy/AmazonEKSClusterAdminPolicy \
  --access-scope type=cluster
```

Important

`AmazonEKSClusterAdminPolicy`Fournit un accès complet à l'administrateur du cluster (équivalent à). `system:masters` C'est pratique pour démarrer mais ne doit pas être utilisé en production. Pour les environnements de production, utilisez des autorisations plus restrictives en associant l'entrée d'accès à des groupes Kubernetes personnalisés et en créant un rôle ou des liaisons appropriés. ClusterRole Consultez la section relative à la configuration de production ci-dessous pour la configuration du moindre privilège.

Étape 3 : enregistrer le cluster dans Argo CD

À l'aide de l'Argo CD CLI :

```
argocd cluster add <cluster-context-name> \
  --aws-cluster-name arn:aws:eks:us-west-2:111122223333:cluster/remote-cluster \
  --name remote-cluster
```

Utilisation d'un secret Kubernetes :

```
apiVersion: v1
kind: Secret
metadata:
  name: remote-cluster
  namespace: argocd
  labels:
    argocd.argoproj.io/secret-type: cluster
stringData:
  name: remote-cluster
  server: arn:aws:eks:us-west-2:111122223333:cluster/remote-cluster
  project: default
```

Appliquez la configuration :

```
kubectl apply -f remote-cluster.yaml
```

Clusters entre comptes

Pour effectuer un déploiement sur des clusters appartenant à différents AWS comptes, procédez comme suit :

1. Dans le compte cible, créez une entrée d'accès sur le cluster EKS cible en utilisant l'Argo CD IAM Capability Role (ARN) du compte source comme principal.
2. Associer une politique d'accès aux autorisations Kubernetes RBAC appropriées
3. Enregistrez le cluster dans Argo CD à l'aide de son ARN de cluster EKS

Aucune création de rôle IAM ou configuration de politique de confiance supplémentaire n'est requise : les entrées d'accès EKS gèrent l'accès entre comptes.

Le format ARN du cluster inclut la région, de sorte que les déploiements entre régions utilisent le même processus que les déploiements dans la même région.

Vérifier l'enregistrement du cluster

Afficher les clusters enregistrés :

```
kubectl get secrets -n argocd -l argocd.argoproj.io/secret-type=cluster
```

Ou vérifiez l'état du cluster dans l'interface utilisateur d'Argo CD sous Paramètres → Clusters.

Clusters privés

La fonctionnalité Argo CD fournit un accès transparent à des clusters EKS entièrement privés sans nécessiter de peering VPC ou de configuration réseau spécialisée.

AWS gère automatiquement la connectivité entre la fonctionnalité Argo CD et les clusters distants privés.

Enregistrez simplement le cluster privé à l'aide de son ARN, aucune configuration réseau supplémentaire n'est requise.

Exigences RBAC d'accès et d'entrée

Lorsque vous créez une fonctionnalité Argo CD, une entrée d'accès EKS est automatiquement créée pour le rôle de capacité, mais aucune autorisation RBAC Kubernetes n'est accordée par défaut. Cette

conception intentionnelle suit le principe du moindre privilège : différents cas d'utilisation nécessitent des autorisations différentes.

Par exemple :* Si vous utilisez le cluster uniquement comme hub Argo CD pour gérer des clusters distants, il n'a pas besoin d'autorisations de déploiement local * Si vous déployez des applications localement, il a besoin d'un accès en lecture à l'échelle du cluster et d'un accès en écriture à des espaces de noms spécifiques * Si vous devez créer CRDs, il nécessite des autorisations d'administrateur de cluster supplémentaires

Vous devez configurer explicitement les autorisations dont Argo CD a besoin en fonction de vos besoins.

Autorisations minimales pour Argo CD

Argo CD a besoin de deux types d'autorisations pour fonctionner sans erreur :

Autorisations de lecture (à l'échelle du cluster) : Argo CD doit être capable de lire tous les types de ressources et les définitions de ressources personnalisées (CRDs) du cluster pour :

- Découverte des ressources et bilans de santé
- Détection de la dérive entre l'état souhaité et l'état réel
- Validation des ressources avant le déploiement

Autorisations d'écriture (spécifiques à l'espace de noms) : Argo CD a besoin d'autorisations de création, de mise à jour et de suppression pour les ressources définies dans Applications :

- Déployez les charges de travail des applications (déploiements, services ConfigMaps, etc.)
- Appliquer des ressources personnalisées (CRDs spécifiques à vos applications)
- Gérer le cycle de vie des applications

Configuration rapide

Pour démarrer rapidement, tester ou développer des environnements, utilisez `AmazonEKSClusterAdminPolicy` :

```
aws eks associate-access-policy \
  --region region-code \
  --cluster-name my-cluster \
```

```
  --principal-arn arn:aws:iam::[.replaceable]111122223333:role/ArgoCDCapabilityRole \
  --policy-arn arn:aws:eks::aws:cluster-access-policy/AmazonEKSClusterAdminPolicy \
  --access-scope type=cluster
```

Important

`AmazonEKSClusterAdminPolicy`Il fournit un accès complet à l'administrateur du cluster (équivalent à`system:masters`), y compris la possibilité de créer CRDs, de modifier des ressources à l'échelle du cluster et de les déployer dans n'importe quel espace de noms. C'est pratique pour le développement POCs mais ne doit pas être utilisé en production. Pour la production, utilisez la configuration de moindre privilège ci-dessous.

Configuration de production avec le moins de privilèges

Pour les environnements de production, créez un RBAC Kubernetes personnalisé qui accorde :

- Accès en lecture à toutes les ressources à l'échelle du cluster (pour la découverte et les bilans de santé)
- Accès en écriture spécifique à l'espace de noms (pour les déploiements)

Étape 1 : associer l'entrée d'accès à un groupe Kubernetes personnalisé

```
aws eks associate-access-policy \
  --region region-code \
  --cluster-name my-cluster \
  --principal-arn arn:aws:iam::[.replaceable]111122223333:role/ArgoCDCapabilityRole \
  --policy-arn arn:aws:eks::aws:cluster-access-policy/AmazonEKSEditPolicy \
  --access-scope type=namespace,namespaces=app-namespace
```

Étape 2 : Création ClusterRole pour un accès en lecture

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: argocd-read-all
rules:
# Read access to all resources for discovery and health checks
```

```
- apiGroups: ["*"]
  resources: ["*"]
  verbs: ["get", "list", "watch"]
```

Étape 3 : créer un rôle pour l'accès en écriture aux espaces de noms des applications

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: argocd-deploy
  namespace: app-namespace
rules:
# Full access to deploy application resources
- apiGroups: ["*"]
  resources: ["*"]
  verbs: ["*"]
```

Étape 4 : lier les rôles au groupe Kubernetes

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: argocd-read-all
subjects:
- kind: Group
  name: eks-access-entry:arn:aws:iam::111122223333:role/ArgoCDCapabilityRole
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: argocd-read-all
  apiGroup: rbac.authorization.k8s.io
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: argocd-deploy
  namespace: app-namespace
subjects:
- kind: Group
  name: eks-access-entry:arn:aws:iam::111122223333:role/ArgoCDCapabilityRole
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role
```

```
  name: argocd-deploy
  apiGroup: rbac.authorization.k8s.io
```

> **Note**
>
> Le format du nom de groupe pour les entrées d'accès est `eks-access-entry:` suivi par l'ARN principal. Répétez cette opération RoleBinding pour chaque espace de noms dans lequel Argo CD doit déployer des applications.

> **Important**
>
> Argo CD doit être capable de lire tous les types de ressources du cluster à des fins de vérification de l'état et de découverte, même s'il n'est déployé que dans des espaces de noms spécifiques. Sans accès en lecture à l'échelle du cluster, Argo CD affichera des erreurs lors de la vérification de l'état de l'application.

Restreindre l'accès au cluster avec Projects

Utilisez les projets pour contrôler les clusters et les espaces de noms dans lesquels les applications peuvent être déployées en configurant les clusters et espaces de noms cibles autorisés dans : `spec.destinations`

```
apiVersion: argoproj.io/v1alpha1
kind: AppProject
metadata:
  name: production
  namespace: argocd
spec:
  destinations:
  - server: arn:aws:eks:us-west-2:111122223333:cluster/prod-cluster
    namespace: '*'
  - server: arn:aws:eks:eu-west-1:111122223333:cluster/prod-eu-cluster
    namespace: '*'
  sourceRepos:
  - 'https://github.com/example/production-apps'
```

Pour en savoir plus, consultez the section called “Projets”.

Ressources supplémentaires

- the section called "Projets"- Organisez les applications et appliquez les limites de sécurité
- the section called "Création d'applications"- Déployez votre première application
- the section called "ApplicationSets"- Déployez sur plusieurs clusters avec ApplicationSets
- the section called "Considérations relatives à Argo CD"- Modèles multi-clusters et configuration entre comptes
- Configuration déclarative du cluster - Référence de configuration du cluster en amont

Travailler avec des projets Argo CD

Les projets Argo CD (AppProject) fournissent un regroupement logique et un contrôle d'accès pour les applications. Les projets définissent les référentiels Git, les clusters cibles et les espaces de noms que les applications peuvent utiliser, ce qui permet la mutualisation et les limites de sécurité dans les instances Argo CD partagées.

Quand utiliser Projects

Utilisez les projets pour :

- Séparez les applications par équipe, environnement ou unité commerciale
- Limiter les référentiels à partir desquels les équipes peuvent effectuer des déploiements
- Limiter les clusters et les espaces de noms vers lesquels les équipes peuvent effectuer des déploiements
- Appliquer les quotas de ressources et les types de ressources autorisés
- Proposez un déploiement d'applications en libre-service avec des garde-fous

Projet par défaut

Chaque fonctionnalité d'Argo CD inclut un `default` projet qui permet d'accéder à tous les référentiels, clusters et espaces de noms. Bien que cela soit utile pour les tests initiaux, créez des projets dédiés avec des restrictions explicites pour l'utilisation en production.

Pour plus de détails sur la configuration de projet par défaut et sur la manière de la restreindre, consultez The Default Project dans la documentation d'Argo CD.

Créer un projet

Créez un projet en appliquant une `AppProject` ressource à votre cluster.

Exemple : projet spécifique à une équipe

```
apiVersion: argoproj.io/v1alpha1
kind: AppProject
metadata:
  name: team-a
  namespace: argocd
spec:
  description: Applications for Team A

  # Source repositories this project can deploy from
  sourceRepos:
    - 'https://github.com/my-org/team-a-*'
    - 'https://github.com/my-org/shared-libs'

  # Destination clusters and namespaces
  destinations:
    - name: dev-cluster
      namespace: team-a-dev
    - name: prod-cluster
      namespace: team-a-prod

  # Allowed resource types
  clusterResourceWhitelist:
    - group: ''
      kind: Namespace

  namespaceResourceWhitelist:
    - group: 'apps'
      kind: Deployment
    - group: ''
      kind: Service
    - group: ''
      kind: ConfigMap
```

Appliquez le projet :

```
kubectl apply -f team-a-project.yaml
```

Configuration du projet

Référentiels de sources

Contrôlez les référentiels Git que les applications de ce projet peuvent utiliser :

```
spec:
  sourceRepos:
    - 'https://github.com/my-org/app-*'  # Wildcard pattern
    - 'https://github.com/my-org/infra'  # Specific repo
```

Vous pouvez utiliser des caractères génériques et des modèles de négation (`!` préfixe) pour autoriser ou refuser des référentiels spécifiques. Pour plus de détails, consultez la section Gestion de projets dans la documentation d'Argo CD.

Restrictions de destination

Limite dans laquelle les applications peuvent être déployées :

```
spec:
  destinations:
    - name: prod-cluster  # Specific cluster by name
      namespace: production
    - name: '*'  # Any cluster
      namespace: team-a-*  # Namespace pattern
```

Important

Utilisez des noms de cluster et des modèles d'espaces de noms spécifiques plutôt que des caractères génériques pour les projets de production. Cela permet d'éviter les déploiements accidentels vers des clusters ou des espaces de noms non autorisés.

Vous pouvez utiliser des caractères génériques et des modèles de négation pour contrôler les destinations. Pour plus de détails, consultez la section Gestion de projets dans la documentation d'Argo CD.

Restrictions de ressources

Contrôlez les types de ressources Kubernetes qui peuvent être déployés :

Ressources étendues au cluster :

```
spec:
  clusterResourceWhitelist:
    - group: ''
      kind: Namespace
    - group: 'rbac.authorization.k8s.io'
      kind: Role
```

Ressources limitées à l'espace de noms :

```
spec:
  namespaceResourceWhitelist:
    - group: 'apps'
      kind: Deployment
    - group: ''
      kind: Service
    - group: ''
      kind: ConfigMap
    - group: 's3.services.k8s.aws'
      kind: Bucket
```

Utilisez des listes noires pour refuser des ressources spécifiques :

```
spec:
  namespaceResourceBlacklist:
    - group: ''
      kind: Secret  # Prevent direct Secret creation
```

Affecter des applications à des projets

Lors de la création d'une application, spécifiez le projet dans le `spec.project` champ :

```
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: my-app
  namespace: argocd
spec:
  project: team-a  # Assign to team-a project
  source:
```

```
    repoURL: https://github.com/my-org/my-app
    path: manifests
  destination:
    name: prod-cluster
    namespace: team-a-prod
```

Les applications sans projet spécifique utilisent le `default` projet.

Rôles du projet et RBAC

Les projets peuvent définir des rôles personnalisés pour un contrôle d'accès précis. Associez les rôles du projet aux utilisateurs et aux groupes d' AWS Identity Center dans la configuration de vos fonctionnalités afin de contrôler qui peut synchroniser, mettre à jour ou supprimer des applications.

Exemple : projet avec rôles de développeur et d'administrateur

```
apiVersion: argoproj.io/v1alpha1
kind: AppProject
metadata:
  name: team-a
  namespace: argocd
spec:
  sourceRepos:
    - '*'
  destinations:
    - name: '*'
      namespace: 'team-a-*'

  roles:
    - name: developer
      description: Developers can sync applications
      policies:
        - p, proj:team-a:developer, applications, sync, team-a/*, allow
        - p, proj:team-a:developer, applications, get, team-a/*, allow
      groups:
        - team-a-developers

    - name: admin
      description: Admins have full access
      policies:
        - p, proj:team-a:admin, applications, *, team-a/*, allow
      groups:
        - team-a-admins
```

Pour plus de détails sur les rôles du projet, les jetons JWT pour les CI/CD pipelines et la configuration du RBAC, consultez la section Rôles du projet dans la documentation Argo CD.

Schémas courants

Projets axés sur l'environnement

Créez des projets distincts pour chaque environnement :

```
apiVersion: argoproj.io/v1alpha1
kind: AppProject
metadata:
  name: production
  namespace: argocd
spec:
  sourceRepos:
    - 'https://github.com/my-org/*'
  destinations:
    - name: prod-cluster
      namespace: '*'
  # Strict resource controls for production
  clusterResourceWhitelist: []
  namespaceResourceWhitelist:
    - group: 'apps'
      kind: Deployment
    - group: ''
      kind: Service
```

Projets basés sur le travail d'équipe

Isolez les équipes avec des projets dédiés :

```
apiVersion: argoproj.io/v1alpha1
kind: AppProject
metadata:
  name: platform-team
  namespace: argocd
spec:
  sourceRepos:
    - 'https://github.com/my-org/platform-*'
  destinations:
    - name: '*'
      namespace: 'platform-*'
  # Platform team can manage cluster resources
```

```
  clusterResourceWhitelist:
    - group: '*'
      kind: '*'
```

Projets multi-clusters

Déployez sur plusieurs clusters avec des politiques cohérentes :

```
apiVersion: argoproj.io/v1alpha1
kind: AppProject
metadata:
  name: global-app
  namespace: argocd
spec:
  sourceRepos:
    - 'https://github.com/my-org/global-app'
  destinations:
    - name: us-west-cluster
      namespace: app
    - name: eu-west-cluster
      namespace: app
    - name: ap-south-cluster
      namespace: app
```

Bonnes pratiques

Commencez par des projets restrictifs : commencez par des autorisations restreintes et étendez-les selon les besoins plutôt que de commencer par un accès étendu.

Utilisez des modèles d'espaces de noms : utilisez des caractères génériques dans les restrictions d'espace de noms (par exemple`team-a-*`) pour permettre de la flexibilité tout en maintenant les limites.

Projets de production distincts : utilisez des projets dédiés pour la production avec des contrôles plus stricts et des politiques de synchronisation manuelle.

Documentez les objectifs du projet : utilisez le `description` champ pour expliquer à quoi sert chaque projet et qui doit l'utiliser.

Vérifiez régulièrement les autorisations des projets : auditez régulièrement les projets pour vous assurer que les restrictions correspondent toujours aux besoins de l'équipe et aux exigences de sécurité.

Ressources supplémentaires

- the section called “Configurer les autorisations ”- Configurer l'intégration du RBAC et de l'Identity Center
- the section called “Création d'applications”- Créez des applications dans des projets
- the section called “ApplicationSets”- À utiliser ApplicationSets avec des projets pour des déploiements multi-clusters
- Documentation des projets Argo CD - Référence complète en amont

Création d'applications

Les applications représentent des déploiements dans des clusters cibles. Chaque application définit une source (dépôt Git) et une destination (cluster et espace de noms). Une fois appliqué, Argo CD créera les ressources spécifiées par des manifestes dans le référentiel Git dans l'espace de noms du cluster. Les applications spécifient souvent des déploiements de charges de travail, mais elles peuvent gérer toutes les ressources Kubernetes disponibles dans le cluster de destination.

Conditions préalables

- Création d'un cluster EKS doté de la fonctionnalité Argo CD
- Accès au référentiel configuré (voirthe section called “Configuration des référentiels”)
- Cluster cible enregistré (voirthe section called “Enregistrer des clusters”)
- `kubectl`configuré pour communiquer avec votre cluster

Création d'une application de base

Définissez une application déployée à partir d'un dépôt Git :

```
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: guestbook
  namespace: argocd
spec:
  project: default
  source:
    repoURL: https://github.com/argoproj/argocd-example-apps
    targetRevision: HEAD
```

```
    path: guestbook
  destination:
    name: in-cluster
    namespace: default
```

> **Note**
>
> `destination.name`À utiliser avec le nom du cluster que vous avez utilisé lors de l'enregistrement du cluster (comme `in-cluster` pour le cluster local). Le `destination.server` champ fonctionne également avec le cluster EKS ARNs, mais l'utilisation de noms de clusters est recommandée pour une meilleure lisibilité.

Appliquez l'application :

```
kubectl apply -f application.yaml
```

Afficher le statut de la demande :

```
kubectl get application guestbook -n argocd
```

Configuration de la source

Référentiel Git :

```
spec:
  source:
    repoURL: https://github.com/example/my-app
    targetRevision: main
    path: kubernetes/manifests
```

Tag ou commit Git spécifique :

```
spec:
  source:
    targetRevision: v1.2.0  # or commit SHA
```

Tableau du casque :

```
spec:
  source:
    repoURL: https://github.com/example/helm-charts
    targetRevision: main
    path: charts/my-app
    helm:
      valueFiles:
      - values.yaml
      parameters:
      - name: image.tag
        value: v1.2.0
```

Diagramme de Helm avec des valeurs provenant d'un dépôt Git externe (modèle multi-source) :

```
spec:
  sources:
  - repoURL: https://github.com/example/helm-charts
    targetRevision: main
    path: charts/my-app
    helm:
      valueFiles:
      - $values/environments/production/values.yaml
  - repoURL: https://github.com/example/config-repo
    targetRevision: main
    ref: values
```

Pour plus d'informations, consultez la section Fichiers Helm Value provenant d'un référentiel Git externe dans la documentation du CD Argo.

Tableau de bord de l'ECR :

```
spec:
  source:
    repoURL: oci://account-id.dkr.ecr.region.amazonaws.com/repository-name
    targetRevision: chart-version
    chart: chart-name
```

Si le rôle de capacité dispose des autorisations ECR requises, le référentiel est utilisé directement et aucune configuration de référentiel n'est requise. Consultez the section called “Configuration des référentiels” pour plus de détails.

Dépôt Git provenant de CodeCommit :

```
spec:
  source:
    repoURL: https://git-codecommit.region.amazonaws.com/v1/repos/repository-name
    targetRevision: main
    path: kubernetes/manifests
```

Si le rôle de capacité dispose des CodeCommit autorisations requises, le référentiel est utilisé directement et aucune configuration de référentiel n'est requise. Consultez the section called "Configuration des référentiels" pour plus de détails.

Dépôt Git provenant de CodeConnections :

```
spec:
  source:
    repoURL: https://codeconnections.region.amazonaws.com/git-http/account-
id/region/connection-id/owner/repository.git
    targetRevision: main
    path: kubernetes/manifests
```

Le format de l'URL du référentiel est dérivé de l'ARN de CodeConnections connexion. Si le rôle de capacité dispose des CodeConnections autorisations requises et qu'une connexion est configurée, le référentiel est utilisé directement et aucune configuration de référentiel n'est requise. Consultez the section called "Configuration des référentiels" pour plus de détails.

Personnalisez :

```
spec:
  source:
    repoURL: https://github.com/example/kustomize-app
    targetRevision: main
    path: overlays/production
    kustomize:
      namePrefix: prod-
```

Politiques de synchronisation

Contrôlez la façon dont Argo CD synchronise les applications.

Synchronisation manuelle (par défaut) :

Les applications nécessitent une approbation manuelle pour être synchronisées :

```
spec:
  syncPolicy: {}  # No automated sync
```

Déclenchez manuellement la synchronisation :

```
kubectl patch application guestbook -n argocd \
  --type merge \
  --patch '{"operation": {"initiatedBy": {"username": "admin"}, "sync": {}}}'
```

Synchronisation automatique :

Les applications se synchronisent automatiquement lorsque des modifications de Git sont détectées :

```
spec:
  syncPolicy:
    automated: {}
```

Auto-guérison :

Annulez automatiquement les modifications manuelles apportées au cluster :

```
spec:
  syncPolicy:
    automated:
      selfHeal: true
```

Lorsqu'il est activé, Argo CD annule toutes les modifications manuelles apportées directement au cluster, garantissant ainsi que Git reste la source de vérité.

Élagage :

Supprimez automatiquement les ressources supprimées de Git :

```
spec:
  syncPolicy:
    automated:
      prune: true
```

> **Warning**
>
> L'élagage supprimera les ressources de votre cluster. À utiliser avec prudence dans les environnements de production.

Synchronisation automatique combinée :

```
spec:
  syncPolicy:
    automated:
      prune: true
      selfHeal: true
    syncOptions:
    - CreateNamespace=true
```

Réessayez la configuration :

Configurez le comportement des nouvelles tentatives en cas d'échec des synchronisations :

```
spec:
  syncPolicy:
    retry:
      limit: 5  # Number of failed sync attempts; unlimited if less than 0
      backoff:
        duration: 5s  # Amount to back off (default unit: seconds, also supports "2m",
 "1h")
        factor: 2  # Factor to multiply the base duration after each failed retry
        maxDuration: 3m  # Maximum amount of time allowed for the backoff strategy
```

Cela est particulièrement utile pour les ressources qui doivent d'abord CRDs être créées, ou lorsque vous travaillez avec des instances kro où le CRD n'est peut-être pas immédiatement disponible.

Options de synchronisation

Configuration de synchronisation supplémentaire :

Créez un espace de noms s'il n'existe pas :

```
spec:
  syncPolicy:
```

```
    syncOptions:
    - CreateNamespace=true
```

Ignorez le cycle à sec en cas de ressources manquantes :

Utile lors de l'application de ressources qui dépendent de celles CRDs qui n'existent pas encore (comme les instances Kro) :

```
spec:
  syncPolicy:
    syncOptions:
    - SkipDryRunOnMissingResource=true
```

Cela peut également être appliqué à des ressources spécifiques à l'aide d'une étiquette apposée sur la ressource elle-même.

Validez les ressources avant de les appliquer :

```
spec:
  syncPolicy:
    syncOptions:
    - Validate=true
```

Appliquer de manière désynchronisée uniquement :

```
spec:
  syncPolicy:
    syncOptions:
    - ApplyOutOfSyncOnly=true
```

Fonctionnalités de synchronisation avancées

Argo CD prend en charge les fonctionnalités de synchronisation avancées pour les déploiements complexes :

- Synchronisation des ondes - Contrôlez l'ordre de création des ressources à l'aide d'`argocd.argoproj.io/sync-wave`annotations
- Crochets de synchronisation - Exécutez des tâches avant ou après la synchronisation avec `argocd.argoproj.io/hook` les annotations (PreSync, PostSync, SyncFail)

- Évaluation de l'état des ressources - Contrôles de santé personnalisés pour les ressources spécifiques à l'application

Pour plus de détails, consultez Sync Waves et Resource Hooks dans la documentation d'Argo CD.

Ignorer les différences

Empêchez Argo CD de synchroniser des champs spécifiques gérés par d'autres contrôleurs (comme la gestion des répliques par HPA) :

```
spec:
  ignoreDifferences:
  - group: apps
    kind: Deployment
    jsonPointers:
    - /spec/replicas
```

Pour plus de détails sur les modèles d'ignorance et les exclusions de champs, voir Diffing Customization dans la documentation d'Argo CD.

Déploiement dans plusieurs environnements

Déployez la même application dans plusieurs environnements :

Développement :

```
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: my-app-dev
  namespace: argocd
spec:
  project: default
  source:
    repoURL: https://github.com/example/my-app
    targetRevision: develop
    path: overlays/development
  destination:
    name: dev-cluster
    namespace: my-app
```

Produktion :

```
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: my-app-prod
  namespace: argocd
spec:
  project: default
  source:
    repoURL: https://github.com/example/my-app
    targetRevision: main
    path: overlays/production
  destination:
    name: prod-cluster
    namespace: my-app
  syncPolicy:
    automated:
      prune: true
      selfHeal: true
```

Surveiller et gérer les applications

Afficher le statut de la demande :

```
kubectl get application my-app -n argocd
```

Accédez à l'interface utilisateur d'Argo CD :

Ouvrez l'interface utilisateur du CD Argo via la console EKS pour afficher la topologie de l'application, l'état de synchronisation, l'état des ressources et l'historique des déploiements. Consultez les instructions the section called “Travailler avec Argo CD” d'accès à l'interface utilisateur.

Applications de rétrogradation :

Revenez à une version précédente à l'aide de l'interface utilisateur Argo CD, de l'Argo CD CLI, ou `targetRevision` en mettant à jour la spécification de l'application avec un commit ou une balise Git précédent.

À l'aide de l'Argo CD CLI :

```
argocd app rollback argocd/my-app <revision-id>
```

> **Note**
>
> Lorsque vous utilisez l'Argo CD CLI avec la fonctionnalité gérée, spécifiez les applications avec le préfixe d'espace de noms :. `namespace/appname`

Pour plus d'informations, consultez la section restauration de l'application argocd dans la documentation d'Argo CD.

Ressources supplémentaires

- the section called "Projets"- Organisez les applications avec des projets pour les environnements à locataires multiples
- the section called "ApplicationSets"- Déployez sur plusieurs clusters à l'aide de modèles
- Spécification de l'application - Référence complète de l'API de l'application
- Options de synchronisation - Configuration de synchronisation avancée

Utiliser ApplicationSets

ApplicationSets générez plusieurs applications à partir de modèles, ce qui vous permet de déployer la même application sur plusieurs clusters, environnements ou espaces de noms avec une seule définition de ressource.

Conditions préalables

- Création d'un cluster EKS doté de la fonctionnalité Argo CD
- Accès au référentiel configuré (voirthe section called "Configuration des référentiels")
- `kubectl`configuré pour communiquer avec votre cluster

> **Note**
>
> Il n'est pas nécessaire de disposer de plusieurs clusters cibles pour ApplicationSets. Vous pouvez utiliser des générateurs autres que le générateur de clusters (tels que des générateurs de listes, de git ou de matrices) pour déployer des applications sans clusters distants.

Comment ApplicationSets travailler

ApplicationSets utilisez des générateurs pour produire des paramètres, puis appliquez ces paramètres à un modèle d'application. Chaque ensemble de paramètres générés crée une application.

Générateurs courants pour les déploiements EKS :

- Générateur de listes - Définissez explicitement les clusters et les paramètres pour chaque environnement
- Générateur de clusters - Déploiement automatique sur tous les clusters enregistrés
- Générateur Git - Génère des applications à partir de la structure du référentiel
- Générateur de matrices - Combinez des générateurs pour des déploiements multidimensionnels
- Générateur de fusion - Fusionnez les paramètres de plusieurs générateurs

Pour une référence complète sur le générateur, consultez ApplicationSet la documentation.

Générateur de listes

Déployez sur plusieurs clusters avec une configuration explicite :

```
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: guestbook-all-clusters
  namespace: argocd
spec:
  generators:
  - list:
      elements:
      - environment: dev
        replicas: "2"
      - environment: staging
        replicas: "3"
      - environment: prod
        replicas: "5"
  template:
    metadata:
      name: 'guestbook-{{environment}}'
    spec:
```

```
    project: default
    source:
      repoURL: https://github.com/example/guestbook
      targetRevision: HEAD
      path: 'overlays/{{environment}}'
    destination:
      name: '{{environment}}-cluster'
      namespace: guestbook
    syncPolicy:
      automated:
        prune: true
        selfHeal: true
```

> **Note**
>
> À utiliser `destination.name` avec les noms de clusters pour une meilleure lisibilité. Le `destination.server` champ fonctionne également avec le cluster EKS ARNs si nécessaire.

Cela crée trois applications : `guestbook-dev`guestbook-staging, et`guestbook-prod`.

Générateur de clusters

Déployez automatiquement sur tous les clusters enregistrés :

```
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: cluster-addons
  namespace: argocd
spec:
  generators:
  - clusters: {}
  template:
    metadata:
      name: '{{name}}-addons'
    spec:
      project: default
      source:
        repoURL: https://github.com/example/cluster-addons
        targetRevision: HEAD
```

```
        path: addons
      destination:
        server: '{{server}}'
        namespace: kube-system
      syncPolicy:
        automated:
          prune: true
          selfHeal: true
```

Cela crée automatiquement une application pour chaque cluster enregistré.

Clusters de filtres :

`matchLabels`À utiliser pour inclure des clusters spécifiques ou `matchExpressions` pour exclure des clusters :

```
spec:
  generators:
  - clusters:
      selector:
        matchLabels:
          environment: production
        matchExpressions:
        - key: skip-appset
          operator: DoesNotExist
```

Générateurs Git

Les générateurs Git créent des applications basées sur la structure du référentiel :

- Générateur de répertoires - Déployez chaque répertoire en tant qu'application distincte (utile pour les microservices)
- Générateur de fichiers : générez des applications à partir de fichiers de paramètres (utile pour les déploiements multi-locataires)

Exemple : déploiement de microservices

```
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: microservices
```

```
  namespace: argocd
spec:
  generators:
  - git:
      repoURL: https://github.com/example/microservices
      revision: HEAD
      directories:
      - path: services/*
  template:
    metadata:
      name: '{{path.basename}}'
    spec:
      project: default
      source:
        repoURL: https://github.com/example/microservices
        targetRevision: HEAD
        path: '{{path}}'
      destination:
        name: my-cluster
        namespace: '{{path.basename}}'
      syncPolicy:
        automated:
          prune: true
          selfHeal: true
        syncOptions:
        - CreateNamespace=true
```

Pour plus de détails sur les générateurs Git et la configuration basée sur des fichiers, consultez Git Generator dans la documentation du CD Argo.

Générateur de matrices

Combinez plusieurs générateurs pour les déployer sur plusieurs dimensions (environnements × clusters) :

```
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: multi-env-multi-cluster
  namespace: argocd
spec:
  generators:
  - matrix:
```

```
      generators:
      - list:
          elements:
          - environment: dev
          - environment: staging
          - environment: prod
      - clusters:
          selector:
            matchLabels:
              region: us-west-2
  template:
    metadata:
      name: 'app-{{environment}}-{{name}}'
    spec:
      project: default
      source:
        repoURL: https://github.com/example/app
        targetRevision: HEAD
        path: 'overlays/{{environment}}'
      destination:
        name: '{{name}}'
        namespace: 'app-{{environment}}'
```

Pour plus de détails sur la combinaison de générateurs, voir Matrix Generator dans la documentation d'Argo CD.

Déploiement multirégional

Déployez sur des clusters répartis dans plusieurs régions :

```
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: global-app
  namespace: argocd
spec:
  generators:
  - list:
      elements:
      - clusterName: prod-us-west
        region: us-west-2
      - clusterName: prod-us-east
        region: us-east-1
```

```
      - clusterName: prod-eu-west
        region: eu-west-1
  template:
    metadata:
      name: 'app-{{region}}'
    spec:
      project: default
      source:
        repoURL: https://github.com/example/app
        targetRevision: HEAD
        path: kubernetes
        helm:
          parameters:
          - name: region
            value: '{{region}}'
      destination:
        name: '{{clusterName}}'
        namespace: app
      syncPolicy:
        automated:
          prune: true
          selfHeal: true
```

Gérez ApplicationSets

Afficher ApplicationSets et générer les applications :

```
kubectl get applicationsets -n argocd
kubectl get applications -n argocd -l argocd.argoproj.io/application-set-
name=<applicationset-name>
```

Mettez à jour et ApplicationSet :

Modifiez les ApplicationSet spécifications et réappliquez. Argo CD met automatiquement à jour toutes les applications générées :

```
kubectl apply -f applicationset.yaml
```

Supprimer un ApplicationSet :

```
kubectl delete applicationset <name> -n argocd
```

> **Warning**
>
> La suppression et la ApplicationSet suppression de toutes les applications générées. Si ces applications l'ont fait`prune: true`, leurs ressources seront également supprimées des clusters cibles.
> Pour préserver les ressources déployées lors de la suppression d'un ApplicationSet, définissez `.syncPolicy.preserveResourcesOnDeletion` sur `true` dans la ApplicationSet spécification. Pour plus d'informations, consultez la section Élagage des applications et suppression des ressources dans la documentation d'Argo CD.

> **Important**
>
> La ApplicationSets fonctionnalité d'Argo CD comporte des considérations de sécurité que vous devez connaître avant de l'utiliser ApplicationSets. Pour plus d'informations, consultez ApplicationSet la section Sécurité dans la documentation d'Argo CD.

Ressources supplémentaires

- the section called “Projets”- Organisez ApplicationSets avec des projets
- the section called “Création d'applications”- Comprendre la configuration des applications
- ApplicationSet Documentation - Référence et modèles complets du générateur
- Référence du générateur - Spécifications détaillées du générateur

Considérations relatives à Argo CD

Cette rubrique aborde les considérations importantes relatives à l'utilisation de la fonctionnalité EKS pour Argo CD, notamment la planification, les autorisations, l'authentification et les modèles de déploiement multi-clusters.

Planification

Avant de déployer Argo CD, tenez compte des points suivants :

Stratégie de dépôt : déterminez où les manifestes de votre application seront stockés (CodeCommit, GitHub, GitLab, Bitbucket). Planifiez la structure de votre référentiel et votre stratégie de branchement pour différents environnements.

Stratégie RBAC : planifiez les équipes ou les utilisateurs qui devraient avoir un accès administrateur, éditeur ou spectateur. Associez-les à des groupes AWS Identity Center ou à des rôles Argo CD.

Architecture multi-clusters : déterminez si vous allez gérer plusieurs clusters à partir d'une seule instance Argo CD. Envisagez d'utiliser un cluster de gestion dédié pour Argo CD.

Organisation des applications : planifiez la manière dont vous allez structurer les applications et ApplicationSets. Envisagez d'utiliser des projets pour organiser les applications par équipe ou par environnement.

Politiques de synchronisation : décidez si les applications doivent être synchronisées automatiquement ou si elles doivent être approuvées manuellement. La synchronisation automatisée est courante pour le développement, manuelle pour la production.

Permissions

Pour des informations détaillées sur les rôles de capacité IAM, les politiques de confiance et les meilleures pratiques de sécurité, consultez the section called “Capacité : rôle IAM” etthe section called “Considérations relatives aux fonctionnalités d'EKS”.

Présentation du rôle de capacité IAM

Lorsque vous créez une ressource de capacité Argo CD, vous fournissez un rôle de capacité IAM. Contrairement à ACK, Argo CD gère principalement les ressources Kubernetes, et non les ressources directement. AWS Cependant, le rôle de capacité IAM est requis pour :

- Accès aux référentiels Git privés dans CodeCommit
- Intégration à AWS Identity Center pour l'authentification
- Accès aux AWS secrets dans Secrets Manager (si configuré)
- Déploiements entre clusters vers d'autres clusters EKS

CodeCommit intégration

Si vous utilisez des CodeCommit référentiels, joignez une politique avec des autorisations de lecture :

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codecommit:GitPull"
      ],
      "Resource": "*"
    }
  ]
}
```

Important

Pour une utilisation en production, limitez le `Resource` champ à un référentiel spécifique ARNs au lieu de l'utiliser`"*"`.
Exemple :

```
"Resource": "arn:aws:codecommit:us-west-2:111122223333:my-app-repo"
```

Cela limite l'accès de la fonctionnalité Argo CD aux seuls référentiels qu'elle doit gérer.

Intégration de Secrets Manager

Si vous stockez les informations d'identification du référentiel dans Secrets Manager, joignez la politique gérée pour l'accès en lecture :

```
arn:aws:iam::aws:policy/AWSSecretsManagerClientReadOnlyAccess
```

Cette politique inclut les autorisations nécessaires : `secretsmanager:GetSecretValuesecretsmanager:DescribeSecret`, et les autorisations de déchiffrement KMS.

Configuration de base

Pour les fonctionnalités de base d'Argo CD avec les référentiels Git publics, aucune politique IAM supplémentaire n'est requise au-delà de la politique de confiance.

Authentification

AWS Intégration à Identity Center

La fonctionnalité gérée d'Argo CD s'intègre directement à AWS Identity Center (anciennement AWS SSO), ce qui vous permet d'utiliser votre fournisseur d'identité existant pour l'authentification.

Lorsque vous configurez AWS l'intégration d'Identity Center :

1. Les utilisateurs accèdent à l'interface utilisateur du CD Argo via la console EKS
2. Ils s'authentifient à l'aide AWS d'Identity Center (qui peut être fédéré auprès de votre fournisseur d'identité d'entreprise)
3. AWS Identity Center fournit des informations sur les utilisateurs et les groupes à Argo CD
4. Argo CD associe les utilisateurs et les groupes aux rôles RBAC en fonction de votre configuration
5. Les utilisateurs ne voient que les applications et les ressources auxquelles ils sont autorisés à accéder

Simplification de l'accès grâce aux ensembles d'autorisations Identity Center

AWS Identity Center propose deux chemins d'authentification distincts lorsque vous travaillez avec Argo CD :

Authentification via l'API Argo CD : Identity Center fournit une authentification SSO à l'interface utilisateur et à l'API d'Argo CD. Ceci est configuré via les mappages de rôles RBAC de la fonctionnalité Argo CD.

Accès au cluster EKS : la fonctionnalité Argo CD utilise le rôle IAM fourni par le client pour s'authentifier auprès des clusters EKS par le biais d'entrées d'accès. Ces entrées d'accès peuvent être configurées manuellement pour ajouter ou supprimer des autorisations.

Vous pouvez utiliser les ensembles d'autorisations Identity Center pour simplifier la gestion des identités en autorisant une seule identité à accéder à la fois aux clusters Argo CD et EKS. Cela réduit les frais généraux en vous obligeant à gérer une seule identité sur les deux systèmes, plutôt que de conserver des informations d'identification distinctes pour l'accès au CD Argo et l'accès au cluster.

Mappages de rôles RBAC

Argo CD possède des rôles intégrés que vous pouvez associer aux utilisateurs et aux groupes AWS d'Identity Center :

ADMINISTRATEUR : accès complet à toutes les applications et à tous les paramètres. Peut créer, mettre à jour et supprimer des applications. Peut gérer la configuration d'Argo CD.

EDITEUR : Peut créer et modifier des applications. Impossible de modifier les paramètres d'Argo CD ou de supprimer des applications.

VIEWER : accès en lecture seule aux applications. Peut consulter l'état et l'historique de la demande. Impossible d'apporter des modifications.

> **Note**
>
> Les noms de rôles distinguent les majuscules des minuscules et doivent être en majuscules (ADMIN, EDITOR, VIEWER).

> **Important**
>
> L'intégration d'EKS Capabilities à AWS Identity Center prend en charge jusqu'à 1 000 identités par fonctionnalité Argo CD. Une identité peut être un utilisateur ou un groupe.

Déploiements multiclusters

La fonctionnalité gérée d'Argo CD prend en charge les déploiements multi-clusters, ce qui vous permet de gérer des applications dans des clusters de développement, de préparation et de production à partir d'une seule instance Argo CD.

Comment fonctionne le multi-cluster

Lorsque vous enregistrez des clusters supplémentaires avec Argo CD :

1. Vous créez des secrets de cluster qui font référence aux clusters EKS cibles par ARN
2. Vous créez des applications ou ApplicationSets qui ciblent différents clusters
3. Argo CD se connecte à chaque cluster pour déployer et surveiller les ressources
4. Vous visualisez et gérez tous les clusters à partir d'une seule interface utilisateur Argo CD

Prérequis pour le multi-cluster

Avant d'enregistrer des clusters supplémentaires :

- Créez une entrée d'accès sur le cluster cible pour le rôle de fonctionnalité Argo CD
- Garantir la connectivité réseau entre la fonctionnalité Argo CD et les clusters cibles
- Vérifiez les autorisations IAM pour accéder aux clusters cibles

Enregistrer un cluster

Enregistrez les clusters à l'aide de Kubernetes Secrets dans l'espace de noms. `argocd`

Obtenez l'ARN du cluster cible. Remplacez *`region-code`* par la AWS région dans laquelle se trouve votre cluster cible et remplacez *`target-cluster`* par le nom de votre cluster cible.

```
aws eks describe-cluster \
  --region region-code \
  --name target-cluster \
  --query 'cluster.arn' \
  --output text
```

Créez un secret de cluster à l'aide de l'ARN du cluster :

```
apiVersion: v1
kind: Secret
metadata:
  name: target-cluster
  namespace: argocd
  labels:
    argocd.argoproj.io/secret-type: cluster
type: Opaque
stringData:
  name: target-cluster
  server: arn:aws:eks:us-west-2:111122223333:cluster/target-cluster
  project: default
```

> **Important**
>
> Utilisez l'ARN du cluster EKS `server` sur le terrain, et non l'URL du serveur d'API Kubernetes. La fonctionnalité gérée nécessite ARNs d'identifier les clusters cibles.

Appliquez le secret :

```
kubectl apply -f cluster-secret.yaml
```

Configurer l'entrée d'accès sur le cluster cible

Le cluster cible doit disposer d'une entrée d'accès qui accorde au rôle Argo CD l'autorisation de déployer des applications. Remplacez *region-code* par la AWS région dans laquelle se trouve votre cluster cible, remplacez *target-cluster* par le nom de votre cluster cible et remplacez l'ARN par l'ARN de votre rôle de capacité Argo CD.

```
aws eks create-access-entry \
  --region region-code \
  --cluster-name target-cluster \
  --principal-arn arn:aws:iam::[.replaceable]111122223333:role/ArgoCDCapabilityRole \
  --type STANDARD \
  --kubernetes-groups system:masters
```

> **Note**
>
> Pour une utilisation en production, pensez à utiliser des groupes Kubernetes plus restrictifs au lieu de. `system:masters`

Accès au cluster privé

La fonctionnalité gérée par Argo CD peut être déployée sur des clusters EKS entièrement privés sans nécessiter de peering VPC ou de configuration réseau spécialisée. AWS gère automatiquement la connectivité entre la fonctionnalité Argo CD et les clusters distants privés. Assurez-vous que les contrôles d'accès à votre référentiel et les politiques RBAC d'Argo CD sont correctement configurés.

Déploiements entre comptes

Pour les déploiements entre comptes, ajoutez le rôle de capacité Argo CD IAM du compte source à l'entrée d'accès EKS du cluster cible :

1. Dans le compte cible, créez une entrée d'accès sur le cluster EKS cible
2. Utilisez l'Argo CD IAM Capability Role (ARN) du compte source comme principal
3. Configurer les autorisations RBAC Kubernetes appropriées pour l'entrée d'accès
4. Enregistrez le cluster cible dans Argo CD à l'aide de son ARN de cluster EKS

Aucune création de rôle IAM ou configuration de politique de confiance supplémentaire n'est requise : les entrées d'accès EKS gèrent l'accès entre comptes.

Bonnes pratiques

Utilisez des sources déclaratives comme source de vérité : stockez tous les manifestes de vos applications dans des sources déclaratives (référentiels Git, registres Helm ou images OCI), ce qui permet le contrôle des versions, les pistes d'audit et la collaboration.

Mettez en œuvre un RBAC approprié : utilisez AWS l'intégration d'Identity Center pour contrôler qui peut accéder aux applications dans Argo CD et les gérer. Argo CD permet un contrôle d'accès précis aux ressources des applications (déploiements, modules, ConfigMaps secrets).

Utilisation ApplicationSets pour les déploiements multi-environnements : ApplicationSets à utiliser pour déployer des applications sur plusieurs clusters ou espaces de noms avec différentes configurations.

Gestion du cycle de vie

Politiques de synchronisation des applications

Contrôlez la façon dont Argo CD synchronise les applications :

Synchronisation manuelle : les applications nécessitent une approbation manuelle pour synchroniser les modifications. Recommandé pour les environnements de production.

Synchronisation automatique : les applications se synchronisent automatiquement lorsque des modifications de Git sont détectées. Commun pour les environnements de développement et de mise en scène.

Réparation automatique : annule automatiquement les modifications manuelles apportées au cluster. Garantit que l'état du cluster correspond à Git.

Élagage : Supprime automatiquement les ressources supprimées de Git. À utiliser avec prudence car cela peut supprimer des ressources.

État de santé des applications

Argo CD surveille en permanence l'état des applications :

- Sain : toutes les ressources fonctionnent comme prévu

- Progression : des ressources sont en cours de création ou de mise à jour
- Dégradé : certaines ressources ne sont pas saines
- Suspendu : l'application est suspendue
- Manquant : des ressources sont absentes du cluster

Synchroniser les fenêtres

Configurez les fenêtres de synchronisation pour contrôler le moment où les applications peuvent être synchronisées :

- Autoriser les synchronisations uniquement pendant les fenêtres de maintenance
- Bloquer les synchronisations pendant les heures ouvrables
- Programmez des synchronisations automatiques à des heures spécifiques
- Utilisez des fenêtres de synchronisation dans les situations où vous devez apporter des modifications et arrêter toute synchronisation (scénarios de rupture de vitre)

Configuration du webhook pour une synchronisation plus rapide

Par défaut, Argo CD interroge les référentiels Git toutes les 6 minutes pour détecter les modifications. Pour des déploiements plus réactifs, configurez les webhooks Git pour déclencher des synchronisations immédiates lorsque des modifications sont apportées.

Les webhooks offrent plusieurs avantages :

- Réponse de synchronisation immédiate lorsque le code est envoyé (secondes au lieu de minutes)
- Réduction des frais de sondage et amélioration des performances du système
- Utilisation plus efficace des limites de débit des API
- Meilleure expérience utilisateur grâce à des commentaires plus rapides

Point de terminaison Webhook

L'URL du webhook suit le modèle`${serverUrl}/api/webhook`, où se `serverUrl` trouve l'URL de votre serveur Argo CD.

Par exemple, si l'URL de votre serveur Argo CD est`https://abc123.eks-capabilities.us-west-2.amazonaws.com`, l'URL du webhook est :

```
https://abc123.eks-capabilities.us-west-2.amazonaws.com/api/webhook
```

Configurer les webhooks par le fournisseur Git

GitHub: Dans les paramètres de votre dépôt, ajoutez un webhook avec l'URL du webhook Argo CD. Définissez le type de contenu sur `application/json` et sélectionnez « Juste l'événement push ».

GitLab: Dans les paramètres de votre projet, ajoutez un webhook avec l'URL du webhook Argo CD. Activez les « événements push » et éventuellement « Tag les événements push ».

Bitbucket : Dans les paramètres de votre dépôt, ajoutez un webhook avec l'URL du webhook Argo CD. Sélectionnez « Repository push » comme déclencheur.

CodeCommit: Créez une EventBridge règle Amazon qui déclenche les changements d'état CodeCommit du référentiel et envoie des notifications au point de terminaison du webhook Argo CD.

Pour obtenir des instructions détaillées sur la configuration du webhook, voir Configuration du webhook sur Argo CD.

> **Note**
>
> Les webhooks complètent les sondages, ils ne les remplacent pas. Argo CD continue d'interroger les référentiels comme mécanisme de secours au cas où les notifications du webhook seraient manquées.

Étapes suivantes

- the section called “Travailler avec Argo CD”- Apprenez à créer et à gérer des applications Argo CD
- the section called “Résolution des problèmes”- Résoudre les problèmes liés à Argo CD
- the section called “Travailler avec les capacités”- Gérez vos ressources de capacités Argo CD

Résoudre les problèmes liés aux fonctionnalités d'Argo CD

Cette rubrique fournit des conseils de dépannage relatifs à la fonctionnalité EKS pour Argo CD, notamment les vérifications de l'état des fonctionnalités, les problèmes de synchronisation des applications, l'authentification des référentiels et les déploiements multiclusters.

> **Note**
>
> Les fonctionnalités EKS sont entièrement gérées et exécutées en dehors de votre cluster. Vous n'avez pas accès aux journaux du serveur Argo CD ni à l'espace de `argocd` noms. Le dépannage se concentre sur l'état des fonctionnalités, l'état des applications et la configuration.

La fonctionnalité est ACTIVE mais les applications ne se synchronisent pas

Si la fonctionnalité de votre Argo CD affiche l'`ACTIVE`état mais que les applications ne se synchronisent pas, vérifiez l'état de la fonctionnalité et l'état de l'application.

Vérifiez l'état de santé des capacités :

Vous pouvez consulter les problèmes liés à l'état et à l'état des fonctionnalités dans la console EKS ou à l'aide de la AWS CLI.

Console :

1. Ouvrez la console Amazon EKS à l'adresse https://console.aws.amazon.com/eks/ home#/clusters.
2. Sélectionnez le nom de votre cluster.
3. Sélectionnez l'onglet Observabilité.
4. Sélectionnez Surveiller le cluster.
5. Choisissez l'onglet Fonctionnalités pour afficher l'état et l'état de toutes les fonctionnalités.

AWS CLI :

```
# View capability status and health
aws eks describe-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name my-argocd

# Look for issues in the health section
```

Causes courantes :

- Référentiel non configuré : dépôt Git non ajouté à Argo CD
- Échec de l'authentification : clé SSH, jeton ou CodeCommit informations d'identification non valides
- Application non créée : aucune ressource d'application n'existe dans le cluster
- Politique de synchronisation : synchronisation manuelle requise (synchronisation automatique non activée)
- Autorisations IAM : autorisations manquantes pour CodeCommit ou Secrets Manager

Vérifiez l'état de la demande :

```
# List applications
kubectl get application -n argocd

# View sync status
kubectl get application my-app -n argocd -o jsonpath='{.status.sync.status}'

# View application health
kubectl get application my-app -n argocd -o jsonpath='{.status.health}'
```

Vérifiez les conditions de candidature :

```
# Describe application to see detailed status
kubectl describe application my-app -n argocd

# View application health
kubectl get application my-app -n argocd -o jsonpath='{.status.health}'
```

Applications bloquées dans l'état « En cours »

Si une application s'affiche `Progressing` mais n'atteint jamais les `Healthy` objectifs, vérifiez l'état des ressources et les événements de l'application.

Vérifiez l'état des ressources :

```
# View application resources
kubectl get application my-app -n argocd -o jsonpath='{.status.resources}'

# Check for unhealthy resources
kubectl describe application my-app -n argocd | grep -A 10 "Health Status"
```

Causes courantes :

- Déploiement non prêt : les pods ne démarrent pas ou les sondes de disponibilité échouent
- Dépendances des ressources : ressources attendant que d'autres ressources soient prêtes
- Erreurs d'extraction d'image : les images du conteneur ne sont pas accessibles
- Ressources insuffisantes : le cluster ne dispose pas de processeur ou de mémoire pour les pods

Vérifiez la configuration du cluster cible (pour les configurations multi-clusters) :

```
# List registered clusters
kubectl get secret -n argocd -l argocd.argoproj.io/secret-type=cluster

# View cluster secret details
kubectl get secret cluster-secret-name -n argocd -o yaml
```

Défaillances d'authentification du référentiel

Si Argo CD ne peut pas accéder à vos référentiels Git, vérifiez la configuration de l'authentification.

Pour les CodeCommit référentiels :

Vérifiez que le rôle de capacité IAM dispose d' CodeCommit autorisations :

```
# View IAM policies
aws iam list-attached-role-policies --role-name my-argocd-capability-role
aws iam list-role-policies --role-name my-argocd-capability-role

# Get specific policy details
aws iam get-role-policy --role-name my-argocd-capability-role --policy-name policy-name
```

Le rôle nécessite une `codecommit:GitPull` autorisation pour accéder aux référentiels.

Pour les référentiels Git privés :

Vérifiez que les informations d'identification du référentiel sont correctement configurées :

```
# Check repository secret exists
kubectl get secret -n argocd repo-secret-name -o yaml
```

Assurez-vous que le secret contient les informations d'authentification correctes (clé SSH, jeton ou nom d'utilisateur/mot de passe).

Pour les référentiels utilisant Secrets Manager :

```
# Verify IAM Capability Role has Secrets Manager permissions
aws iam list-attached-role-policies --role-name my-argocd-capability-role

# Test secret retrieval
aws secretsmanager get-secret-value --secret-id arn:aws:secretsmanager:region-
code:111122223333:secret:my-secret
```

Problèmes de déploiement multicluster

Si les applications ne sont pas déployées sur des clusters distants, vérifiez l'enregistrement du cluster et la configuration des accès.

Vérifiez l'enregistrement du cluster :

```
# List registered clusters
kubectl get secret -n argocd -l argocd.argoproj.io/secret-type=cluster

# Verify cluster secret format
kubectl get secret CLUSTER_SECRET_NAME -n argocd -o yaml
```

Assurez-vous que le `server` champ contient l'ARN du cluster EKS, et non l'URL de l'API Kubernetes.

Vérifiez l'entrée d'accès au cluster cible :

Sur le cluster cible, vérifiez que le rôle Argo CD Capability possède une entrée d'accès :

```
# List access entries (run on target cluster or use AWS CLI)
aws eks list-access-entries --cluster-name target-cluster

# Describe specific access entry
aws eks describe-access-entry \
  --cluster-name target-cluster \
  --principal-arn arn:aws:iam::[.replaceable]111122223333:role/my-argocd-capability-
role
```

Vérifiez les autorisations IAM pour les comptes multiples :

Pour les déploiements entre comptes, vérifiez que le rôle Argo CD Capability possède une entrée d'accès sur le cluster cible. La fonctionnalité gérée utilise les entrées d'accès EKS pour l'accès entre comptes, et non l'hypothèse du rôle IAM.

Pour en savoir plus sur la configuration multi-clusters, consultezthe section called “Enregistrer des clusters”.

Étapes suivantes

- the section called “Considérations relatives à Argo CD”- Considérations et meilleures pratiques relatives à Argo CD
- the section called “Travailler avec Argo CD”- Création et gestion des applications Argo CD
- the section called “Enregistrer des clusters”- Configurer des déploiements multi-clusters
- the section called “Fonctionnalités de résolution des problèmes”- Conseils généraux de résolution des problèmes liés aux fonctionnalités

Comparaison de la fonctionnalité EKS pour Argo CD à celle d'Argo CD autogéré

La fonctionnalité EKS pour Argo CD fournit une expérience Argo CD entièrement gérée qui s'exécute dans EKS. Pour une comparaison générale des fonctionnalités EKS par rapport aux solutions autogérées, voirthe section called “Considérations”. Cette rubrique se concentre sur les différences spécifiques à Argo CD, notamment l'authentification, la gestion multi-clusters et la prise en charge des fonctionnalités en amont.

Différences par rapport à l'Argo CD en amont

La fonctionnalité EKS pour Argo CD est basée sur le CD Argo en amont, mais elle diffère en termes d'accès, de configuration et d'intégration aux AWS services.

RBAC et authentification : cette fonctionnalité inclut trois rôles RBAC (administrateur, éditeur, lecteur) et utilise AWS Identity Center pour l'authentification au lieu de l'authentification intégrée d'Argo CD. Configurez les mappages de rôles via le `rbacRoleMapping` paramètre de la fonctionnalité pour

mapper les groupes Identity Center aux rôles Argo CD, et non via Argo CD. `argocd-rbac-cm` ConfigMap L'interface utilisateur du CD Argo est hébergée avec sa propre URL directe (trouvez-la dans la console EKS sous l'onglet Capabilities de votre cluster), et l'accès à l'API utilise AWS l'authentification et l'autorisation via IAM.

Configuration du cluster : cette fonctionnalité ne configure pas automatiquement le cluster local ou les hub-and-spoke topologies. Vous configurez vos clusters cibles de déploiement et vos entrées d'accès EKS. Cette fonctionnalité prend uniquement en charge les clusters Amazon EKS en tant que cibles de déploiement utilisant le cluster EKS ARNs (et non le serveur d'API Kubernetes). URLs La fonctionnalité n'ajoute pas automatiquement le cluster local (`kubernetes.default.svc`) en tant que cible de déploiement. Pour effectuer le déploiement sur le même cluster où la fonctionnalité a été créée, enregistrez explicitement ce cluster à l'aide de son ARN.

Accès aux clusters distants simplifié : cette fonctionnalité simplifie les déploiements multi-clusters en utilisant les entrées d'accès EKS pour accorder un accès Argo CD aux clusters distants, éliminant ainsi le besoin de configurer des rôles IAM pour les comptes de service (IRSA) ou de définir des hypothèses de rôles IAM entre comptes. Cette fonctionnalité fournit également un accès transparent à des clusters EKS entièrement privés sans nécessiter de peering VPC ou de configuration réseau spécialisée. Elle AWS gère automatiquement la connectivité entre la fonctionnalité Argo CD et les clusters distants privés.

Intégration directe des AWS services : la fonctionnalité fournit une intégration directe aux AWS services par le biais des autorisations IAM du rôle de capacité. Vous pouvez CodeCommit référencer des référentiels, des diagrammes ECR Helm et CodeConnections directement dans les ressources de l'application sans créer de configurations de référentiel. Cela simplifie l'authentification et élimine le besoin de gérer des informations d'identification distinctes pour les AWS services. Consultez the section called “Configuration des référentiels” pour plus de détails.

Support des espaces de noms : cette fonctionnalité vous oblige à spécifier un espace de noms unique dans lequel les ressources Argo CD ApplicationSet, application et AppProject personnalisées doivent être créées.

> **Note**
>
> Cette restriction d'espace de noms s'applique uniquement aux ressources personnalisées d'Argo CD (Application, ApplicationSet, AppProject). Les charges de travail de votre application peuvent être déployées dans n'importe quel espace de noms de n'importe quel cluster cible. Par exemple, si vous créez la fonctionnalité avec un espace de nom`argocd`, toutes les applications CRs doivent être créées dans l'`argocd`espace de noms, mais ces

applications peuvent déployer des charges de travail versdefault, productionstaging, ou tout autre espace de noms.

> **Note**
>
> La fonctionnalité gérée comporte des exigences spécifiques en matière d'utilisation et de AppProject configuration de la CLI :
>
> - Lorsque vous utilisez l'Argo CD CLI, spécifiez les applications avec le préfixe d'espace de noms : `argocd app sync namespace/appname`
> - AppProject les ressources doivent spécifier `.spec.sourceNamespaces` les espaces de noms que le projet peut surveiller pour les applications (généralement définis sur l'espace de noms que vous avez spécifié lors de la création de la fonctionnalité)
> - Les annotations de suivi des ressources utilisent le format suivant : `namespace_appname:group/kind:namespace/name`

Fonctionnalités non prises en charge : les fonctionnalités suivantes ne sont pas disponibles dans la fonctionnalité gérée :

- Plugins de gestion de configuration (CMPs) pour la génération de manifestes personnalisés
- Scripts Lua personnalisés pour l'évaluation de l'état des ressources (les contrôles de santé intégrés pour les ressources standard sont pris en charge)
- Le contrôleur des notifications
- Fournisseurs SSO personnalisés (seul AWS Identity Center est pris en charge, y compris l'identité fédérée tierce via AWS Identity Center)
- Extensions d'interface utilisateur et bannières personnalisées
- Accès direct à `argocd-cmargocd-params`, et autres configurations ConfigMaps
- Modification du délai de synchronisation (fixé à 120 secondes)

Compatibilité : les applications ApplicationSets fonctionnent de la même manière qu'Argo CD en amont, sans aucune modification de vos manifestes. La fonctionnalité utilise les mêmes Kubernetes APIs et CRDs les outils `kubectl` fonctionnent donc de la même manière. Cette fonctionnalité prend entièrement en charge les applications et ApplicationSets les GitOps flux de travail avec

synchronisation automatique, les déploiements multiclusters, les politiques de synchronisation (automatisées, élagage, auto-guérison), les vagues de synchronisation et les hooks, l'évaluation de l'état des ressources Kubernetes standard, les fonctionnalités de restauration, les sources de référentiel Git (HTTPS et SSH), Helm, Kustomize et les manifestes YAML simples, les informations d'identification des GitHub applications, les projets de mutualisation, les exclusions et inclusions de ressources.

Utilisation de l'Argo CD CLI avec la fonctionnalité gérée

L'Argo CD CLI fonctionne de la même manière que l'Argo CD en amont pour la plupart des opérations, mais l'authentification et l'enregistrement du cluster diffèrent.

Conditions préalables

Installez l'Argo CD CLI en suivant les instructions d'installation en amont.

Configuration

Configurez la CLI à l'aide de variables d'environnement :

1. Obtenez l'URL du serveur Argo CD depuis la console EKS (sous l'onglet Capabilities de votre cluster) ou à l'aide de la AWS CLI. Le `https://` préfixe doit être supprimé :

```
export ARGOCD_SERVER=$(aws eks describe-capability \
  --cluster-name my-cluster \
  --capability-name my-argocd \
  --query 'capability.configuration.argoCd.serverUrl' \
  --output text \
  --region region-code | sed 's|^https://||')
```

2. Générez un jeton de compte depuis l'interface utilisateur d'Argo CD (Paramètres → Comptes → admin → Générer un nouveau jeton), puis définissez-le comme variable d'environnement :

```
export ARGOCD_AUTH_TOKEN="your-token-here"
```

> **Important**
>
> Cette configuration utilise le jeton de compte administrateur pour la configuration initiale et les flux de travail de développement. Pour les cas d'utilisation en production, utilisez des rôles

et des jetons adaptés au projet afin de respecter le principe du moindre privilège. Pour plus d'informations sur la configuration des rôles de projet et du RBAC, consultezthe section called "Configurer les autorisations ".

1. Définissez l'option gRPC requise :

```
export ARGOCD_OPTS="--grpc-web"
```

Une fois ces variables d'environnement définies, vous pouvez utiliser l'Argo CD CLI sans la `argocd login` commande.

Principales différences

La fonctionnalité gérée présente les limites de la CLI suivantes :

- `argocd admin`les commandes ne sont pas prises en charge (elles nécessitent un accès direct au pod)
- `argocd login`n'est pas pris en charge (utilisez plutôt des jetons de compte ou de projet)
- `argocd cluster add`nécessite l'`--aws-cluster-name`indicateur avec l'ARN du cluster EKS

Exemple : enregistrer un cluster

Enregistrez un cluster EKS pour le déploiement d'applications :

```
# Get the cluster ARN
CLUSTER_ARN=$(aws eks describe-cluster \
  --name my-cluster \
  --query 'cluster.arn' \
  --output text)

# Register the cluster
argocd cluster add $CLUSTER_ARN \
  --aws-cluster-name $CLUSTER_ARN \
  --name in-cluster \
  --project default
```

Pour la documentation complète de la CLI Argo CD, consultez la référence de la CLI Argo CD.

Chemin de migration

Vous pouvez passer d'un Argo CD autogéré à la fonctionnalité gérée :

1. Vérifiez la configuration actuelle de votre Argo CD pour détecter les fonctionnalités non prises en charge (contrôleur de notifications, bilans de santé personnalisés CMPs, extensions d'interface utilisateur)
2. Faites évoluer vos contrôleurs de CD Argo autogérés jusqu'à ce qu'il n'y ait plus de répliques pour éviter les conflits
3. Créez une ressource de capacité Argo CD sur votre cluster
4. Exportez vos applications existantes ApplicationSets, et AppProjects
5. Migrer les informations d'identification du référentiel, les secrets du cluster et les modèles d'informations d'identification du référentiel (records)
6. Si vous utilisez des clés GPG, des certificats TLS ou des hôtes connus SSH, migrez également ces configurations
7. Mettre à jour `destination.server` les champs pour utiliser les noms de clusters ou les clusters EKS ARNs
8. Appliquez-les à l'instance Argo CD gérée
9. Vérifiez que les applications se synchronisent correctement
10. Mettez hors service votre installation autogérée d'Argo CD

La fonctionnalité gérée utilise le même CD Argo APIs et les mêmes définitions de ressources, de sorte que vos manifestes existants fonctionnent avec un minimum de modifications.

Étapes suivantes

- the section called “Création d’une fonctionnalité Argo CD”- Création d'une ressource de capacité Argo CD
- the section called “Travailler avec Argo CD”- Déployez votre première application
- the section called “Considérations relatives à Argo CD”- Configurer AWS l'intégration d'Identity Center

Composition des ressources avec kro (Kube Resource Orchestrator)

kro (Kube Resource Orchestrator) est un projet open source natif de Kubernetes qui vous permet de définir des Kubernetes personnalisés à l'aide d'une configuration simple et directe. APIs Avec kro, vous pouvez facilement configurer de nouvelles personnalisations APIs qui créent un groupe d'objets Kubernetes et les opérations logiques entre eux.

Grâce aux fonctionnalités d'EKS, Kro est entièrement géré par AWS, ce qui élimine le besoin d'installer, de maintenir et de dimensionner les contrôleurs Kro sur vos clusters.

Comment fonctionne Kro

kro introduit une définition de ressource personnalisée (CRD) appelée `ResourceGraphDefinition` (RGD) qui permet de créer des Kubernetes personnalisés de manière simple et rationalisée. APIs Lorsque vous créez un`ResourceGraphDefinition`, kro utilise des extensions Kubernetes natives pour créer et gérer de nouvelles APIs extensions dans votre cluster. À partir de cette spécification de ressource unique, kro créera et enregistrera un nouveau CRD pour vous en fonction de votre spécification et s'adaptera pour gérer vos ressources personnalisées nouvellement définies.

RGDs peut inclure plusieurs ressources, et kro déterminera les interdépendances et l'ordre des ressources, pour que vous n'ayez pas à le faire. Vous pouvez utiliser une syntaxe simple pour injecter de la configuration d'une ressource à l'autre, ce qui simplifie considérablement les compositions et élimine le besoin d'opérateurs de « colle » dans votre cluster. Avec kro, vos ressources personnalisées peuvent inclure des ressources Kubernetes natives ainsi que toutes les définitions de ressources personnalisées (CRDs) installées dans le cluster.

kro prend en charge un seul type de ressource principale :

- ResourceGraphDefinition (RGD) : définit une ressource personnalisée Kubernetes, encapsulant une ou plusieurs ressources Kubernetes natives ou personnalisées sous-jacentes

En plus de cette ressource, kro créera et gérera le cycle de vie de vos ressources personnalisées créées avec elle, ainsi que de toutes les ressources qui les constituent.

kro s'intègre parfaitement à AWS Controllers for Kubernetes (ACK), ce qui vous permet de composer des ressources de charge de travail avec des ressources pour créer des abstractions de AWS haut

niveau. Cela vous permet de créer vos propres éléments constitutifs du cloud, de simplifier la gestion des ressources et d'activer des modèles réutilisables avec des paramètres de configuration par défaut et immuables basés sur les normes de votre organisation.

Les avantages de Kro

kro permet aux équipes de plateforme de créer des Kubernetes personnalisés APIs qui composent plusieurs ressources en abstractions de niveau supérieur. Cela simplifie la gestion des ressources en permettant aux développeurs de déployer des applications complexes à l'aide de ressources personnalisées simples, standardisées et versionnées. Vous définissez des modèles réutilisables pour les combinaisons de ressources communes, ce qui permet une création de ressources cohérente au sein de votre organisation.

kro utilise le langage d'expression commun (CEL) dans Kubernetes pour transmettre des valeurs entre les ressources et intégrer une logique conditionnelle, offrant ainsi une flexibilité dans la composition des ressources. Vous pouvez composer à la fois les ressources Kubernetes et les AWS ressources gérées par ACK de manière personnalisée unifiée APIs, ce qui permet de définir des applications et des infrastructures complètes.

kro prend en charge la configuration déclarative via les manifestes Kubernetes, ce qui permet aux GitOps flux de travail et à l'infrastructure de devenir des pratiques de code qui s'intègrent parfaitement à vos processus de développement existants. Dans le cadre des fonctionnalités gérées par EKS, kro est entièrement géré par AWS, ce qui élimine le besoin d'installer, de configurer et de maintenir des contrôleurs kro sur vos clusters.

Exemple : création d'un ResourceGraphDefinition

L'exemple suivant montre une application simple `ResourceGraphDefinition` qui crée une application Web avec un déploiement et un service :

```
apiVersion: kro.run/v1alpha1
kind: ResourceGraphDefinition
metadata:
  name: web-application
spec:
  schema:
    apiVersion: v1alpha1
    kind: WebApplication
    spec:
      name: string
```

```
      replicas: integer | default=3
  resources:
    - id: deployment
      template:
        apiVersion: apps/v1
        kind: Deployment
        metadata:
          name: ${schema.spec.name}
        spec:
          replicas: ${schema.spec.replicas}
    - id: service
      template:
        apiVersion: v1
        kind: Service
        metadata:
          name: ${schema.spec.name}
```

Lorsque les utilisateurs créent des instances de la ressource `WebApplication` personnalisée, kro crée automatiquement les ressources de déploiement et de service correspondantes, en gérant leur cycle de vie ainsi que la ressource personnalisée.

Intégration avec d'autres fonctionnalités gérées par EKS

kro s'intègre aux autres fonctionnalités gérées par EKS.

- AWS Contrôleurs pour Kubernetes (ACK) : utilisez kro pour composer les ressources ACK en abstractions de niveau supérieur, simplifiant ainsi la gestion des ressources. AWS
- Argo CD : utilisez Argo CD pour gérer le déploiement de ressources personnalisées Kro sur plusieurs clusters, en activant des GitOps flux de travail pour les éléments constitutifs de votre plate-forme et les piles d'applications.

Débuter avec Kro

Pour commencer à utiliser la fonctionnalité EKS pour Kro :

1. Créez une ressource de capacité Kro sur votre cluster EKS via la AWS console, la AWS CLI ou votre infrastructure préférée en tant qu'outil de code.
2. Créez ResourceGraphDefinitions (RGDs) qui définit vos compositions personnalisées APIs et vos compositions de ressources.

3. Appliquez des instances de vos ressources personnalisées pour provisionner et gérer les Kubernetes et les ressources sous-jacentes. AWS

Création d'une fonctionnalité Kro

Cette rubrique explique comment créer une fonctionnalité Kro sur votre cluster Amazon EKS.

Conditions préalables

Avant de créer une fonctionnalité Kro, assurez-vous de disposer des éléments suivants :

- Un cluster Amazon EKS existant exécutant une version de Kubernetes prise en charge (toutes les versions du support standard et étendu sont prises en charge)
- Autorisations IAM suffisantes pour créer des ressources de capacités sur les clusters EKS
- (Pour CLI/ekSCTL) L'outil CLI approprié est installé et configuré

> **Note**
>
> Contrairement à ACK et Argo CD, kro n'a pas besoin d'autorisations IAM supplémentaires au-delà de la politique de confiance. kro fonctionne entièrement au sein de votre cluster et n'effectue pas d'appels d'API. AWS Cependant, vous devez tout de même fournir un rôle de capacité IAM avec la politique de confiance appropriée. Pour plus d'informations sur la configuration des autorisations RBAC de Kubernetes pour kro, consultez. the section called “Configurer les autorisations ”

Choisissez votre outil

Vous pouvez créer une fonctionnalité kro à l'aide de la AWS Management Console AWS CLI ou de eksctl :

- the section called “Console”- Utilisez la console pour une expérience guidée
- the section called “ AWS CLI”- Utilisez la AWS CLI pour la création de scripts et l'automatisation
- the section called “eksctl”- Utilisez eksctl pour une expérience native de Kubernetes

Que se passe-t-il lorsque vous créez une capacité Kro

Lorsque vous créez une fonctionnalité Kro :

1. EKS crée le service Kro Capability et le configure pour surveiller et gérer les ressources de votre cluster
2. Des définitions de ressources personnalisées (CRDs) sont installées dans votre cluster
3. Une entrée d'accès est automatiquement créée pour votre rôle de capacité IAM, `AmazonEKSKROPolicy` qui accorde les autorisations de gestion ResourceGraphDefinitions et de gestion de leurs instances (voirthe section called “Considérations relatives aux fonctionnalités d'EKS”)
4. La fonctionnalité assume le rôle de capacité IAM que vous fournissez (utilisé uniquement pour la relation de confiance)
5. kro commence à surveiller les `ResourceGraphDefinition` ressources et leurs instances
6. L'état de capacité passe de `CREATING` à `ACTIVE`

Une fois actif, vous pouvez créer ResourceGraphDefinitions pour définir des paramètres personnalisés APIs et créer des instances de ceux-ci APIs.

> **Note**
>
> L'entrée d'accès créée automatiquement inclut celle `AmazonEKSKROPolicy` qui accorde à Kro les autorisations de gestion ResourceGraphDefinitions et leurs instances. Pour permettre à kro de créer les ressources Kubernetes sous-jacentes définies dans vos fichiers ResourceGraphDefinitions (telles que les déploiements, les services ou les ressources ACK), vous devez configurer des politiques d'entrée d'accès supplémentaires. Pour en savoir plus sur les entrées d'accès et sur la façon de configurer des autorisations supplémentaires, consultez the section called “Configurer les autorisations ” etthe section called “Considérations relatives aux fonctionnalités d'EKS”.

Étapes suivantes

Après avoir créé la fonctionnalité Kro :

- the section called “concepts kro”- Comprendre les concepts Kro et la composition des ressources

- the section called "concepts kro"- En savoir plus sur SimpleSchema les expressions CEL et les modèles de composition des ressources

Création d'une fonctionnalité Kro à l'aide de la console

Cette rubrique décrit comment créer une fonctionnalité kro (Kube Resource Orchestrator) à l'aide du. AWS Management Console

Création de la fonctionnalité Kro

1. Ouvrez la console Amazon EKS à l'adresse https://console.aws.amazon.com/eks/ home#/clusters.
2. Sélectionnez le nom de votre cluster pour ouvrir la page détaillée du cluster.
3. Choisissez l'onglet Fonctionnalités.
4. Dans le volet de navigation de gauche, choisissez kro (Kube Resource Orchestrator).
5. Choisissez Create Kro Capability.
6. Pour le rôle de capacité IAM :
   - Si vous possédez déjà un rôle de capacité IAM, sélectionnez-le dans le menu déroulant
   - Si vous devez créer un rôle, choisissez Create kro role

     Cela ouvre la console IAM dans un nouvel onglet avec une politique de confiance préremplie. Le rôle ne nécessite aucune autorisation IAM supplémentaire puisque kro fonctionne entièrement au sein de votre cluster.

     Après avoir créé le rôle, retournez à la console EKS et le rôle sera automatiquement sélectionné.

     > **Note**
     >
     > Contrairement à ACK et Argo CD, kro n'a pas besoin d'autorisations IAM supplémentaires au-delà de la politique de confiance. kro fonctionne entièrement au sein de votre cluster et n'effectue pas d'appels d'API. AWS
7. Choisissez Créer.

Le processus de création de capacités commence.

Vérifiez que la fonctionnalité est active

1. Dans l'onglet Capabilities, consultez l'état des capacités Kro.
2. Attendez que le statut passe de `CREATING` à`ACTIVE`.
3. Une fois active, la fonctionnalité est prête à être utilisée.

Pour plus d'informations sur l'état des fonctionnalités et le dépannage, consultezthe section called "Travailler avec les capacités".

Accorder des autorisations pour gérer les ressources Kubernetes

Lorsque vous créez une fonctionnalité kro, une entrée d'accès EKS est automatiquement créée avec le`AmazonEKSKROPolicy`, ce qui permet à kro de gérer ses ResourceGraphDefinitions instances. Cependant, aucune autorisation n'est accordée par défaut pour créer les ressources Kubernetes sous-jacentes (telles que les déploiements, les services ConfigMaps, etc.) définies dans votre. ResourceGraphDefinitions

Cette conception intentionnelle suit le principe du moindre privilège : différents nécessitent des autorisations différentes ResourceGraphDefinitions . Vous devez configurer explicitement les autorisations dont Kro a besoin en fonction des ressources que vous ResourceGraphDefinitions allez gérer.

Pour une mise en route rapide, des environnements de test ou de développement, utilisez `AmazonEKSClusterAdminPolicy` :

1. Dans la console EKS, accédez à l'onglet Accès de votre cluster.
2. Sous Entrées d'accès, recherchez l'entrée correspondant à votre rôle de capacité Kro (il comportera l'ARN du rôle que vous avez créé précédemment).
3. Choisissez l'entrée d'accès pour en ouvrir les détails.
4. Dans la section Politiques d'accès, choisissez Politique d'accès associée.
5. Sélectionnez `AmazonEKSClusterAdminPolicy` dans la liste des politiques.
6. Pour Etendue d'accès, sélectionnez Cluster.
7. Choisissez Associer.

> **Important**
>
> Il `AmazonEKSClusterAdminPolicy` accorde des autorisations étendues pour créer et gérer toutes les ressources Kubernetes, y compris la possibilité de créer n'importe quel type de ressource dans tous les espaces de noms. C'est pratique pour le développement POCs mais ne doit pas être utilisé en production. Pour la production, créez des politiques RBAC personnalisées qui n'accordent que les autorisations nécessaires pour les ressources spécifiques que vous ResourceGraphDefinitions allez gérer. Pour obtenir des conseils sur la configuration des autorisations de moindre privilège, reportez-vous the section called "Configurer les autorisations " aux sections et. the section called "Considérations relatives aux fonctionnalités d'EKS"

Vérifiez que les ressources personnalisées sont disponibles

Une fois la fonctionnalité activée, vérifiez que les ressources personnalisées kro sont disponibles dans votre cluster.

Utilisation de la console

1. Accédez à votre cluster dans la console Amazon EKS
2. Choisissez l'onglet Ressources
3. Choisissez les extensions
4. Choisissez CustomResourceDefinitions

Vous devriez voir le type de `ResourceGraphDefinition` ressource répertorié.

Utilisation de kubectl

```
kubectl api-resources | grep kro.run
```

Vous devriez voir le type de `ResourceGraphDefinition` ressource répertorié.

Étapes suivantes

- the section called "concepts kro"- Comprendre les concepts Kro et la composition des ressources
- the section called "concepts kro"- En savoir plus sur SimpleSchema les expressions CEL et les modèles de composition

- the section called "Travailler avec les capacités"- Gérez votre ressource Kro Capability

Création d'une fonctionnalité Kro à l'aide de la CLI AWS

Cette rubrique décrit comment créer une fonctionnalité kro (Kube Resource Orchestrator) à l'aide de la CLI. AWS

Conditions préalables

- AWS CLI : version `2.12.3` ou ultérieure. Pour vérifier votre version, lancez`aws --version`. Pour plus d'informations, consultez la section Installation dans le guide de l'utilisateur de l'interface de ligne de AWS commande.
- **`kubectl`** : outil de ligne de commande pour travailler avec des clusters Kubernetes. Pour de plus amples informations, veuillez consulter the section called "Configurer kubectl et eksctl".

Étape 1 : Création d'un rôle de capacité IAM

Créez un fichier de politique de confiance :

```
cat > kro-trust-policy.json << 'EOF'
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "capabilities.eks.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ]
    }
  ]
}
EOF
```

Créez le rôle IAM :

```
aws iam create-role \
```

```
  --role-name KROCapabilityRole \
  --assume-role-policy-document file://kro-trust-policy.json
```

> **Note**
>
> Contrairement à ACK et Argo CD, kro ne nécessite pas d'autorisations IAM supplémentaires. kro fonctionne entièrement au sein de votre cluster et n'effectue pas d'appels d'API. AWS Le rôle est uniquement nécessaire pour établir une relation de confiance avec le service des capacités EKS.

Étape 2 : Création de la fonctionnalité Kro

Créez la ressource de capacité Kro sur votre cluster. *region-code*Remplacez-le par la AWS région où se trouve votre cluster (par exemple`us-west-2`) et *my-cluster* par le nom de votre cluster.

```
aws eks create-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name my-kro \
  --type KRO \
  --role-arn arn:aws:iam::$(aws sts get-caller-identity --query Account --output
 text):role/KROCapabilityRole \
  --delete-propagation-policy RETAIN
```

La commande revient immédiatement, mais la fonctionnalité met un certain temps à devenir active car EKS crée l'infrastructure de capacités et les composants requis. EKS installera les définitions de ressources personnalisées Kubernetes associées à cette fonctionnalité dans votre cluster lors de sa création.

> **Note**
>
> Si vous recevez un message d'erreur indiquant que le cluster n'existe pas ou que vous n'êtes pas autorisé, vérifiez :
>
> - Le nom du cluster est correct
> - Votre AWS CLI est configurée pour la bonne région
> - Vous disposez des autorisations IAM requises

Étape 3 : vérifier que la fonctionnalité est active

Attendez que la fonctionnalité soit activée. Remplacez *region-code* par la AWS région dans laquelle se trouve votre cluster et remplacez *my-cluster* par le nom de votre cluster.

```
aws eks describe-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name my-kro \
  --query 'capability.status' \
  --output text
```

La fonctionnalité est prête lorsque l'état s'afficheACTIVE.

Vous pouvez également consulter les détails complets des fonctionnalités :

```
aws eks describe-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name my-kro
```

Étape 4 : octroyer des autorisations pour gérer les ressources Kubernetes

Lorsque vous créez une fonctionnalité kro, une entrée d'accès EKS est automatiquement créée avec leAmazonEKSKROPolicy, ce qui permet à kro de gérer ses ResourceGraphDefinitions instances. Cependant, aucune autorisation n'est accordée par défaut pour créer les ressources Kubernetes sous-jacentes (telles que les déploiements, les services ConfigMaps, etc.) définies dans votre. ResourceGraphDefinitions

Cette conception intentionnelle suit le principe du moindre privilège : différents nécessitent des autorisations différentes ResourceGraphDefinitions . Par exemple :* A ResourceGraphDefinition qui crée uniquement ConfigMaps et Secrets a besoin d'autorisations différentes de celles qui créent des déploiements et des services * A ResourceGraphDefinition qui crée des ressources ACK a besoin d'autorisations pour ces ressources personnalisées spécifiques * Certains ResourceGraphDefinitions peuvent uniquement lire les ressources existantes sans en créer de nouvelles

Vous devez configurer explicitement les autorisations dont Kro a besoin en fonction des ressources que vous ResourceGraphDefinitions allez gérer.

Configuration rapide

Pour démarrer rapidement, tester ou développer des environnements, utilisez `AmazonEKSClusterAdminPolicy` :

Obtenez l'ARN du rôle de capacité :

```
CAPABILITY_ROLE_ARN=$(aws eks describe-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name my-kro \
  --query 'capability.roleArn' \
  --output text)
```

Associez la politique d'administration du cluster :

```
aws eks associate-access-policy \
  --region region-code \
  --cluster-name my-cluster \
  --principal-arn $CAPABILITY_ROLE_ARN \
  --policy-arn arn:aws:eks::aws:cluster-access-policy/AmazonEKSClusterAdminPolicy \
  --access-scope type=cluster
```

Important

Il `AmazonEKSClusterAdminPolicy` accorde des autorisations étendues pour créer et gérer toutes les ressources Kubernetes, y compris la possibilité de créer n'importe quel type de ressource dans tous les espaces de noms. C'est pratique pour le développement POCs mais ne doit pas être utilisé en production. Pour la production, créez des politiques RBAC personnalisées qui n'accordent que les autorisations nécessaires pour les ressources spécifiques que vous ResourceGraphDefinitions allez gérer. Pour obtenir des conseils sur la configuration des autorisations de moindre privilège, reportez-vous the section called “Configurer les autorisations ” aux sections et. the section called “Considérations relatives aux fonctionnalités d'EKS”

Étape 5 : vérifier que les ressources personnalisées sont disponibles

Une fois la fonctionnalité activée, vérifiez que les ressources personnalisées kro sont disponibles dans votre cluster :

```
kubectl api-resources | grep kro.run
```

Vous devriez voir le type de `ResourceGraphDefinition` ressource répertorié.

Étapes suivantes

- the section called "concepts kro"- Comprendre les concepts Kro et la composition des ressources
- the section called "concepts kro"- En savoir plus sur SimpleSchema les expressions CEL et les modèles de composition
- the section called "Travailler avec les capacités"- Gérez votre ressource Kro Capability

Création d'une capacité kro à l'aide de eksctl

Cette rubrique décrit comment créer une fonctionnalité kro (Kube Resource Orchestrator) à l'aide de eksctl.

> **Note**
>
> Les étapes suivantes nécessitent la version eksctl `0.220.0` ou une version ultérieure. Pour vérifier votre version, lancez`eksctl version`.

Étape 1 : Création d'un rôle de capacité IAM

Créez un fichier de politique de confiance :

```
cat > kro-trust-policy.json << 'EOF'
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "capabilities.eks.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ]
```

```
    }
  ]
}
EOF
```

Créez le rôle IAM :

```
aws iam create-role \
  --role-name KROCapabilityRole \
  --assume-role-policy-document file://kro-trust-policy.json
```

> **Note**
>
> Contrairement à ACK et Argo CD, kro n'a pas besoin d'autorisations IAM supplémentaires au-delà de la politique de confiance. kro fonctionne entièrement au sein de votre cluster et n'effectue pas d'appels d'API. AWS

Étape 2 : Création de la fonctionnalité Kro

Créez la capacité kro à l'aide de eksctl. Remplacez *region-code* par la AWS région dans laquelle se trouve votre cluster et remplacez *my-cluster* par le nom de votre cluster.

```
eksctl create capability \
  --region region-code \
  --cluster my-cluster \
  --name my-kro \
  --type KRO \
  --role-arn arn:aws:iam::[.replaceable]111122223333:role/KROCapabilityRole
```

La commande revient immédiatement, mais la fonctionnalité met un certain temps à devenir active.

Étape 3 : vérifier que la fonctionnalité est active

Vérifiez l'état des capacités. Remplacez *region-code* par la AWS région dans laquelle se trouve votre cluster et remplacez *my-cluster* par le nom de votre cluster.

```
eksctl get capability \
  --region region-code \
  --cluster my-cluster \
```

```
  --name my-kro
```

La fonctionnalité est prête lorsque l'état s'afficheACTIVE.

Étape 4 : accorder des autorisations pour gérer les ressources Kubernetes

Par défaut, kro peut uniquement créer ResourceGraphDefinitions et gérer leurs instances. Pour permettre à kro de créer et de gérer les ressources Kubernetes sous-jacentes définies dans votre fichier ResourceGraphDefinitions, associez la politique d'AmazonEKSClusterAdminPolicyaccès à l'entrée d'accès de la fonctionnalité.

Obtenez l'ARN du rôle de capacité :

```
CAPABILITY_ROLE_ARN=$(aws eks describe-capability \
  --region region-code \
  --cluster my-cluster \
  --name my-kro \
  --query 'capability.roleArn' \
  --output text)
```

Associez la politique d'administration du cluster :

```
aws eks associate-access-policy \
  --region region-code \
  --cluster my-cluster \
  --principal-arn $CAPABILITY_ROLE_ARN \
  --policy-arn arn:aws:eks::aws:cluster-access-policy/AmazonEKSClusterAdminPolicy \
  --access-scope type=cluster
```

> **⚠ Important**
>
> Il AmazonEKSClusterAdminPolicy accorde des autorisations étendues pour créer et gérer toutes les ressources Kubernetes et vise à rationaliser le démarrage. Pour une utilisation en production, créez des politiques RBAC plus restrictives qui n'accordent que les autorisations nécessaires pour les ressources spécifiques que vous ResourceGraphDefinitions allez gérer. Pour obtenir des conseils sur la configuration des autorisations de moindre privilège, reportez-vous the section called “Configurer les autorisations ” aux sections et. the section called “Considérations relatives aux fonctionnalités d'EKS”

Étape 5 : vérifier que les ressources personnalisées sont disponibles

Une fois la fonctionnalité activée, vérifiez que les ressources personnalisées kro sont disponibles dans votre cluster :

```
kubectl api-resources | grep kro.run
```

Vous devriez voir le type de `ResourceGraphDefinition` ressource répertorié.

Étapes suivantes

- the section called "concepts kro"- Comprendre les concepts Kro et la composition des ressources
- the section called "concepts kro"- En savoir plus sur SimpleSchema les expressions CEL et les modèles de composition
- the section called "Travailler avec les capacités"- Gérez votre ressource Kro Capability

concepts kro

kro permet aux équipes de plateforme de créer des Kubernetes personnalisés APIs qui composent plusieurs ressources en abstractions de niveau supérieur. Cette rubrique présente un exemple pratique, puis explique les concepts de base que vous devez comprendre lorsque vous travaillez avec la capacité EKS pour kro.

Débuter avec Kro

Après avoir créé la fonctionnalité Kro (voirthe section called "Créez une capacité KRO"), vous pouvez commencer à créer une APIs utilisation personnalisée ResourceGraphDefinitions dans votre cluster.

Voici un exemple complet qui crée une abstraction d'application Web simple :

```
apiVersion: kro.run/v1alpha1
kind: ResourceGraphDefinition
metadata:
  name: webapplication
spec:
  schema:
    apiVersion: v1alpha1
    kind: WebApplication
    group: kro.run
    spec:
```

```
      name: string | required=true
      image: string | default="nginx:latest"
      replicas: integer | default=3
  resources:
  - id: deployment
    template:
      apiVersion: apps/v1
      kind: Deployment
      metadata:
        name: ${schema.spec.name}
      spec:
        replicas: ${schema.spec.replicas}
        selector:
          matchLabels:
            app: ${schema.spec.name}
        template:
          metadata:
            labels:
              app: ${schema.spec.name}
          spec:
            containers:
            - name: app
              image: ${schema.spec.image}
              ports:
              - containerPort: 80
  - id: service
    template:
      apiVersion: v1
      kind: Service
      metadata:
        name: ${schema.spec.name}
      spec:
        selector:
          app: ${schema.spec.name}
        ports:
        - protocol: TCP
          port: 80
          targetPort: 80
```

Après avoir appliqué cela ResourceGraphDefinition, les équipes d'application peuvent créer des applications Web à l'aide de votre API simplifiée :

```
apiVersion: kro.run/v1alpha1
```

```
kind: WebApplication
metadata:
  name: my-app
spec:
  name: my-app
  replicas: 5
```

kro crée automatiquement le déploiement et le service avec la configuration appropriée. Comme il `image` n'est pas spécifié, il utilise la valeur par défaut `nginx:latest` du schéma.

Concepts de base

> **Important**
>
> kro valide ResourceGraphDefinitions au moment de la création, pas au moment de l'exécution. Lorsque vous créez un RGD, kro valide la syntaxe du CEL, vérifie le type des expressions par rapport aux schémas Kubernetes réels, vérifie l'existence des champs et détecte les dépendances circulaires. Cela signifie que les erreurs sont détectées immédiatement lorsque vous créez le RGD, avant que les instances ne soient déployées.

ResourceGraphDefinition

A ResourceGraphDefinition (RGD) définit une API Kubernetes personnalisée en spécifiant :

- Schéma : structure de l'API à l'aide SimpleSchema du format (noms de champs, types, valeurs par défaut, validation)
- Ressources : modèles pour les Kubernetes sous-jacents ou AWS ressources à créer
- Dépendances - Comment les ressources sont liées les unes aux autres (détectée automatiquement à partir des références aux champs)

Lorsque vous appliquez un RGD, kro enregistre une nouvelle définition de ressource personnalisée (CRD) dans votre cluster. Les équipes d'application peuvent ensuite créer des instances de votre API personnalisée, et kro se charge de créer et de gérer toutes les ressources sous-jacentes.

Pour plus d'informations, consultez la section ResourceGraphDefinition Présentation dans la documentation de kro.

SimpleSchema format

SimpleSchema fournit un moyen simplifié de définir des schémas d'API sans avoir besoin de connaissances d'OpenAPI :

```
schema:
  apiVersion: v1alpha1
  kind: Database
  spec:
    name: string | required=true description="Database name"
    size: string | default="small" enum=small,medium,large
    replicas: integer | default=1 minimum=1 maximum=5
```

SimpleSchema prend en charge `string` `integer`boolean,, et `number` types avec des contraintes telles que `required`default,maximum,minimum/enum, et`pattern`.

Pour plus d'informations, consultez SimpleSchemala documentation de kro.

Expressions CEL

kro utilise le langage d'expression commun (CEL) pour référencer les valeurs de manière dynamique et ajouter une logique conditionnelle. Les expressions CEL sont encapsulées `${ }` et peuvent être utilisées de deux manières :

Expressions autonomes - La valeur complète du champ est une expression unique :

```
spec:
  replicas: ${schema.spec.replicaCount}  # Expression returns integer
  labels: ${schema.spec.labelMap}        # Expression returns object
```

Le résultat de l'expression remplace la valeur complète du champ et doit correspondre au type attendu du champ.

Modèles de chaînes : une ou plusieurs expressions incorporées dans une chaîne :

```
metadata:
  name: "${schema.spec.prefix}-${schema.spec.name}"  # Multiple expressions
  annotation: "Created by ${schema.spec.owner}"      # Single expression in string
```

Toutes les expressions des modèles de chaînes doivent renvoyer des chaînes. `string()`À utiliser pour convertir d'autres types :`"replicas-${string(schema.spec.count)}"`.

Références de champs - Accédez aux valeurs des spécifications de l'instance à l'aide `schema.spec` de :

```
template:
  metadata:
    name: ${schema.spec.name}-deployment
    namespace: ${schema.metadata.namespace}  # Can also reference metadata
  spec:
    replicas: ${schema.spec.replicas}
```

Accès aux champs facultatif - À utiliser ? pour les champs qui n'existent peut-être pas :

```
# For ConfigMaps or Secrets with unknown structure
value: ${configmap.data.?DATABASE_URL}

# For optional status fields
ready: ${deployment.status.?readyReplicas > 0}
```

Si le champ n'existe pas, l'expression est renvoyée `null` au lieu d'échouer.

Ressources conditionnelles - N'incluez les ressources que lorsque les conditions sont remplies :

```
resources:
- id: ingress
  includeWhen:
    - ${schema.spec.enableIngress == true}
  template:
    # ... ingress configuration
```

Le `includeWhen` champ accepte une liste d'expressions booléennes. Toutes les conditions doivent être remplies pour que la ressource soit créée. Actuellement, seuls `includeWhen` les `schema.spec` champs peuvent être référencés.

Transformations - Transformez les valeurs à l'aide d'opérateurs et de fonctions ternaires :

```
template:
  spec:
    resources:
      requests:
        memory: ${schema.spec.size == "small" ? "512Mi" : "2Gi"}
```

```
    # String concatenation
    image: ${schema.spec.registry + "/" + schema.spec.imageName}

    # Type conversion
    port: ${string(schema.spec.portNumber)}
```

Références entre ressources : valeurs de référence provenant d'autres ressources :

```
resources:
- id: bucket
  template:
    apiVersion: s3.services.k8s.aws/v1alpha1
    kind: Bucket
    spec:
      name: ${schema.spec.name}-data

- id: configmap
  template:
    apiVersion: v1
    kind: ConfigMap
    data:
      BUCKET_NAME: ${bucket.spec.name}
      BUCKET_ARN: ${bucket.status.ackResourceMetadata.arn}
```

Lorsque vous référencez une autre ressource dans une expression CEL, cela crée automatiquement une dépendance. kro s'assure que la ressource référencée est créée en premier.

Pour plus d'informations, consultez la section Expressions CEL dans la documentation de kro.

Dépendances des ressources

kro déduit automatiquement les dépendances à partir des expressions CEL : vous ne spécifiez pas l'ordre, vous décrivez les relations. Lorsqu'une ressource en référence une autre à l'aide d'une expression CEL, kro crée une dépendance et détermine l'ordre de création correct.

```
resources:
- id: bucket
  template:
    apiVersion: s3.services.k8s.aws/v1alpha1
    kind: Bucket
    spec:
      name: ${schema.spec.name}-data
```

```
- id: notification
  template:
    apiVersion: s3.services.k8s.aws/v1alpha1
    kind: BucketNotification
    spec:
      bucket: ${bucket.spec.name}  # Creates dependency: notification depends on bucket
```

L'expression `${bucket.spec.name}` crée une dépendance. kro crée un graphe acyclique dirigé (DAG) de toutes les ressources et de leurs dépendances, puis calcule un ordre topologique pour la création.

Ordre de création : les ressources sont créées dans l'ordre topologique (les dépendances d'abord).

Création parallèle : les ressources sans dépendances sont créées simultanément.

Ordre de suppression : les ressources sont supprimées dans l'ordre topologique inverse (les personnes dépendantes d'abord).

Dépendances circulaires : non autorisées : kro rejette les dépendances circulaires lors de la ResourceGraphDefinitions validation.

Vous pouvez consulter l'ordre de création calculé :

```
kubectl get resourcegraphdefinition my-rgd -o jsonpath='{.status.topologicalOrder}'
```

Pour plus d'informations, consultez la section Inférence graphique dans la documentation de kro.

Composer avec ACK

kro fonctionne parfaitement avec la capacité EKS permettant à ACK de composer des AWS ressources avec des ressources Kubernetes :

```
resources:
# Create {aws} S3 bucket with ACK
- id: bucket
  template:
    apiVersion: s3.services.k8s.aws/v1alpha1
    kind: Bucket
    spec:
      name: ${schema.spec.name}-files

# Inject bucket details into Kubernetes ConfigMap
```

```
- id: config
  template:
    apiVersion: v1
    kind: ConfigMap
    data:
      BUCKET_NAME: ${bucket.spec.name}
      BUCKET_ARN: ${bucket.status.ackResourceMetadata.arn}

# Use ConfigMap in application deployment
- id: deployment
  template:
    apiVersion: apps/v1
    kind: Deployment
    spec:
      template:
        spec:
          containers:
          - name: app
            envFrom:
            - configMapRef:
                name: ${config.metadata.name}
```

Ce modèle vous permet de créer des AWS ressources, d'extraire leurs détails (ARNs, URLs, points de terminaison) et de les injecter dans la configuration de votre application, le tout géré comme une seule unité.

Pour plus de modèles de composition et des exemples avancés, voirthe section called "Considérations".

Étapes suivantes

- the section called "Considérations"- Découvrez les modèles spécifiques à EKS, le RBAC et l'intégration avec ACK et Argo CD
- Documentation kro - Documentation kro complète, y compris les expressions CEL avancées, les modèles de validation et le dépannage

Configurer les autorisations Kro

Contrairement à ACK et Argo CD, kro ne nécessite pas d'autorisations IAM. kro fonctionne entièrement au sein de votre cluster Kubernetes et n'effectue pas d'appels d'API. AWS Contrôlez l'accès aux ressources Kro à l'aide de la norme Kubernetes RBAC.

Comment fonctionnent les autorisations avec Kro

kro utilise deux types de ressources Kubernetes avec des portées différentes :

ResourceGraphDefinitions: ressources étendues au cluster qui définissent la personnalisation. APIs Généralement géré par les équipes de plateforme qui conçoivent et maintiennent les normes organisationnelles.

Instances : ressources personnalisées étendues à l'espace de noms créées à partir de. ResourceGraphDefinitions Peut être créé par les équipes chargées des applications disposant des autorisations RBAC appropriées.

Par défaut, la fonctionnalité Kro dispose des autorisations nécessaires pour gérer ResourceGraphDefinitions ses instances via la politique de saisie d'`AmazonEKSKROPolicy`accès. Cependant, kro a besoin d'autorisations supplémentaires pour créer et gérer les ressources Kubernetes sous-jacentes définies dans votre répertoire ResourceGraphDefinitions (telles que les déploiements, les services ou les ressources ACK). Vous devez accorder ces autorisations par le biais de politiques d'entrée d'accès ou de Kubernetes RBAC. Pour plus de détails sur l'octroi de ces autorisations, voir Autorisations de ressources arbitraires kro.

Autorisations de l'équipe de plateforme

Les équipes de la plateforme ont besoin d'autorisations pour créer et gérer ResourceGraphDefinitions.

Exemple ClusterRole pour les équipes de plateforme :

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: kro-platform-admin
rules:
- apiGroups: ["kro.run"]
  resources: ["resourcegraphdefinitions"]
  verbs: ["*"]
```

Associez les membres de l'équipe de la plateforme :

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
```

```
  name: platform-team-kro-admin
subjects:
- kind: Group
  name: platform-team
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: kro-platform-admin
  apiGroup: rbac.authorization.k8s.io
```

Autorisations de l'équipe d'application

Les équipes d'application ont besoin d'autorisations pour créer des instances de ressources personnalisées dans leurs espaces de noms.

Exemple de rôle pour les équipes de candidature :

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: kro-app-developer
  namespace: my-app
rules:
- apiGroups: ["kro.run"]
  resources: ["webapps", "databases"]
  verbs: ["create", "get", "list", "update", "delete", "patch"]
```

Associez les membres de l'équipe de candidature :

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: app-team-kro-developer
  namespace: my-app
subjects:
- kind: Group
  name: app-team
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role
  name: kro-app-developer
  apiGroup: rbac.authorization.k8s.io
```

> **Note**
>
> Le groupe d'API du rôle (`kro.run`dans cet exemple) doit correspondre à celui `apiVersion` défini dans votre ResourceGraphDefinition schéma.

Accès en lecture seule

Accordez un accès en lecture seule à la vue ResourceGraphDefinitions et aux instances sans autorisation de modification.

Lecture seule ClusterRole :

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: kro-viewer
rules:
- apiGroups: ["kro.run"]
  resources: ["resourcegraphdefinitions"]
  verbs: ["get", "list", "watch"]
```

Rôle en lecture seule pour les instances :

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: kro-instance-viewer
  namespace: my-app
rules:
- apiGroups: ["kro.run"]
  resources: ["webapps", "databases"]
  verbs: ["get", "list", "watch"]
```

Accès à plusieurs espaces de noms

Accordez aux équipes d'application l'accès à plusieurs espaces de noms à l'aide de ClusterRoles with RoleBindings.

ClusterRole pour l'accès à plusieurs espaces de noms :

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: kro-multi-namespace-developer
rules:
- apiGroups: ["kro.run"]
  resources: ["webapps"]
  verbs: ["create", "get", "list", "update", "delete"]
```

Liez à des espaces de noms spécifiques :

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: app-team-dev-access
  namespace: development
subjects:
- kind: Group
  name: app-team
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: kro-multi-namespace-developer
  apiGroup: rbac.authorization.k8s.io
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: app-team-staging-access
  namespace: staging
subjects:
- kind: Group
  name: app-team
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: kro-multi-namespace-developer
  apiGroup: rbac.authorization.k8s.io
```

Bonnes pratiques

Principe du moindre privilège : accordez uniquement les autorisations minimales nécessaires pour les responsabilités de chaque équipe.

Utilisez des groupes plutôt que des utilisateurs individuels : liez les rôles à des groupes plutôt qu'à des utilisateurs individuels pour faciliter la gestion.

Problèmes liés à la plateforme et à l'application distincts : les équipes chargées de la plateforme gèrent ResourceGraphDefinitions, les équipes chargées des applications gèrent les instances.

Isolation des espaces de noms : utilisez des espaces de noms pour isoler différentes équipes ou environnements, le RBAC contrôlant l'accès à chaque espace de noms.

Accès en lecture seule pour l'audit : fournissez un accès en lecture seule aux équipes de sécurité et de conformité à des fins d'audit.

Étapes suivantes

- the section called “concepts kro”- Comprendre les concepts Kro et la composition des ressources
- the section called “concepts kro”- Comprendre SimpleSchema les expressions CEL et les modèles de composition
- the section called “Considérations relatives aux fonctionnalités d'EKS”- Passez en revue les meilleures pratiques de sécurité en termes de fonctionnalités

considérations relatives à Kro pour EKS

Cette rubrique aborde les considérations importantes relatives à l'utilisation de la fonctionnalité EKS pour kro, notamment le moment d'utiliser la composition des ressources, les modèles RBAC et l'intégration avec d'autres fonctionnalités EKS.

Quand utiliser Kro

kro est conçu pour créer des modèles d'infrastructure réutilisables et personnalisés APIs qui simplifient la gestion complexe des ressources.

Utilisez Kro lorsque vous devez :

- Créez des plateformes en libre-service simplifiées APIs pour les équipes d'application

- Standardisez les modèles d'infrastructure entre les équipes (base de données, sauvegarde et surveillance)
- Gérez les dépendances des ressources et transmettez des valeurs entre les ressources
- Créez des abstractions personnalisées qui masquent la complexité de l'implémentation
- Composez plusieurs ressources ACK en blocs de construction de niveau supérieur
- Activez GitOps les flux de travail pour les infrastructures complexes

N'utilisez pas kro lorsque :

- Gestion de ressources simples et autonomes (utilisez directement les ressources ACK ou Kubernetes)
- Vous avez besoin d'une logique d'exécution dynamique (kro est déclaratif, pas impératif)
- Les ressources n'ont pas de dépendances ni de configuration partagée

kro excelle dans la création de « voies pavées », des modèles réutilisables et opiniâtres qui permettent aux équipes de déployer facilement et correctement des infrastructures complexes.

Modèles RBAC

kro permet de séparer les préoccupations entre les équipes de plateforme qui créent ResourceGraphDefinitions et les équipes d'application qui créent les instances.

Responsabilités de l'équipe de plateforme

Les équipes de la plateforme créent et gèrent ResourceGraphDefinitions (RGDs) qui définissent la personnalisation APIs.

Autorisations nécessaires :

- Création, mise à jour, suppression ResourceGraphDefinitions
- Gérer les types de ressources sous-jacents (déploiements, services, ressources ACK)
- Accès à tous les espaces de noms où RGDs seront utilisés

Exemple ClusterRole pour l'équipe de la plateforme :

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
```

```
metadata:
  name: platform-kro-admin
rules:
- apiGroups: ["kro.run"]
  resources: ["resourcegraphdefinitions"]
  verbs: ["*"]
- apiGroups: ["*"]
  resources: ["*"]
  verbs: ["get", "list", "watch"]
```

Pour une configuration RBAC détaillée, voirthe section called “Configurer les autorisations ”.

Responsabilités de l'équipe de candidature

Les équipes d'application créent des instances de ressources personnalisées définies par RGDs sans avoir à comprendre la complexité sous-jacente.

Autorisations nécessaires :

- Création, mise à jour, suppression d'instances de ressources personnalisées
- Accès en lecture à leur espace de noms
- Aucun accès aux ressources sous-jacentes ou RGDs

Avantages :

- Les équipes utilisent des outils simples et de haut niveau APIs
- Les équipes de plateforme contrôlent les détails de mise en œuvre
- Réduction du risque de mauvaise configuration
- Intégration plus rapide pour les nouveaux membres de l'équipe

Intégration avec d'autres fonctionnalités d'EKS

Composer des ressources ACK

kro est particulièrement puissant lorsqu'il est associé à ACK pour créer des modèles d'infrastructure.

Schémas courants :

- Application avec stockage : compartiment S3 + file d'attente SQS + configuration des notifications

- Pile de base de données : instance RDS + groupe de paramètres + groupe de sécurité + secret Secrets Manager
- Mise en réseau : VPC + sous-réseaux + tables de routage + groupes de sécurité + passerelles NAT
- Calcul avec stockage : EC2 instance + volumes EBS + profil d'instance IAM

kro gère l'ordre des dépendances, transmet les valeurs entre les ressources (chaînes similaires ARNs et chaînes de connexion) et gère le cycle de vie complet en tant qu'unité unique.

Pour des exemples de composition de ressources ACK, voirthe section called “Concepts d'ACK”.

GitOps avec Argo CD

Utilisez le CD EKS Capability for Argo pour déployer à la fois des instances RGDs et des instances à partir de référentiels Git.

Organisation du référentiel :

- Dépôt de plateforme : contenus ResourceGraphDefinitions gérés par l'équipe de la plateforme
- Dépôts d'applications : contiennent des instances de ressources personnalisées gérées par les équipes chargées des applications
- Dépôt partagé : contient à la fois des instances RGDs et des instances pour les petites organisations

Considérations :

- Déployez RGDs avant les instances (les ondes de synchronisation Argo CD peuvent vous aider)
- Utilisez des projets Argo CD distincts pour les équipes chargées des plateformes et des applications
- L'équipe de la plateforme contrôle l'accès au référentiel RGD
- Les équipes chargées des applications ont un accès en lecture seule aux définitions RGD

Pour en savoir plus sur Argo CD, voirthe section called “Travailler avec Argo CD”.

Organiser ResourceGraphDefinitions

Organisez RGDs en fonction de l'objectif, de la complexité et de la propriété.

Par objectif :

- Infrastructure : piles de bases de données, mise en réseau, stockage
- Application : applications Web APIs, tâches par lots
- Plateforme : services partagés, surveillance, journalisation

Par complexité :

- Simple : 2 à 3 ressources avec un minimum de dépendances
- Modéré : 5 à 10 ressources avec certaines dépendances
- Complexe : plus de 10 ressources avec des dépendances complexes

Conventions de dénomination :

- Utilisez des noms descriptifs :`webapp-with-database`, `s3-notification-queue`
- Incluez la version dans le nom pour les modifications majeures : `webapp-v2`
- Utilisez des préfixes cohérents pour les éléments suivants RGDs :`platform-` , `app-`

Stratégie d'espace de noms :

- RGDs sont limités à la taille d'un cluster (et non à un espace de noms)
- Les instances sont dotées d'un espace de noms
- Utilisez des sélecteurs d'espaces de noms RGDs pour contrôler où les instances peuvent être créées

Versionnage et mises à jour

Planifiez l'évolution du RGD et la migration des instances.

Mises à jour du RGD :

- Changements permanents : mettez à jour le RGD en place (ajoutez des champs facultatifs, de nouvelles ressources avec IncludeWhen)
- Changements radicaux : créer un nouveau RGD avec un nom différent (webapp-v2)
- Obsolète : Marquez les anciens RGDs avec des annotations, communiquez le calendrier de migration

Migration d'instance :

- Créez de nouvelles instances avec un RGD mis à jour
- Validez que les nouvelles instances fonctionnent correctement
- Supprimer les anciennes instances
- kro gère automatiquement les mises à jour des ressources sous-jacentes

Bonnes pratiques :

- Testez d'abord les modifications du RGD dans des environnements hors production
- Utiliser le versionnement sémantique dans les noms RGD pour les modifications majeures
- Documenter les modifications majeures et les voies de migration
- Fournir des exemples de migration pour les équipes chargées des applications

Validation et tests

Validez RGDs avant le déploiement en production.

Stratégies de validation :

- Validation du schéma : kro valide automatiquement la structure RGD
- Instances à sec : créez des instances de test dans les espaces de noms de développement
- Tests d'intégration : vérifier que les ressources composées fonctionnent ensemble
- Application des politiques : utilisez des contrôleurs d'admission pour faire appliquer les normes organisationnelles

Problèmes courants à tester :

- Dépendances des ressources et classement
- Transfert de valeur entre les ressources (expressions CEL)
- Inclusion conditionnelle de ressources (IncludeWhen)
- Propagation du statut à partir des ressources sous-jacentes
- Autorisations RBAC pour la création d'instances

Documentation en amont

Pour des informations détaillées sur l'utilisation de kro :

- Débuter avec kro - Création ResourceGraphDefinitions
- Expressions CEL - Écrire des expressions CEL
- Kro Guides - Modèles de composition avancés
- Dépannage - Dépannage et débogage

Étapes suivantes

- the section called "Configurer les autorisations "- Configurer le RBAC pour les équipes chargées des plateformes et des applications
- the section called "concepts kro"- Comprendre les concepts Kro et le cycle de vie des ressources
- the section called "Résolution des problèmes"- Résoudre les problèmes liés à KRO
- the section called "Concepts d'ACK"- En savoir plus sur les ressources d'ACK pour la composition
- the section called "Travailler avec Argo CD"- Déploiement RGDs et instances avec GitOps

Résoudre les problèmes liés aux fonctionnalités Kro

Cette rubrique fournit des conseils de dépannage relatifs à la fonctionnalité EKS pour kro, notamment les vérifications de l'état des fonctionnalités, les autorisations RBAC, les erreurs d'expression CEL et les problèmes de composition des ressources.

> **Note**
>
> Les fonctionnalités EKS sont entièrement gérées et exécutées en dehors de votre cluster. Vous n'avez pas accès aux journaux du contrôleur ni à l'espace de `kro-system` noms. Le dépannage se concentre sur l'état des capacités, la configuration RBAC et l'état des ressources.

La fonctionnalité est ACTIVE mais ResourceGraphDefinitions ne fonctionne pas

Si votre capacité Kro affiche un `ACTIVE` état mais ResourceGraphDefinitions ne crée pas de ressources sous-jacentes, vérifiez l'état de la fonctionnalité, les autorisations RBAC et l'état des ressources.

Vérifiez l'état de santé des capacités :

Vous pouvez consulter les problèmes liés à l'état et à l'état des fonctionnalités dans la console EKS ou à l'aide de la AWS CLI.

Console :

1. Ouvrez la console Amazon EKS à l'adresse https://console.aws.amazon.com/eks/ home#/clusters.
2. Sélectionnez le nom de votre cluster.
3. Sélectionnez l'onglet Observabilité.
4. Sélectionnez Surveiller le cluster.
5. Choisissez l'onglet Fonctionnalités pour afficher l'état et l'état de toutes les fonctionnalités.

AWS CLI :

```
# View capability status and health
aws eks describe-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name my-kro

# Look for issues in the health section
```

Causes courantes :

- Permissions RBAC manquantes : kro n'a pas les autorisations nécessaires pour créer des ressources Kubernetes sous-jacentes
- Expressions CEL non valides : erreurs de syntaxe dans ResourceGraphDefinition
- Dépendances des ressources : les ressources dépendantes ne sont pas prêtes
- Validation du schéma : l'instance ne répond pas aux exigences du schéma RGD

Vérifiez les autorisations RBAC :

```
# Check if capability has cluster admin policy
kubectl get accessentry -A | grep kro
```

Si la fonctionnalité ne dispose pas des autorisations requises, associez-la à l'`AmazonEKSClusterAdminPolicy`entrée d'accès de la fonctionnalité kro ou créez des politiques RBAC plus restrictives pour une utilisation en production. Consultez the section called “Configurer les autorisations ” pour plus de détails.

Vérifier ResourceGraphDefinition le statut :

```
# List all RGDs
kubectl get resourcegraphdefinition

# Describe specific RGD
kubectl describe resourcegraphdefinition my-rgd

# Check for validation errors
kubectl get resourcegraphdefinition my-rgd -o jsonpath='{.status.conditions}'
```

ResourceGraphDefinitions ont trois conditions de statut clés :

- `ResourceGraphAccepted`- Si le RGD a réussi la validation (syntaxe CEL, vérification de type, existence du champ)
- `KindReady`- Si le CRD de votre API personnalisée a été généré et enregistré
- `ControllerReady`- Si kro surveille activement les instances de votre API personnalisée

Si tel `ResourceGraphAccepted` est le cas`False`, vérifiez que le message de condition ne contient pas d'erreurs de validation, telles que des champs inconnus, des incohérences de type ou des dépendances circulaires.

Instances créées mais les ressources sous-jacentes n'apparaissent pas

Si des instances de ressources personnalisées existent mais que les ressources Kubernetes sous-jacentes (déploiements, services ConfigMaps) ne sont pas créées, vérifiez que kro dispose des autorisations nécessaires et vérifiez qu'il n'y a pas d'erreurs de composition.

Vérifiez l'état de l'instance :

```
# Describe the instance (replace with your custom resource kind and name)
```

```
kubectl describe custom-kind
         my-instance

# View instance events
kubectl get events --field-selector involvedObject.name=my-instance

# Check instance status conditions
kubectl get custom-kind
         my-instance -o jsonpath='{.status.conditions}'

# Check instance state
kubectl get custom-kind
         my-instance -o jsonpath='{.status.state}'
```

Les instances possèdent un `state` champ indiquant un statut élevé :

- `ACTIVE`- L'instance est en cours d'exécution avec succès
- `IN_PROGRESS`- L'instance est en cours de traitement ou de réconciliation
- `FAILED`- L'instance n'a pas pu être réconciliée
- `DELETING`- L'instance est en cours de suppression
- `ERROR`- Une erreur s'est produite lors du traitement

Les instances sont également soumises à quatre conditions de statut :

- `InstanceManaged`- Les finaliseurs et les étiquettes sont correctement définis
- `GraphResolved`- Graphe d'exécution créé et ressources résolues
- `ResourcesReady`- Toutes les ressources créées et prêtes
- `Ready`- Santé globale de l'instance (elle ne l'est que `True` lorsque toutes les sous-conditions le sont`True`)

Concentrez-vous sur la `Ready` condition pour déterminer l'état de santé de l'instance. Si tel `Ready` est le cas`False`, vérifiez les sous-conditions pour identifier la phase qui a échoué.

Vérifiez les autorisations RBAC :

La fonctionnalité Kro nécessite des autorisations pour créer les ressources Kubernetes sous-jacentes définies dans votre. ResourceGraphDefinitions

```
# Check if the capability has the AmazonEKSClusterAdminPolicy
kubectl get accessentry -A | grep kro
```

Si des autorisations sont manquantes, `AmazonEKSClusterAdminPolicy` associez-les à l'entrée d'accès de la fonctionnalité kro ou créez des politiques RBAC plus restrictives pour une utilisation en production. Consultez the section called “Configurer les autorisations ” pour plus de détails.

Erreurs d'expression CEL

Les erreurs d'expression CEL sont détectées au moment de ResourceGraphDefinition la création, et non lors de la création des instances. kro valide toute la syntaxe des CEL, vérifie le type des expressions par rapport aux schémas Kubernetes et vérifie l'existence des champs lorsque vous créez le RGD.

Erreurs de validation CEL courantes :

- Référence de champ non définie : référence à un champ qui n'existe pas dans le schéma ou dans la ressource
- Incompatibilité de type : l'expression renvoie un type incorrect (par exemple, une chaîne où un entier est attendu)
- Syntaxe non valide : crochets, guillemets ou opérateurs manquants dans l'expression CEL
- Type de ressource inconnu : Référencement d'un CRD qui n'existe pas dans le cluster

Vérifiez l'état de validation du RGD :

```
# Check if RGD was accepted
kubectl get resourcegraphdefinition my-rgd -o jsonpath='{.status.conditions[?
(@.type=="ResourceGraphAccepted")]}'

# View detailed validation errors
kubectl describe resourcegraphdefinition my-rgd
```

Si tel `ResourceGraphAccepted` est le cas`False`, le message de condition contient l'erreur de validation.

Exemples d'expressions CEL valides :

```
# Reference schema field
${schema.spec.appName}

# Conditional expression
${schema.spec.replicas > 1}

# String template (expressions must return strings)
name: "${schema.spec.appName}-service"

# Standalone expression (can be any type)
replicas: ${schema.spec.replicaCount}

# Resource reference
${deployment.status.availableReplicas}

# Optional field access (returns null if field doesn't exist)
${configmap.data.?DATABASE_URL}
```

Les dépendances des ressources ne sont pas résolues

kro déduit automatiquement les dépendances à partir des expressions CEL et crée les ressources dans le bon ordre. Si les ressources ne sont pas créées comme prévu, vérifiez l'ordre des dépendances et l'état de préparation des ressources.

Afficher l'ordre de création calculé :

```
# See the order kro will create resources
kubectl get resourcegraphdefinition my-rgd -o jsonpath='{.status.topologicalOrder}'
```

Cela montre l'ordre calculé en fonction des références d'expression CEL entre les ressources.

Vérifiez l'état de préparation des ressources :

```
# View instance status to see which resources are ready
kubectl get custom-kind
         my-instance -o jsonpath='{.status}'

# Check specific resource status
kubectl get deployment my-deployment -o jsonpath='{.status.conditions}'
```

Vérifiez les conditions ReadyWhen (si elles sont utilisées) :

Le champ `readyWhen` est facultatif. Si elles ne sont pas spécifiées, les ressources sont considérées comme prêtes immédiatement après leur création. Si vous avez défini `readyWhen` des conditions, vérifiez qu'elles vérifient correctement l'état de préparation des ressources :

```
resources:
  - id: deployment
    readyWhen:
      - ${deployment.status.availableReplicas == deployment.spec.replicas}
```

Vérifiez les événements liés aux ressources :

```
# View events for the underlying resources
kubectl get events -n namespace --sort-by='.lastTimestamp'
```

Défaillances de validation du schéma

Si les instances ne sont pas créées en raison d'erreurs de validation du schéma, vérifiez que l'instance répond aux exigences du schéma RGD.

Vérifiez les erreurs de validation :

```
# Attempt to create instance and view error
kubectl apply -f instance.yaml

# View existing instance validation status
kubectl describe custom-kind
         my-instance | grep -A 5 "Validation"
```

Problèmes de validation courants :

- Champs obligatoires manquants : l'instance ne fournit pas tous les champs de schéma obligatoires
- Incompatibilité de type : fourniture d'une chaîne où un entier est attendu
- Valeur d'énumération non valide : utilisation d'une valeur ne figurant pas dans la liste autorisée
- Incompatibilité du modèle : la chaîne ne correspond pas au modèle regex

Passez en revue le schéma RGD :

```
# View the schema definition
kubectl get resourcegraphdefinition my-rgd -o jsonpath='{.spec.schema}'
```

Assurez-vous que votre instance fournit tous les champs obligatoires avec les types corrects.

Étapes suivantes

- the section called "Considérations"- considérations clés et meilleures pratiques
- the section called "Configurer les autorisations "- Configurer le RBAC pour les équipes chargées des plateformes et des applications
- the section called "concepts kro"- Comprendre les concepts Kro et le cycle de vie des ressources
- the section called "Fonctionnalités de résolution des problèmes"- Conseils généraux de résolution des problèmes liés aux fonctionnalités

Comparaison entre la capacité EKS pour kro et celle pour kro autogérée

L'EKS Capability for kro fournit les mêmes fonctionnalités que le kro autogéré, mais avec des avantages opérationnels significatifs. Pour une comparaison générale des fonctionnalités EKS par rapport aux solutions autogérées, voirthe section called "Considérations".

La capacité EKS pour kro utilise les mêmes contrôleurs kro en amont et est entièrement compatible avec le kro en amont. ResourceGraphDefinitions, les expressions CEL et la composition des ressources fonctionnent de manière identique. Pour une documentation complète et des exemples de kro, consultez la documentation de kro.

Voie de migration

Vous pouvez passer d'une solution kro autogérée à une fonctionnalité gérée sans aucune interruption de service.

> **Important**
>
> Avant de migrer, assurez-vous que votre contrôleur kro autogéré exécute la même version que la capacité EKS pour kro. Vérifiez la version des fonctionnalités dans la console EKS ou à l'aide de `aws eks describe-capability` celle-ci, puis mettez à niveau votre installation autogérée en conséquence. Cela permet d'éviter les problèmes de compatibilité lors de la migration.

1. Mettez à jour votre contrôleur Kro autogéré pour l'utiliser `kube-system` pour les baux électoraux des dirigeants :

```
helm upgrade --install kro \
  oci://ghcr.io/awslabs/kro/kro-chart \
  --namespace kro \
  --set leaderElection.namespace=kube-system
```

Cela déplace le bail du contrôleur vers`kube-system`, ce qui permet à la capacité gérée de se coordonner avec celui-ci.

2. Créez la fonctionnalité Kro sur votre cluster (voirthe section called “Créez une capacité KRO”)
3. La fonctionnalité gérée reconnaît les instances existantes ResourceGraphDefinitions et prend en charge la réconciliation
4. Diminuez ou supprimez progressivement les déploiements KRO autogérés :

```
helm uninstall kro --namespace kro
```

Cette approche permet aux deux contrôleurs de coexister en toute sécurité pendant la migration. La fonctionnalité gérée adopte automatiquement ResourceGraphDefinitions les instances précédemment gérées par Kro autogéré, garantissant ainsi un rapprochement continu sans conflits.

Étapes suivantes

- the section called “Créez une capacité KRO”- Création d'une ressource de capacité Kro
- the section called “concepts kro”- Comprendre les concepts Kro et la composition des ressources

Dépannage des fonctionnalités EKS

Cette rubrique fournit des conseils de dépannage généraux pour les fonctionnalités EKS, notamment des vérifications de l'état des fonctionnalités, des problèmes courants et des liens vers des solutions de dépannage spécifiques aux fonctionnalités.

> **Note**
>
> Les fonctionnalités EKS sont entièrement gérées et exécutées en dehors de votre cluster. Vous n'avez pas accès aux journaux ni aux espaces de noms des contrôleurs. Le dépannage se concentre sur l'état des fonctionnalités, l'état des ressources et la configuration.

Approche générale du dépannage

Pour résoudre les problèmes liés aux fonctionnalités EKS, suivez cette approche générale :

1. Vérifier l'état des capacités : `aws eks describe-capability` à utiliser pour consulter l'état des capacités et les problèmes de santé
2. Vérifier le statut des ressources : vérifiez les ressources Kubernetes (CRDs) que vous avez créées pour connaître les conditions de statut et les événements
3. Vérifiez les autorisations IAM : assurez-vous que le rôle de capacité dispose des autorisations nécessaires
4. Vérifier la configuration : vérifier que la configuration spécifique à la capacité est correcte

Vérifier l'état des capacités

Toutes les fonctionnalités EKS fournissent des informations sur l'état de santé via la console EKS et l'`describe-capability`API.

Console :

1. Ouvrez la console Amazon EKS à l'adresse https://console.aws.amazon.com/eks/ home#/clusters.
2. Sélectionnez le nom de votre cluster.
3. Sélectionnez l'onglet Observabilité.
4. Sélectionnez Surveiller le cluster.
5. Choisissez l'onglet Fonctionnalités pour afficher l'état et l'état de toutes les fonctionnalités.

L'onglet Fonctionnalités indique :

- Nom et type de capacité
- État actuel
- Problèmes de santé, avec description

AWS CLI :

```
aws eks describe-capability \
  --region region-code \
  --cluster-name my-cluster \
```

```
  --capability-name my-capability-name
```

La réponse inclut :

- statut : État actuel des capacités (`CREATINGACTIVE`,`UPDATING`,`DELETING`,`CREATE_FAILED`,`UPDATE_FAILED`)
- santé : informations relatives à l'état de santé, y compris les éventuels problèmes détectés par la fonctionnalité

Statuts de capacité communs

CRÉATION : La capacité est en cours de configuration.

ACTIF : La fonctionnalité fonctionne et est prête à être utilisée. Si les ressources ne fonctionnent pas comme prévu, vérifiez l'état des ressources et les autorisations IAM.

MISE À JOUR : Les modifications de configuration sont en cours d'application. Attendez que le statut revienne à`ACTIVE`.

CREATE_FAILED ou UPDATE_FAILED : le programme d'installation ou de mise à jour a rencontré une erreur. Consultez la section santé pour plus de détails. Causes courantes :

- Politique de confiance des rôles IAM incorrecte ou manquante
- Le rôle IAM n'existe pas ou n'est pas accessible
- Problèmes d'accès au cluster
- Paramètres de configuration non valides

Vérifier l'état des ressources Kubernetes

Les fonctionnalités EKS créent et gèrent des définitions de ressources personnalisées Kubernetes (CRDs) dans votre cluster. Lors du dépannage, vérifiez l'état des ressources que vous avez créées :

```
# List resources of a specific type
kubectl get resource-kind -A

# Describe a specific resource to see conditions and events
kubectl describe resource-kind
         resource-name -n namespace
```

```
# View resource status conditions
kubectl get resource-kind
         resource-name -n namespace -o jsonpath='{.status.conditions}'

# View events related to the resource
kubectl get events --field-selector involvedObject.name=resource-name -n namespace
```

Les conditions d'état des ressources fournissent des informations sur :

- Si la ressource est prête
- Toute erreur rencontrée
- État actuel du rapprochement

Vérifiez les autorisations IAM et l'accès au cluster

De nombreux problèmes de capacité sont dus à des problèmes d'autorisation IAM ou à une configuration d'accès au cluster manquante. Vérifiez à la fois les autorisations du rôle de capacité et les entrées d'accès au cluster.

Vérifier les autorisations des rôles IAM

Vérifiez que le rôle de capacité dispose des autorisations nécessaires :

```
# List attached managed policies
aws iam list-attached-role-policies --role-name my-capability-role

# List inline policies
aws iam list-role-policies --role-name my-capability-role

# Get specific policy details
aws iam get-role-policy --role-name my-capability-role --policy-name policy-name

# View the role's trust policy
aws iam get-role --role-name my-capability-role --query 'Role.AssumeRolePolicyDocument'
```

La politique de confiance doit permettre au principal du `capabilities.eks.amazonaws.com` service :

```
{
```

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

Vérifiez les entrées d'accès et les politiques d'accès d'EKS

Toutes les fonctionnalités nécessitent des entrées d'accès et des politiques d'accès EKS appropriées sur le cluster sur lequel elles fonctionnent.

Vérifiez que l'entrée d'accès existe :

```
aws eks list-access-entries \
  --cluster-name my-cluster \
  --region region-code
```

Recherchez l'ARN du rôle de capacité dans la liste. Si elle est absente, la fonctionnalité ne peut pas accéder au cluster.

Vérifiez les politiques d'accès jointes à l'entrée :

```
aws eks list-associated-access-policies \
  --cluster-name my-cluster \
  --principal-arn arn:aws:iam::111122223333:role/my-capability-role \
  --region region-code
```

Toutes les fonctionnalités nécessitent des politiques d'accès appropriées :

- ACK : nécessite des autorisations pour créer et gérer des ressources Kubernetes
- kro : nécessite des autorisations pour créer et gérer des ressources Kubernetes
- Argo CD : nécessite des autorisations pour créer et gérer des applications, et nécessite des entrées d'accès sur les clusters cibles distants pour les déploiements multi-clusters

Pour les déploiements multi-clusters Argo CD :

En cas de déploiement sur des clusters distants, vérifiez que le rôle de capacité possède une entrée d'accès sur chaque cluster cible :

```
# Check Access Entry on target cluster
aws eks describe-access-entry \
  --cluster-name target-cluster \
  --principal-arn arn:aws:iam::111122223333:role/argocd-capability-role \
  --region region-code
```

Si l'entrée d'accès est absente sur un cluster cible, Argo CD ne peut pas y déployer d'applications. Voir the section called “Enregistrer des clusters” pour les détails de configuration.

Résolution des problèmes spécifiques aux capacités

Pour obtenir des conseils de dépannage détaillés spécifiques à chaque type de fonctionnalité :

- the section called “Résolution des problèmes”- Résoudre les problèmes liés à la création de ressources ACK, aux autorisations IAM et à l'accès entre comptes
- the section called “Résolution des problèmes”- Résoudre les problèmes de synchronisation des applications, d'authentification des référentiels et de déploiements multiclusters
- the section called “Résolution des problèmes”- Résolution des problèmes ResourceGraphDefinitions, expressions CEL et autorisations RBAC

Problèmes courants liés à toutes les fonctionnalités

Capacité bloquée dans l'état CREATING

Si une capacité reste `CREATING` active plus longtemps que prévu :

1. Vérifiez l'état des fonctionnalités pour détecter des problèmes spécifiques dans la console (Observabilité > Surveiller le cluster > onglet Fonctionnalités) ou à l'aide de la AWS CLI :

```
aws eks describe-capability \
  --region region-code \
  --cluster-name my-cluster \
  --capability-name my-capability-name \
```

```
  --query 'capability.health'
```

2. Vérifiez que le rôle IAM existe et que la politique de confiance est correcte
3. Assurez-vous que votre cluster est accessible et sain
4. Vérifiez s'il existe des problèmes au niveau du cluster susceptibles d'empêcher la configuration des fonctionnalités

Ressources non créées ou mises à jour

Si la fonctionnalité existe `ACTIVE` mais que les ressources ne sont pas créées ou mises à jour :

1. Vérifiez l'état de la ressource pour détecter les conditions d'erreur
2. Vérifiez les autorisations IAM pour les AWS services (ACK) ou les référentiels spécifiques (Argo CD)
3. Vérifiez les autorisations RBAC pour créer des ressources sous-jacentes (kro)
4. Vérifiez les spécifications des ressources pour détecter les erreurs de validation

L'état des capacités montre des problèmes

Il `describe-capability` présente des problèmes de santé :

1. Lisez attentivement les descriptions du problème : elles indiquent souvent le problème spécifique
2. Corriger la cause première (autorisations IAM, erreurs de configuration, etc.)
3. La fonctionnalité sera automatiquement rétablie une fois le problème résolu

Étapes suivantes

- the section called “Travailler avec les capacités”- Gérer les ressources en matière de capacités
- the section called “Résolution des problèmes”- Dépannage spécifique à ACK
- the section called “Résolution des problèmes”- Résolution des problèmes spécifiques à Argo CD
- the section called “Résolution des problèmes”- résolution des problèmes spécifiques à Kro
- the section called “Considérations relatives aux fonctionnalités d'EKS”- Meilleures pratiques en matière de sécurité en matière de fonctionnalités

Organiser et surveiller les ressources du cluster

Ce chapitre porte sur les rubriques suivantes relatives à la gestion de votre cluster. Vous pouvez également afficher des informations sur vos ressources Kubernetes avec la AWS Management Console.

- Le tableau de bord Kubernetes est une interface utilisateur web polyvalente destinée aux clusters Kubernetes. Il permet aux utilisateurs de gérer les applications exécutées dans le cluster et de les dépanner, ainsi que de gérer le cluster lui-même. Pour plus d'informations, consultez le référentiel GitHub du tableau de bord Kubernetes.
- the section called "Serveur de métriques" : le serveur de métriques Kubernetes est un agrégateur de données d'utilisation des ressources dans votre cluster. Il n'est pas déployé par défaut dans votre cluster, mais il est utilisé par les modules complémentaires de Kubernetes, tels que Kubernetes Dashboard et the section called "Horizontal Pod Autoscaler". Dans cette rubrique, vous allez apprendre à installer le serveur de métriques.
- the section called "Déployer des applications avec Helm" : le gestionnaire de paquets Helm pour Kubernetes vous aide à installer et à gérer des applications sur votre cluster Kubernetes. Cette rubrique vous aide à installer et à exécuter les binaires Helm, afin de pouvoir installer et gérer des graphiques à l'aide de la CLI Helm sur votre ordinateur local.
- the section called "Balisage de vos ressources " : pour vous aider à gérer vos ressources Amazon EKS, vous pouvez attribuer vos propres métadonnées à chaque ressource sous forme d'identifications. Cette rubrique décrit les identifications et explique comment les créer.
- the section called "Quotas de service" : votre compte AWS dispose de quotas par défaut, anciennement appelés limites, pour chaque service AWS. Découvrez les quotas pour Amazon EKS et la manière de les augmenter.

Surveillance et optimisation des coûts du cluster Amazon EKS

La surveillance des coûts est un élément essentiel de la gestion des clusters Kubernetes sur Amazon EKS. En obtenant une visibilité complète sur les coûts de votre cluster, vous pouvez optimiser l'utilisation des ressources, définir des budgets et prendre des décisions fondées sur les données concernant vos déploiements. Amazon EKS propose deux solutions de surveillance des coûts, chacune présentant des avantages uniques, pour vous aider à suivre et répartir efficacement vos coûts :

Données de ventilation des coûts de facturation AWS pour Amazon EKS : cette fonctionnalité native s'intègre de manière fluide à la console de facturation AWS, vous permettant d'analyser et de répartir les coûts à l'aide de la même interface et des mêmes flux de travail que pour vos autres services AWS. Grâce à la ventilation des coûts, vous obtenez une visibilité directe sur vos coûts Kubernetes aux côtés de vos autres dépenses AWS, ce qui facilite l'optimisation des coûts à l'échelle de l'environnement AWS. Vous pouvez également exploiter les fonctionnalités existantes de la facturation AWS, telles que les catégories de coûts et la détection des anomalies de coûts, afin d'améliorer encore vos capacités de gestion financière. Pour plus d'informations, consultez la section Comprendre les données de ventilation des coûts dans le Guide de facturation AWS.

Kubecost : Amazon EKS prend en charge Kubecost, un outil de surveillance des coûts Kubernetes. Kubecost propose une approche native Kubernetes et riche en fonctionnalités pour la surveillance des coûts, offrant une analyse détaillée des coûts par ressource Kubernetes, des recommandations d'optimisation des coûts et des tableaux de bord et rapports prêts à l'emploi. Kubecost extrait également des données tarifaires précises grâce à son intégration avec le rapport sur les coûts et l'utilisation AWS, garantissant une vue exacte de vos coûts Amazon EKS. Découvrez comment installer Kubecost. Consultez la page Kubecost sur AWS Marketplace pour obtenir des informations sur l'obtention d'un abonnement Kubecost gratuit.

Afficher les coûts par module dans la AWS facturation avec répartition des coûts fractionnée

Surveillance des coûts à l'aide des données de répartition des coûts AWS fractionnés pour Amazon EKS

Vous pouvez utiliser AWS les données de répartition des coûts pour Amazon EKS afin d'obtenir une visibilité précise des coûts pour vos clusters Amazon EKS. Cela vous permet d'analyser, d'optimiser et de refacturer les coûts et l'utilisation de vos applications Kubernetes. Vous répartissez les coûts des applications entre les unités commerciales et les équipes individuelles en fonction des ressources de EC2 processeur et de mémoire Amazon consommées par votre application Kubernetes. Les données de répartition des coûts pour Amazon EKS offrent une visibilité sur le coût par pod et vous permettent d'agréger les données de coût par pod à l'aide de l'espace de noms, du cluster et d'autres primitives Kubernetes. Voici quelques exemples de primitives Kubernetes que vous pouvez utiliser pour analyser les données de répartition des coûts Amazon EKS.

- Nom du cluster
- Déploiement

- Namespace
- Nœud
- Nom de la charge de travail
- Type de charge de travail

Les balises de répartition des coûts définies par l'utilisateur sont également prises en charge. Pour plus d'informations sur l'utilisation des données de répartition des coûts fractionnés, consultez la section Comprendre les données de répartition des coûts dans le Guide AWS de l'utilisateur de facturation.

Configurer les rapports sur les coûts et l'utilisation

Vous pouvez activer les données de répartition des coûts pour EKS dans la console de gestion des coûts, l'interface de ligne de AWS commande ou le AWS SDKs.

Utilisez les éléments suivants pour les données de répartition des coûts :

1. Accepter le partage des données relatives à la répartition des coûts. Pour plus d'informations, consultez la section Activation des données de répartition des coûts fractionnés dans le Guide de l'utilisateur du rapport sur les AWS coûts et l'utilisation.
2. Inclure les données dans un rapport nouveau ou existant.
3. Consultez le rapport. Vous pouvez utiliser la console de Facturation et gestion des coûts ou consulter les fichiers de rapports dans Amazon Simple Storage Service.

Installer Kubecost

Amazon EKS prend en charge Kubecost, que vous pouvez utiliser pour surveiller vos coûts ventilés par ressources Kubernetes, notamment les pods, les nœuds, les espaces de noms et les étiquettes. Cette rubrique traite de l'installation de Kubecost et de l'accès au tableau de bord Kubecost.

Amazon EKS fournit un ensemble AWS optimisé de Kubecost pour la visibilité des coûts des clusters. Vous pouvez utiliser vos contrats de AWS support existants pour obtenir de l'aide. Pour plus d'informations sur les versions disponibles de Kubecost, consultez the section called "En savoir plus sur Kubecost".

> **Note**
>
> Kubecost v3 introduit des améliorations architecturales majeures, notamment des performances nettement plus rapides et des capacités d'automatisation améliorées. En savoir plus sur Kubecost v3.
> Kubecost v2 introduit plusieurs nouvelles fonctionnalités importantes. En savoir plus sur Kubecost v2..

Pour plus d'informations sur Kubecost, consultez la documentation Kubecost et la foire aux questions.

Installer le bundle Kubecost optimisé pour Amazon EKS

Vous pouvez utiliser l'une des procédures suivantes pour installer le bundle Kubecost optimisé pour Amazon EKS :

- Avant de commencer, il est recommandé de consulter la section Présentation de l'architecture Kubecost afin de comprendre comment Kubecost fonctionne sur Amazon EKS.
- Si vous débutez avec Amazon EKS, nous vous recommandons d'utiliser le module complémentaire Amazon EKS pour l'installation, car il simplifie l'installation du bundle Kubecost optimisé pour Amazon EKS. Pour plus d'informations, consultez Déploiement de Kubecost sur un cluster Amazon EKS à l'aide du module complémentaire Amazon EKS.
- Pour personnaliser l'installation, vous pouvez configurer votre bundle Kubecost optimisé pour Amazon EKS avec Helm. Pour plus d'informations, consultez la section Déploiement de Kubecost sur un cluster Amazon EKS à l'aide de Helm dans la documentation Kubecost.

> **Important**
>
> Pour Kubecost v3, l'emplacement du graphique Helm est devenu. `public.ecr.aws/kubecost/kubecost` Si vous effectuez une mise à niveau depuis la v2, mettez à jour les références de votre référentiel Helm en conséquence.

> **Note**
>
> Pour les déploiements multi-clusters avec Kubecost v3, vous avez besoin d'un stockage d'objets compatible AWS S3 (S3 pour les clients EKS) pour le stockage des métriques. Cela remplace le stockage compatible Prometheus utilisé dans la version 2. Pour plus d'informations, consultez la section Installation multi-clusters dans la documentation de Kubecost.

Accéder au tableau de bord Kubecost

Une fois la configuration du bundle Kubecost optimisé pour Amazon EKS terminée, vous devriez avoir accès au tableau de bord Kubecost. Pour de plus amples informations, veuillez consulter the section called “Accéder au tableau de bord Kubecost”.

Accéder au tableau de bord Kubecost

Conditions préalables

1. Assurez-vous que l'état des pods liés à kubecost est « En cours d'exécution ».

```
kubectl get pods --namespace kubecost
```

Accéder au tableau de bord Kubecost

1. Sur votre appareil, activez la redirection de port pour afficher le tableau de bord Kubecost.
   - Si kubecost v3 est installé à l'aide de helm :

     ```
     kubectl port-forward deployment/kubecost-frontend 9090 --namespace kubecost
     ```

   - Si kubecost v1 ou v2 est installé à l'aide de helm :

     ```
     kubectl port-forward deployment/kubecost-cost-analyzer 9090 --namespace kubecost
     ```

   - Si kubecost est installé à l'aide du module complémentaire Amazon EKS :

     ```
     kubectl port-forward deployment/cost-analyzer 9090 --namespace kubecost
     ```

Vous pouvez également utiliser le contrôleur d'équilibreur de charge AWS pour exposer Kubecost et utiliser Amazon Cognito pour l'authentification, l'autorisation et la gestion des utilisateurs. Pour plus d'informations, consultez Comment utiliser Application Load Balancer et Amazon Cognito pour authentifier les utilisateurs de vos applications web Kubernetes.

2. Depuis le même appareil qui vous a servi à effectuer l'étape précédente, ouvrez un navigateur Web et entrez l'adresse suivante.

```
http://localhost:9090
```

La page Aperçu Kubecost s'affiche dans votre navigateur. La collecte des métriques par Kubecost peut prendre entre 5 et 10 minutes (voire plus), selon la taille de votre cluster. Vous pouvez consulter vos dépenses Amazon EKS, y compris les coûts cumulés des clusters, les coûts associés aux ressources Kubernetes et les dépenses mensuelles agrégées.

3. Pour suivre les coûts au niveau du cluster, balisez vos ressources Amazon EKS pour la facturation. Pour de plus amples informations, veuillez consulter the section called "Balisage de vos ressources pour facturation".
   - Allocation des coûts : Consultez les coûts mensuels d'Amazon EKS et les coûts cumulés pour chacun de vos espaces de noms et d'autres dimensions au cours des sept derniers jours. Cela est utile pour comprendre quels secteurs de votre application contribuent aux dépenses d'Amazon EKS.
   - Ressources : consultez les coûts des actifs AWS d'infrastructure associés à vos ressources Amazon EKS.

En savoir plus sur Kubecost

Amazon EKS fournit un ensemble AWS optimisé de Kubecost pour la visibilité des coûts des clusters. Amazon EKS prend en charge Kubecost, que vous pouvez utiliser pour surveiller vos coûts ventilés par ressources Kubernetes, notamment les pods, les nœuds, les espaces de noms et les étiquettes.

Cette rubrique traite des versions disponibles de Kubecost et des différences entre les niveaux proposés. EKS prend en charge les versions 1, 2 et 3 de Kubecost. Chaque version est disponible en plusieurs niveaux. Vous pouvez utiliser le bundle Kubecost optimisé pour Amazon EKS pour vos clusters Amazon EKS sans frais supplémentaires. L'utilisation de AWS services associés, tels qu'Amazon Managed Service pour Prometheus, peut vous être facturée. Vous pouvez également utiliser vos contrats de AWS support existants pour obtenir de l'aide.

En tant qu'administrateur de la plateforme Kubernetes et responsable financier, vous pouvez utiliser Kubecost pour visualiser la répartition des frais Amazon EKS, répartir les coûts et refacturer les unités organisationnelles telles que les équipes d'application. Vous pouvez fournir à vos équipes internes et à vos unités commerciales des données de coûts transparentes et précises sur la base de leur AWS facture réelle. En outre, vous pouvez également obtenir des recommandations personnalisées pour l'optimisation des coûts en fonction de leur environnement d'infrastructure et des modèles d'utilisation au sein de leurs clusters. Pour plus d'informations sur Kubecost, consultez la documentation Kubecost.

Quelle est la différence entre le bundle personnalisé de Kubecost et la version gratuite de Kubecost (également connue sous le nom de) ? OpenCost

AWS et Kubecost ont collaboré pour proposer une version personnalisée de Kubecost. Cette version inclut un sous-ensemble de fonctions commerciales sans frais supplémentaires. Consultez les tableaux ci-dessous pour connaître les fonctionnalités incluses dans l'offre personnalisée de Kubecost.

Kubecost v3

Quelle est la différence entre Kubecost v2 et v3 ?

Kubecost 3.0 est une mise à niveau architecturale majeure qui offre des performances nettement plus rapides, une évolutivité améliorée et des capacités d'optimisation proactives. Le changement le plus important est la migration vers une ClickHouse base de données, en remplacement de DuckDB depuis la version 2.8, qui fournit des requêtes nettement plus rapides et des performances plus fiables à grande échelle. Kubecost 3.0 introduit également un agent unifié qui combine les fonctionnalités de Kubecost et de Cloudability, éliminant ainsi la dépendance à Prometheus et réduisant l'encombrement mémoire tout en préservant la compatibilité. OpenCost

> **Important**
>
> Consultez la documentation de Kubecost avant de passer à la version 3. La migration depuis la version 2 nécessite une planification minutieuse et peut avoir un impact sur la disponibilité des rapports pendant la transition. L'emplacement du graphique Helm a été modifié en`public.ecr.aws/kubecost/kubecost`.

Principales améliorations architecturales de la version 3 :

- ClickHouse Base de données : remplace DuckDB pour des requêtes nettement plus rapides et une meilleure évolutivité
- Agent unifié : combine les fonctionnalités de Kubecost et de Cloudability, éliminant ainsi la dépendance à Prometheus
- Stockage compatible S3 pour plusieurs clusters : pour les déploiements multi-clusters, la version 3 utilise un stockage d'objets compatible AWS S3 (S3 pour les clients EKS) au lieu d'un stockage compatible avec Prometheus comme Amazon Managed Service for Prometheus. L' FinOps agent extrait les métriques de l'API Kubernetes et les transfère vers un stockage compatible S3, puis l'agrégateur extrait ces données, effectue les étapes de dérivation et affiche les résultats dans le frontend. Pour plus d'informations, consultez le guide d'installation de plusieurs clusters et de clusters secondaires dans la documentation de Kubecost.
- Empreinte mémoire réduite : réduction significative des besoins en ressources tout en préservant les fonctionnalités
- Architecture simplifiée : topologie de pod à conteneur unique pour une mise à l'échelle indépendante et une résilience améliorée
- Automatisation améliorée : dimensionnement automatisé des demandes de conteneurs avec prise en compte de plusieurs clusters et profils personnalisés

Avantages du bundle optimisé Amazon EKS dans la version 3 :

Le pack Kubecost optimisé pour Amazon EKS continue d'être disponible sans frais supplémentaires et est exempté de la nouvelle limite de dépenses de 100 000 USD introduite dans le niveau gratuit de Kubecost v3. Les utilisateurs d'EKS conservent un accès complet à toutes les fonctionnalités de dépenses de Kubernetes, quel que soit le niveau de dépenses.

Comparaison des fonctionnalités de base :

Fonctionnalité	Niveau 3.0 gratuit de Kubecost	Bundle Kubecost 3.0 optimisé pour Amazon EKS	Kubecost Entreprise 3.0
Visibilité des coûts du cluster	Clusters illimités , limités à des dépenses de 100 000 dollars américains sur 30 jours	Multicluster unifié sans limites de dépenses	Nombre unifié et illimité de clusters dans un nombre illimité d'environ

Fonctionnalité	Niveau 3.0 gratuit de Kubecost	Bundle Kubecost 3.0 optimisé pour Amazon EKS	Kubecost Entreprise 3.0
			nements (c'est-à-dire multi-cloud)
Backend de base de données	ClickHouse (local)	ClickHouse avec stockage compatible S3 pour les métriques multi-clusters	ClickHouse avec des options de base de données personnal isées
Performance	Requêtes nettement plus rapides par rapport à la version v2	Requêtes nettement plus rapides par rapport à la version v2	Requêtes nettement plus rapides par rapport à la version v2
Empreinte mémoire	Réduit par rapport à la v2 (aucune dépendance à Prometheus)	Réduit par rapport à la v2 (aucune dépendance à Prometheus)	Réduit par rapport à la v2 (aucune dépendance à Prometheus)
Dimensionnement automatique des demandes de conteneurs	Disponible (limité à 250 cœurs)	Disponible sans limites de base	Disponible sans limites de base
Limites de dépenses	100 000$ US sur 30 jours	Aucune limite de dépenses	Aucune limite de dépenses
Automatisation multi-clusters	Limité	Connaissance complète de plusieurs clusters grâce à une messagerie sécurisée	Connaissance complète de plusieurs clusters grâce à une messagerie sécurisée

Kubecost v2

Quelle est la différence entre Kubecost v1 et v2 ?

Kubecost 2.0 est une mise à jour majeure par rapport aux versions précédentes et comprend de nouvelles fonctionnalités importantes, notamment une toute nouvelle API dorsale. Notez que l'allocation et APIs les actifs sont entièrement rétrocompatibles. Veuillez consulter la documentation Kubecost afin d'assurer une transition en douceur. Pour obtenir la liste complète des améliorations, veuillez consulter l'annonce relative à Kubecost v2.0 et les notes de mise à jour complètes.

> **Important**
>
> Consultez la documentation de Kubecost avant de procéder à la mise à niveau. La mise à niveau peut avoir un impact sur la disponibilité des rapports.

Comparaison des fonctionnalités de base :

Fonctionnalité	Kubecost niveau gratuit 2.0	Ensemble Kubecost optimisé pour Amazon EKS 2.0	Kubecost Entreprise 2.0
Visibilité des coûts du cluster	Clusters illimités jusqu'à 250 cœurs	Multi-cluster unifié sans limites de cœur lorsqu'il est intégré au service géré Amazon pour Prometheus	Nombre unifié et illimité de clusters dans un nombre illimité d'environnements (c'est-à-dire multi-cloud)
Déploiement	Hébergé par l'utilisateur	Hébergé par l'utilisateur	Hébergé par l'utilisateur, hébergé par Kubecost (locataire dédié), SaaS
Bases de données prises en charge	Prométhée local	Service géré Amazon pour Prometheus ou Prometheus local	Toutes les saveurs Prometheus et bases de données personnalisées
Prise en charge de la conservation des	15 jours	Données historiques illimitées	Données historiques illimitées

Fonctionnalité	Kubecost niveau gratuit 2.0	Ensemble Kubecost optimisé pour Amazon EKS 2.0	Kubecost Entreprise 2.0
bases de données (mesures brutes)			
API Kubecost et rétention de l'interface utilisateur (ETL)	15 jours	15 jours	Illimité
Visibilité du cloud hybride	-	Clusters Amazon EKS et Amazon EKS Anywhere	Multi-cloud et cloud hybride
Alertes et rapports récurrents	Uniquement pris en charge sur le cluster principal, limité à 250 cœurs	Alertes d'efficacité, alertes budgétaires, alertes de modification des dépenses et bien plus encore prises en charge dans tous les clusters	Alertes d'efficacité, alertes budgétaires, alertes de modification des dépenses et bien plus encore prises en charge dans tous les clusters
Rapports enregistrés	-	Rapports utilisant 15 jours de métriques	Rapports utilisant des données historiques et des métriques illimitées
Intégration de la facturation dans le cloud	Uniquement pris en charge sur le cluster principal, limité à 250 cœurs	Assistance tarifaire personnalisée pour AWS (y compris plusieurs clusters et plusieurs comptes)	Assistance personnalisée pour la tarification de n'importe quel cloud
Recommandations en matière d'épargne	Uniquement pris en charge sur le cluster principal, limité à 250 cœurs	Informations sur le cluster principal, mais sans limite de 250 cœurs	Informations sur plusieurs clusters

Fonctionnalité	Kubecost niveau gratuit 2.0	Ensemble Kubecost optimisé pour Amazon EKS 2.0	Kubecost Entreprise 2.0
Gouvernance : audits	-	-	Audit des événements liés aux coûts historiques
Prise en charge de l'authentification unique (SSO)	-	Prise en charge d'Amazon Cognito	Okta, Auth0, PingID et tout autre élément KeyCloak personnalisé
Contrôle d'accès basé sur les rôles (RBAC) avec SAML 2.0	-	-	Okta, Auth0, PingID et tout autre élément KeyCloak personnalisé
Formation et intégration en entreprise	-	-	Service complet de formation et d'intégration FinOps
Équipes	-	-	Oui

Nouvelles fonctions :

Les fonctionnalités suivantes ont des limites métriques :

- Agrégateur Kubecost
- Surveillance réseau
- Actions Kubecost
- Collections
- Détection des anomalies
- Demande de conteneur adapté
- Prévisions Kubecost
- Saisie semi-automatique pour le filtrage et l'agrégation

Limites métriques :

Métrique	Kubecost Free Tier 2.0	Ensemble Kubecost optimisé pour Amazon EKS 2.0	Kubecost Entrepris e 2.0
Taille du cluster	Clusters illimités jusqu'à 250 cœurs	Illimité	Illimité
Coinservation des métriques	15 jours	15 jours	Illimité
Prise en charge multi-clusters	Non disponible	Available	Available
Limites de cœurs	250 cœurs par cluster	Aucune limite de cœurs	Aucune limite de cœurs

Kubecost v1

Fonctionnalité	offre gratuite Kubecost	Ensemble Kubecost optimisé pour Amazon EKS	Kubecost Entreprise
Déploiement	Hébergé par l'utilisa teur	Hébergé par l'utilisa teur	Hébergé par l'utilisa teur ou hébergé par Kubecost (SaaS)
Nombre de clusters pris en charge	Illimité	Illimité	Illimité
Bases de données prises en charge	Prométhée local	Prometheus local ou service géré Amazon pour Prometheus	Prometheus, service géré Amazon pour Prometheus, Cortex ou Thanos

Fonctionnalité	offre gratuite Kubecost	Ensemble Kubecost optimisé pour Amazon EKS	Kubecost Entreprise
Prise en charge de la conservation des bases de données	15 jours	Données historiques illimitées	Données historiques illimitées
Rétention de l'API Kubecost (ETL)	15 jours	15 jours	Données historiques illimitées
Visibilité des coûts du cluster	Clusters uniques	Multicluster unifié	Multicluster unifié
Visibilité du cloud hybride	-	Clusters Amazon EKS et Amazon EKS Anywhere	Support multicloud et cloud hybride
Alertes et rapports récurrents	-	Prise en charge des alertes d'efficacité, des alertes de budget, des alertes de modification des dépenses, et bien d'autres encore	Prise en charge des alertes d'efficacité, des alertes de budget, des alertes de modification des dépenses, et bien d'autres encore
Rapports enregistrés	-	Rapports utilisant des données sur 15 jours	Rapports utilisant des données historiques illimitées
Intégration de la facturation dans le cloud	Nécessaire pour chaque cluster individuel	Assistance tarifaire personnalisée pour AWS (y compris plusieurs clusters et plusieurs comptes)	Assistance tarifaire personnalisée pour AWS (y compris plusieurs clusters et plusieurs comptes)
Recommandations en matière d'épargne	Informations sur un seul cluster	Informations sur un seul cluster	Informations sur plusieurs clusters

Fonctionnalité	offre gratuite Kubecost	Ensemble Kubecost optimisé pour Amazon EKS	Kubecost Entreprise
Gouvernance : audits	-	-	Audit des événements liés aux coûts historiques
Prise en charge de l'authentification unique (SSO)	-	Prise en charge d'Amazon Cognito	Okta, Auth0, PingID, KeyCloak
Contrôle d'accès basé sur les rôles (RBAC) avec SAML **2.0**	-	-	Okta, Auth0, PingID, Keycloak
Formation et intégration en entreprise	-	-	Service complet de formation et d'intégration FinOps

Questions fréquentes (FAQ)

Consultez les questions et réponses courantes suivantes concernant l'utilisation de Kubecost avec Amazon EKS.

Qu'est-ce que la fonctionnalité de conservation (ETL) de l'API Kubecost ?

La fonctionnalité ETL de Kubecost agrège et organise les métriques afin d'offrir une visibilité sur les coûts à différents niveaux de granularité (tels que `namespace-level`, `pod-level`, et `deployment-level`). Pour le bundle Kubecost optimisé pour Amazon EKS, les clients obtiennent des données et des informations issues des métriques des 15 derniers jours.

Qu'est-ce que la fonctionnalité d'alertes et de rapports récurrents ? Quelles alertes et quels rapports comprend-il ?

Les alertes Kubecost permettent aux équipes de recevoir des mises à jour en temps réel sur les dépenses Kubernetes et les dépenses cloud. Les rapports récurrents permettent aux équipes de recevoir des vues personnalisées de l'historique des dépenses liées à Kubernetes et au cloud. Les

deux sont configurables à l'aide de l'interface utilisateur Kubecost ou des valeurs Helm. Ils prennent en charge les e-mails, Slack et Microsoft Teams.

Que contiennent les rapports enregistrés ?

Les rapports enregistrés dans Kubecost sont des vues prédéfinies des métriques de coût et d'efficacité. Ils incluent le coût par cluster, l'espace de noms, l'étiquette et bien plus encore.

Qu'est-ce que l'intégration de la facturation dans le cloud?

L'intégration à AWS la facturation APIs permet à Kubecost d'afficher out-of-cluster les coûts (comme Amazon S3). De plus, cela permet à Kubecost de rapprocher ses prévisions au sein du cluster avec les données de facturation réelles afin de tenir compte de l'utilisation ponctuelle, des plans d'économies et des remises accordées aux entreprises.

Que comprennent les recommandations en matière d'épargne?

Kubecost fournit des informations et des fonctionnalités d'automatisation pour aider les utilisateurs à optimiser leur infrastructure Kubernetes et leurs dépenses.

Cette fonctionnalité est-elle payante ?

Non. Vous pouvez utiliser le bundle Kubecost optimisé pour Amazon EKS sans frais supplémentaires. Si vous souhaitez des fonctionnalités supplémentaires de Kubecost qui ne sont pas incluses, vous pouvez acheter une licence d'entreprise de Kubecost via le AWS Marketplace ou directement auprès de Kubecost.

Une assistance est-elle disponible pour le bundle Kubecost optimisé pour Amazon EKS ?

Oui, uniquement si vous utilisez le bundle Kubecost optimisé pour Amazon EKS.

Comment puis-je obtenir de l'assistance pour le bundle Kubecost optimisé pour Amazon EKS ?

Vous pouvez ouvrir un dossier d'assistance auprès de l'équipe d' AWS assistance sur Contact AWS.

Ai-je besoin d'une licence pour utiliser les fonctionnalités Kubecost fournies par l'intégration Amazon EKS ?

Non.

Puis-je intégrer Kubecost au rapport sur les AWS coûts et l'utilisation pour des rapports plus précis ?

Oui. Vous pouvez configurer Kubecost pour intégrer les données du rapport sur les AWS coûts et l'utilisation afin d'obtenir une visibilité précise des coûts, y compris les remises, les tarifs ponctuels,

les prix des instances réservées, etc. Pour plus d'informations, consultez la section Intégration de la facturation Cloud AWS dans la documentation Kubecost.

Cette version prend-elle en charge la gestion des coûts des clusters Kubernetes autogérés sur Amazon ? EC2

Non. Le bundle Kubecost optimisé pour Amazon EKS est uniquement compatible avec les clusters Amazon EKS.

Kubecost peut-il suivre les coûts d'Amazon EKS sur Fargate ? AWS

Kubecost met tout en œuvre pour afficher la visibilité des coûts des clusters pour Amazon EKS sur Fargate, mais avec une précision moindre qu'avec Amazon EKS sur Amazon. EC2 Cela s'explique principalement par la différence dans la manière dont votre consommation vous est facturée. Avec Amazon EKS sur Fargate, vous êtes facturé pour les ressources consommées. Avec Amazon EKS sur les EC2 nœuds Amazon, les ressources mises en service vous sont facturées. Kubecost calcule le coût d'un EC2 nœud Amazon en fonction de la spécification du nœud, qui inclut le processeur, la RAM et le stockage éphémère. Avec Fargate, les coûts sont calculés en fonction des ressources demandées pour les pods Fargate.

How can I get updates and new versions of Kubecost ?

Vous pouvez mettre à niveau votre version de Kubecost à l'aide des procédures de mise à niveau Helm standard. Pour Kubecost v3, les dernières versions sont disponibles sur le nouvel emplacement du graphique Helm. `public.ecr.aws/kubecost/kubecost` Les versions précédentes (v1 et v2) restent disponibles dans la galerie publique Amazon ECR.

> **Important**
>
> Lors de la mise à niveau vers Kubecost v3, notez que l'emplacement du graphique Helm est passé de à. `public.ecr.aws/kubecost/cost-analyzer` `public.ecr.aws/kubecost/kubecost` Mettez à jour les références de votre référentiel Helm en conséquence.

La **`kubectl-cost`** CLI est-elle prise en charge ? Comment l'installer?

Oui. `Kubectl-cost` est un outil open source développé par Kubecost (licence Apache 2.0) qui fournit un accès CLI aux métriques de répartition des coûts Kubernetes. Pour procéder à l'installation`kubectl-cost`, reportez-vous à la section Installation sur GitHub.

L'interface utilisateur Kubecost est-elle prise en charge ? Comment y accéder ?

Kubecost fournit un tableau de bord Web auquel vous pouvez accéder via la redirection de port `kubectl`, une entrée ou un équilibreur de charge. Vous pouvez également utiliser le AWS Load Balancer Controller pour exposer Kubecost et utiliser Amazon Cognito pour l'authentification, les autorisations et la gestion des utilisateurs. Pour plus d'informations, consultez Comment utiliser Application Load Balancer et Amazon Cognito pour authentifier les utilisateurs de vos applications Web Kubernetes sur le blog. AWS

La nouvelle limite de dépenses de 100 000$ dans Kubecost v3 affecte-t-elle les utilisateurs d'Amazon EKS ?

Non La limite de dépenses de 100 000 USD sur 30 jours introduite dans le niveau gratuit de Kubecost v3 ne s'applique pas aux utilisateurs du bundle Kubecost optimisé pour Amazon EKS. Les utilisateurs d'EKS conservent un accès complet à toutes les fonctionnalités de dépenses de Kubernetes, quel que soit le niveau de dépenses.

Quelles sont les principales améliorations de performances de Kubecost v3 ?

Kubecost v3 introduit des améliorations de performances substantielles grâce à son backend de ClickHouse base de données, qui fournit des requêtes nettement plus rapides par rapport au DuckDB utilisé dans la version 2.8. En outre, l'architecture unifiée des agents élimine la dépendance à Prometheus, réduisant ainsi l'encombrement de la mémoire tout en préservant l'intégralité des fonctionnalités et de la compatibilité. OpenCost

Quel est le backend de stockage utilisé par Kubecost v3 pour les déploiements multi-clusters ?

Kubecost v3 utilise un stockage d'objets compatible AWS S3 (S3 pour les clients EKS) pour le stockage des métriques multi-clusters, en remplacement du stockage compatible Prometheus utilisé dans la version v2. L' FinOps agent collecte des métriques à partir de l'API Kubernetes et les transmet vers un stockage compatible S3. L'agrégateur récupère ensuite ces données, effectue des calculs de coûts et affiche les résultats dans le frontend. Pour obtenir des instructions détaillées sur la configuration de plusieurs clusters, consultez le guide d'installation de plusieurs clusters et de clusters secondaires dans la documentation de Kubecost.

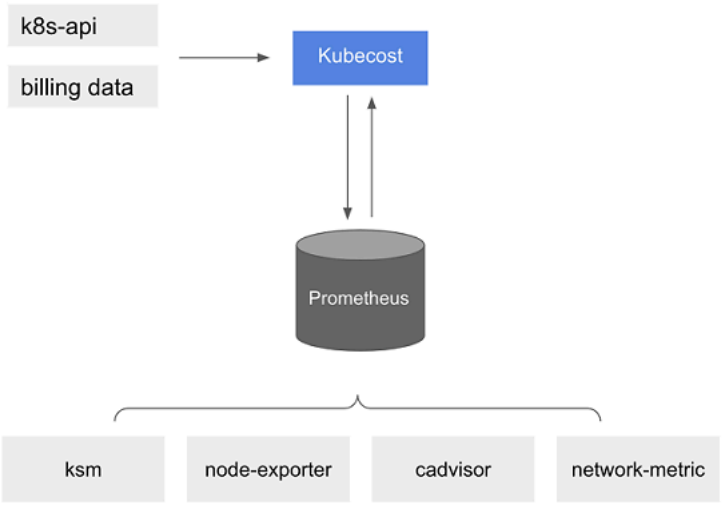
Puis-je passer directement de Kubecost v1 à v3 ?

Non La mise à niveau directe de la v1 vers la v3 n'est pas prise en charge. Vous devez d'abord passer à la version v2, puis migrer vers la version 3. Consultez la documentation de Kubecost pour

obtenir des conseils de migration détaillés, car le processus nécessite une planification minutieuse et peut avoir un impact sur la disponibilité des rapports pendant la transition.

Fonctionnalités supplémentaires de Kubecost

- Les fonctionnalités suivantes sont disponibles dans Kubecost v1, v2 et v3.
  - Indicateurs des coûts d'exportation — La surveillance optimisée des coûts d'Amazon EKS est déployée avec Kubecost. Dans les versions 1 et 2, Kubecost s'intègre à Prometheus pour le stockage et le traitement des métriques. Dans la version 3, Kubecost utilise une ClickHouse base de données pour améliorer considérablement les performances tout en préservant la compatibilité. OpenCost Pour les déploiements multi-clusters dans la version 3, les métriques sont stockées dans un stockage d'objets compatible AWS S3 (S3 pour les clients EKS) au lieu d'un stockage compatible avec Prometheus. Kubecost lit les métriques, effectue des calculs de répartition des coûts et fournit des données via son interface APIs utilisateur. L'architecture varie selon les versions mais conserve des fonctionnalités cohérentes.



Vous pouvez rédiger des requêtes pour intégrer les données Kubecost dans votre système de business intelligence actuel pour une analyse plus approfondie. Vous pouvez également l'utiliser comme source de données pour votre tableau de bord Grafana actuel afin d'afficher les coûts du cluster Amazon EKS que vos équipes internes connaissent bien. Pour en savoir plus sur la façon d'écrire des requêtes, consultez la documentation de OpenCost configuration ou utilisez les exemples de modèles Grafana JSON dans le référentiel Kubecost Github comme références.

- AWS Intégration des rapports sur les coûts et l'utilisation : pour effectuer des calculs de répartition des coûts pour votre cluster Amazon EKS, Kubecost extrait les informations de tarification publiques des AWS services et des AWS ressources à partir de l'API AWS Price

List. Vous pouvez également intégrer Kubecost au rapport sur les AWS coûts et l'utilisation pour améliorer la précision des informations de tarification spécifiques à votre AWS compte. Ces informations incluent les programmes de remise pour les entreprises, l'utilisation d'instances réservées, les plans d'épargne et l'utilisation ponctuelle. Pour en savoir plus sur le fonctionnement de l'intégration des rapports sur les AWS coûts et l'utilisation, consultez la section Intégration de la facturation dans le AWS cloud dans la documentation de Kubecost.

Afficher l'utilisation des ressources avec le serveur de métriques Kubernetes

Le serveur de métriques Kubernetes est un agrégateur de données d'utilisation des ressources dans votre cluster. Il n'est pas déployé par défaut dans les clusters Amazon EKS. Pour plus d'informations, consultez Kubernetes Metrics Server sur. GitHub Le serveur Metrics est couramment utilisé par d'autres modules complémentaires Kubernetes, tels que les déploiements de pods à l'échelle avec Horizontal Pod Autoscaler ou le tableau de bord Kubernetes. Pour plus d'informations, consultez Pipeline de métriques de ressource dans la documentation Kubernetes. Cette rubrique explique comment déployer le serveur de métriques Kubernetes sur votre cluster Amazon EKS.

> **Important**
>
> Les métriques sont destinées à point-in-time l'analyse et ne constituent pas une source précise pour l'analyse historique. Elles ne peuvent pas être utilisées comme solution de surveillance ou à d'autres fins de mise à l'échelle non auto. Pour de plus amples informations sur la surveillance, consultez Grappes de moniteurs.

Considérations

- Si vous déployez manuellement Kubernetes Metrics Server sur des nœuds Fargate à l'aide du manifeste, configurez le déploiement `metrics-server` pour qu'il utilise un port autre que le port par défaut `10250`. Ce port est réservé à Fargate. La version complémentaire Amazon EKS de Metrics Server est préconfigurée pour utiliser le port `10251`.
- Assurez-vous que les groupes de sécurité et le réseau ACLs autorisent le port `10250` entre les `metrics-server` pods et tous les autres nœuds et pods. Le serveur de métriques Kubernetes utilise toujours le port `10250` pour collecter les métriques provenant d'autres points de terminaison

du cluster. Si vous déployez sur des nœuds Fargate, autorisez à la fois le port alternatif configuré pour le serveur de métriques et le port `10250`.

Déployer en tant que module complémentaire communautaire avec Amazon EKS Add-ons

Nouveau : vous pouvez désormais déployer Metrics Server en tant que module complémentaire communautaire à l'aide de la AWS console ou d'Amazon EKS APIs.

Déploiement avec AWS console

1. Ouvrez votre cluster EKS dans la AWS console
2. Dans l'onglet « Modules complémentaires », sélectionnez Obtenir plus de modules complémentaires.
3. Dans la section « Extensions communautaires », sélectionnez Metrics Server, puis Suivant
4. EKS détermine la version appropriée du module complémentaire correspondant à votre cluster. Vous pouvez modifier la version à l'aide du menu déroulant Version.
5. Sélectionnez Suivant, puis Créer pour installer le module complémentaire.

Ressources supplémentaires

En savoir plus sur the section called "Extensions communautaires".

L'installation ou la mise à jour des modules complémentaires communautaires se fait de la même manière que pour les autres modules complémentaires Amazon EKS.

- the section called "Créer un module complémentaire"
- the section called "Mettre à jour un module complémentaire"
- the section called "Supprimer un module complémentaire"

Déployer avec Manifeste

Nouveau : vous pouvez désormais déployer Metrics Server en tant que module complémentaire communautaire à l'aide de la AWS console ou d'Amazon EKS APIs. Ces instructions d'installation du manifeste seront archivées.

1. Déployez le serveur de métriques à l'aide de la commande suivante :

```
kubectl apply -f https://github.com/kubernetes-sigs/metrics-server/releases/latest/
download/components.yaml
```

   Si vous utilisez Fargate, vous devez modifier ce fichier. Dans la configuration par défaut, le serveur de métriques utilise le port 10250. Ce port est réservé sur Fargate. Remplacez les références au port 10250 dans components.yaml par un autre port, tel que 10251.

2. Vérifiez que le déploiement du `metrics-server` exécute le nombre souhaité de pods avec la commande suivante.

```
kubectl get deployment metrics-server -n kube-system
```

   L'exemple qui suit illustre un résultat.

```
NAME             READY   UP-TO-DATE   AVAILABLE   AGE
metrics-server   1/1     1            1           6m
```

3. Testez le fonctionnement du serveur de métriques en affichant l'utilisation des ressources (CPU/mémoire) des nœuds.

```
kubectl top nodes
```

4. Si vous recevez le message d'erreur `Error from server (Forbidden)`, vous devez mettre à jour votre configuration RBAC Kubernetes. Votre identité Kubernetes RBAC nécessite des autorisations suffisantes pour lire les métriques du cluster. Vérifiez les autorisations d'API Kubernetes minimales requises pour lire les métriques sur. GitHub Découvrez comment accorder aux identités AWS IAM, telles que les rôles, l'accès à APIs Kubernetes.

Déployez des applications avec Helm sur Amazon EKS

Le gestionnaire de paquets Helm pour Kubernetes vous aide à installer et à gérer des applications sur votre cluster Kubernetes. Pour plus d'informations, consultez la documentation Helm. Cette rubrique vous aide à installer et à exécuter les fichiers binaires Helm, afin de pouvoir installer et gérer des charts à l'aide de la CLI Helm sur votre système local.

> **Important**
>
> Pour pouvoir installer des Charts Helm dans votre cluster, vous devez configurer `kubectl` pour qu'il fonctionne avec Amazon EKS. Si vous ne l'avez pas encore fait, veuillez consulter the section called "Accès au cluster avec kubectl" avant de poursuivre. Si la commande suivante aboutit pour votre cluster, votre configuration est correcte.
>
> ```
> kubectl get svc
> ```

1. Exécutez la commande adaptée à votre système d'exploitation client.
   - Si vous utilisez macOS avec Homebrew, installez les fichiers binaires à l'aide de la commande suivante.

     ```
     brew install helm
     ```
   - Pour plus d'options d'installation, consultez la section Installation de Helm dans la documentation Helm.

     > **Note**
     >
     > Si vous recevez un message indiquant que `openssl` doit d'abord être installé, vous pouvez utiliser la commande suivante pour l'installer.

     ```
     sudo yum install openssl
     ```

1. Pour récupérer le nouveau fichier binaire dans votre `PATH`, fermez votre fenêtre de terminal actuelle et ouvrez une nouvelle fenêtre.
2. Vérifiez la version de Helm que vous avez installée.

   ```
   helm version --template='{{ .Version }}{{ "\n" }}'
   ```

   L'exemple qui suit illustre un résultat.

   ```
   v3.17.2
   ```

3. Assurez-vous que la version installée est compatible avec la version de votre cluster. Consultez Supported Version Skew pour en savoir plus. Par exemple, si vous utilisez `3.17.x`, la version Kubernetes prise en charge ne doit pas dépasser la plage `1.29.x ~ 1.32.x`.
4. À ce stade, vous pouvez exécuter toutes les commandes Helm (telles que `helm install` *`chart-name`*) pour installer, modifier, supprimer ou interroger les charts Helm dans votre cluster. Si vous débutez avec Helm et que vous n'avez pas de graphique spécifique à installer, vous pouvez :
   - Tester en installant un exemple de chart. Reportez-vous à la rubrique sur l'installation d'un exemple de chart dans le Guide Quickstart Helm.
   - Créez un exemple de chart et envoyez-le à Amazon ECR. Pour plus d'informations, consultez la rubrique relative à l'envoi d'un chart Helm dans le Guide de l'utilisateur Amazon Elastic Container Registry.
   - Installez un graphique Amazon EKS à partir du référentiel GitHub eks-charts ou ArtifactHub.

Organiser les ressources Amazon EKS à l'aide de balises

Vous pouvez utiliser les balises pour vous aider à gérer vos ressources Amazon EKS. Cette rubrique fournit une vue d'ensemble de la fonction des identifications et indique comment créer des identifications.

Rubriques

- Principes de base des étiquettes
- Balisage de vos ressources
- Restrictions liées aux étiquettes
- Balisage de vos ressources pour facturation
- Gestion des étiquettes à l'aide de la console
- Gestion des identifications à l'aide de la CLI, de l'API ou de eksctl

> **Note**
>
> Les balises sont un type de métadonnées distinct des étiquettes et annotations Kubernetes. Pour plus d'informations sur ces autres types de métadonnées, consultez les sections suivantes dans la documentation Kubernetes :

- Étiquettes et sélecteurs
- Annotations

Principes de base des étiquettes

Une balise est une étiquette que vous affectez à une ressource AWS. Chaque balise est constituée d'une clé et d'une valeur facultative.

Avec les balises, vous pouvez classer vos ressources AWS par catégorie. Par exemple, vous pouvez classer les ressources par objectif, propriétaire ou environnement. Lorsque vous avez de nombreuses ressources de même type, vous pouvez utiliser les balises que vous avez attribuées à une ressource spécifique pour identifier cette dernière rapidement. Par exemple, vous pouvez définir un ensemble de balises pour vos clusters Amazon EKS afin de vous aider à suivre le propriétaire et le niveau de pile de chaque cluster. Nous vous recommandons de concevoir un ensemble cohérent de clés de balise pour chaque type de ressource. Vous pouvez rechercher et filtrer les ressources en fonction des balises que vous ajoutez.

Une fois que vous avez ajouté une balise, vous pouvez modifier les clés et valeurs de balise ou supprimer les balises d'une ressource à tout moment. Si vous supprimez une ressource, ses balises sont également supprimées.

Les identifications n'ont pas de signification sémantique pour Amazon EKS et sont interprétées strictement comme des chaînes de caractères. Vous pouvez définir la valeur d'une balise à une chaîne vide. Cependant, vous ne pouvez pas définir la valeur d'une balise sur null. Si vous ajoutez une balise ayant la même clé qu'une balise existante sur cette ressource, la nouvelle valeur remplace l'ancienne valeur.

Si vous utilisez la gestion des identités et des accès (IAM) AWS, vous pouvez contrôler quels utilisateurs de votre compte AWS sont autorisés à gérer les balises.

Balisage de vos ressources

Les ressources Amazon EKS suivantes prennent en charge les balises :

- clusters
- groupes de nœuds gérés
- profils Fargate

Vous pouvez baliser ces ressources à l'aide des éléments suivants :

- Si vous utilisez la console Amazon EKS, vous pouvez à tout moment appliquer des identifications aux ressources nouvelles ou existantes. Pour ce faire, utilisez la commande Identifications sur la page de ressources correspondante. Pour de plus amples informations, consultez the section called "Gestion des étiquettes à l'aide de la console".
- Si vous utilisez `eksctl`, vous pouvez appliquer des balises aux ressources lors de leur création à l'aide de l'option `--tags`.
- Si vous utilisez la CLI AWS, l'API Amazon EKS ou un kit SDK AWS, vous pouvez appliquer les balises aux nouvelles ressources à l'aide du paramètre `tags` sur l'action d'API correspondante. Vous pouvez également appliquer des identifications aux ressources à l'aide de l'action d'API `TagResource`. Pour en savoir plus, consultez TagResource.

Lorsque vous exécutez certaines actions de création des ressources, vous pouvez également spécifier des balises pour la ressource en même temps que vous la créez. Si les balises ne peuvent pas être appliquées pendant la création de la ressource, la ressource ne peut pas être créée. Ce mécanisme garantit que les ressources que vous prévoyez de baliser sont créées avec les balises que vous spécifiez ou ne sont pas créées du tout. Si vous balisez des ressources lorsque vous les créez, vous n'avez pas besoin d'exécuter de scripts de balisage personnalisés après avoir créé les ressources.

Les balises ne sont pas propagées vers les autres ressources associées à la ressource que vous créez. Par exemple, les balises de profil Fargate ne se propagent pas aux autres ressources qui sont associées au profil Fargate, telles que les pods qui sont programmés avec ce dernier.

Restrictions liées aux étiquettes

Les restrictions suivantes s'appliquent aux balises :

- Un maximum de 50 balises peut être associé à une ressource.
- Les clés de balises ne peuvent pas être répétées pour une même ressource. Chaque clé de balise doit être unique et ne peut avoir qu'une seule valeur.
- Les clés peuvent contenir jusqu'à 128 caractères en UTF-8.
- Les valeurs peuvent contenir jusqu'à 256 caractères en UTF-8.
- Si plusieurs services AWS et ressources utilisent votre schéma de balisage, limitez les types de caractères que vous utilisez. Certains services peuvent présenter des restrictions sur les

caractères autorisés. Les caractères généralement autorisés sont les lettres, les chiffres, les espaces et les caractères suivants : `+ - = . _ : / @`.

- Les clés et valeurs de balise sont sensibles à la casse.
- N'utilisez pas `aws:`, `AWS:`, ou n'importe quelle combinaison de majuscules ou minuscules comme préfixe pour des clés ou des valeurs. Ils sont réservés uniquement à une utilisation par AWS. Vous ne pouvez pas modifier ni supprimer des clés ou valeurs d'étiquette ayant ce préfixe. Les balises avec ce préfixe ne sont pas comptabilisées comme vos balises pour la limite de ressources.

Balisage de vos ressources pour facturation

Lorsque vous appliquez des balises à des clusters Amazon EKS, vous pouvez les utiliser pour la répartition des coûts dans vos Cost & Usage Reports (Rapports sur les coûts et l'utilisation). Les données de mesure de vos rapports sur les coût et l'utilisation illustrent l'utilisation parmi tous vos clusters Amazon EKS. Pour plus d'informations, consultez le rapport sur les coûts et l'utilisation AWS dans le Guide de facturation AWS.

L'identification de répartition des coûts générée par AWS, en particulier `aws:eks:cluster-name`, vous permet de décomposer les coûts des instances Amazon EC2 par cluster Amazon EKS dans Cost Explorer. Cependant, cette balise ne prend pas en compte les dépenses liées au plan de contrôle. L'identification est automatiquement ajoutée aux instances Amazon EC2 qui participent à un cluster Amazon EKS. Ce comportement se produit indépendamment du fait que les instances soient allouées à l'aide de groupes de nœuds gérés par Amazon EKS, de Karpenter ou directement avec Amazon EC2. Cette identification n'est pas prise en compte dans la limite des 50 identifications. Pour utiliser l'identification, le détenteur d'un compte doit l'activer dans la console de facturation AWS ou à l'aide de l'API. Lorsque le titulaire d'un compte de gestion d'une organisation AWS active la balise, celui-ci est également activé pour tous les comptes des membres de l'organisation.

Vous pouvez également organiser vos informations de facturation en fonction des ressources possédant les mêmes valeurs de clé d'étiquette. Par exemple, vous pouvez étiqueter plusieurs ressources avec un nom d'application spécifique, puis organiser vos informations de facturation. De cette façon, vous pouvez afficher le coût total de cette application dans plusieurs services. Pour plus d'informations sur la configuration d'un rapport de répartition des coûts avec des balises, consultez la section Rapport mensuel de répartition des coûts dans le Guide de facturation AWS.

> **Note**
>
> Si vous venez d'activer la création de rapports, les données du mois en cours peuvent être consultées après 24 heures.

Cost Explorer est un outil de reporting disponible dans l'offre gratuite AWS. Vous pouvez utiliser Cost Explorer pour consulter les graphiques de vos ressources Amazon EKS des 13 derniers mois. Vous pouvez également prévoir vos dépenses sur les trois prochains mois. Vous pouvez afficher les schémas de vos dépenses en ressources AWS au fil du temps. Par exemple, vous pouvez l'utiliser pour identifier les zones qui méritent d'être approfondies et connaître les tendances que vous pouvez utiliser pour comprendre vos coûts. Vous pouvez également spécifier des plages de temps pour les données et afficher des données temporelles par jour ou par mois.

Gestion des étiquettes à l'aide de la console

À l'aide de la console Amazon EKS, vous pouvez gérer les identifications qui sont associées à des clusters et groupes de nœuds gérés nouveaux ou existants.

Lorsque vous sélectionnez une page spécifique aux ressources dans la console Amazon EKS, cette page affiche une liste de ces ressources. Par exemple, si vous sélectionnez Clusters dans le panneau de navigation de gauche, la console affiche une liste de clusters Amazon EKS. Lorsque vous sélectionnez une ressource de l'une de ces listes (par exemple, un cluster spécifique) qui prend en charge les balises, vous pouvez afficher et gérer ses balises dans l'onglet Tags (Balises).

Vous pouvez également utiliser Tag Editor dans la AWS Management Console, qui fournit un moyen unifié de gérer vos balises. Pour plus d'informations, consultez la rubrique Balisage de vos ressources AWS à l'aide de l'éditeur de balises du Guide de l'utilisateur de l'éditeur de balises AWS.

Ajout de balises lors de la création d'une ressource

Vous pouvez ajouter des identifications aux clusters, groupes de nœuds gérés et profils Fargate Amazon EKS lorsque vous les créez. Pour de plus amples informations, consultez the section called "Créer un cluster".

Ajout et suppression de balises sur une ressource

Vous pouvez ajouter ou supprimer les balises qui sont associées à vos clusters directement à partir de la page de la ressource.

1. Ouvrez la console Amazon EKS.
2. Dans la barre de navigation, sélectionnez la région AWS à utiliser.
3. Dans le volet de navigation de gauche, choisissez Clusters.
4. Choisissez un cluster spécifique.
5. Choisissez l'onglet Tags (Identifications), puis Manage tags (Gérer les identifications).
6. Sur la page Gérer les balises, ajoutez ou supprimez vos balises si nécessaire.
   - Pour ajouter une balise, choisissez Ajouter une balise. Puis spécifiez la clé et la valeur de chaque balise.
   - Pour supprimer une balise, sélectionnez Remove tag (Supprimer une balise).
7. Répétez cette procédure pour chaque balise que vous voulez ajouter ou supprimer.
8. Choisissez Update (Mettre à jour) pour terminer.

Gestion des identifications à l'aide de la CLI, de l'API ou de **`eksctl`**

Utilisez les commandes CLI AWS suivantes ou les opérations d'API Amazon EKS pour ajouter, mettre à jour, répertorier et supprimer les identifications de vos ressources. Vous pouvez uniquement utiliser `eksctl` pour ajouter des balises tout en créant simultanément les nouvelles ressources avec une seule commande.

Tâche	AWS CLI	AWS Tools for Windows PowerShell	Action d'API
Ajouter ou remplacer une ou plusieurs étiquettes.	tag-resource	Add-EKSResourceTag	TagResource
Supprimer une ou plusieurs étiquettes.	untag-resource	Remove-EKSResourceTag	UntagResource

Les exemples suivants montrent comment ajouter ou supprimer les étiquettes d'une ressource à l'aide de la CLI AWS.

Exemple 1 : Étiqueter un cluster existant

La commande suivante permet d'étiqueter un cluster existant.

```
aws eks tag-resource --resource-arn resource_ARN --tags team=devs
```

Exemple 2 : Supprimer les étiquettes d'un cluster existant

La commande suivante permet de supprimer une étiquette d'un cluster existant.

```
aws eks untag-resource --resource-arn resource_ARN --tag-keys tag_key
```

Exemple 3 : Afficher la liste des étiquettes d'une ressource

La commande suivante permet de répertorier l'ensemble des balises qui sont associées à une ressource existante.

```
aws eks list-tags-for-resource --resource-arn resource_ARN
```

Lorsque vous utilisez certaines actions de création de ressources, vous pouvez spécifier des balises en même temps que vous créez la ressource. Les actions suivantes prennent en charge la spécification d'une balise lorsque vous créez une ressource.

Tâche	AWS CLI	AWS Tools for Windows PowerShell	Action d'API	eksctl
Créer un cluster	create-cluster	New-EKSCluster	CreateCluster	`create cluster`
Créer un groupe de nœuds géré*	create-nodegroup	New-EKSNodegroup	CreateNodegroup	`create nodegroup`
Créer un profil Fargate	create-fargate-profile	New-EKSFargateProfile	CreateFargateProfile.html	`create fargatepr ofile`

- Si vous souhaitez également baliser des instances Amazon EC2 lorsque vous créez un groupe de nœuds gérés, créez le groupe de nœuds gérés à l'aide d'un modèle de lancement. Pour de plus amples informations, consultez the section called "Étiquetage des instances Amazon EC2". Si vos instances existent déjà, vous pouvez étiqueter manuellement les instances. Pour plus d'informations, consultez Balisage de vos ressources dans le Guide d'utilisateur Amazon EC2.

Afficher et gérer les quotas de service Amazon EKS et Fargate

Amazon EKS a intégré Service Quotas, un AWS service que vous pouvez utiliser pour consulter et gérer vos quotas depuis un emplacement central. Pour plus d'informations, veuillez consulter Qu'est-ce que Service Quotas? dans le Guide de l'utilisateur Service Quotas. Grâce à l'intégration des quotas de service, vous pouvez rapidement rechercher la valeur de vos quotas de service Amazon EKS et AWS Fargate à l'aide de la AWS Management Console CLI and. AWS

Consultez les quotas de service EKS dans le AWS Management Console

1. Ouvrez la console Service Quotas.
2. Dans le volet de navigation de gauche, sélectionnez AWS services.
3. Dans la liste des AWS services, recherchez et sélectionnez Amazon Elastic Kubernetes Service (Amazon EKS) ou Fargate. AWS

   Dans la liste des quotas de service, vous pouvez voir le nom du quota de service, la valeur appliquée (si elle est disponible), le quota AWS par défaut et si la valeur du quota est ajustable.
4. Pour afficher des informations supplémentaires sur un quota de service, notamment la description, choisissez le nom du quota.
5. (Facultatif) Pour demander une augmentation de quota, sélectionnez le quota que vous souhaitez augmenter, sélectionnez Request quota increase (Demander une augmentation de quota), saisissez ou sélectionnez les informations requises, puis sélectionnez Request (Demander).

Pour mieux utiliser les quotas de service à l'aide du AWS Management Console, consultez le Guide de l'utilisateur des quotas de service. Pour demander une augmentation de quota, consultez Demander une augmentation de quota dans le Guide de l'utilisateur de Service Quotas.

Afficher les quotas de service EKS avec la AWS CLI

Exécutez la commande suivante pour afficher vos quotas Amazon EKS.

```
aws service-quotas list-aws-default-service-quotas \
    --query 'Quotas[*].
{Adjustable:Adjustable,Name:QuotaName,Value:Value,Code:QuotaCode}' \
    --service-code eks \
    --output table
```

Exécutez la commande suivante pour afficher vos quotas Fargate.

```
aws service-quotas list-aws-default-service-quotas \
    --query 'Quotas[*].
{Adjustable:Adjustable,Name:QuotaName,Value:Value,Code:QuotaCode}' \
    --service-code fargate \
    --output table
```

> **Note**
>
> Le quota renvoyé est le nombre de tâches Amazon ECS ou de pods Amazon EKS qui peuvent être exécutés simultanément sur Fargate dans ce compte dans la région actuelle. AWS

Pour travailler davantage avec les quotas de service à l'aide de la AWS CLI, consultez service-quotas dans le manuel de référence des commandes de la AWS CLI. Pour demander une augmentation du quota, consultez la request-service-quota-increasecommande dans le manuel de référence des commandes de la AWS CLI.

Service quotas Amazon EKS

AWS recommande d'utiliser le AWS Management Console pour consulter vos quotas actuels. Pour de plus amples informations, veuillez consulter the section called “Consultez les quotas de service EKS dans le AWS Management Console”.

Pour consulter les quotas de service EKS par défaut, consultez la section Points de terminaison et quotas Amazon Elastic Kubernetes Service dans le manuel de référence général. AWS

Ces quotas de service sont répertoriés sous Amazon Elastic Kubernetes Service (Amazon EKS) dans la console Service Quotas. Pour demander une augmentation du quota pour les valeurs affichées comme ajustables, consultez Demande d'augmentation de quota dans le Guide de l'utilisateur Service Quotas.

> **Note**
>
> Les ajustements des composants suivants ne sont pas pris en charge dans Service Quotas : * Associations d’identité du pod par cluster. Pour les limites, voirthe section called “Pod Identity”. * CIDRs pour les réseaux de nœuds distants ou les réseaux de ports distants pour les nœuds hybrides. Pour les limites, voirthe section called “Nœuds hybrides”.

AWS Quotas de service Fargate

Le service AWS Fargate de la console Service Quotas répertorie plusieurs quotas de service. Vous pouvez configurer des alarmes qui vous alertent lorsque votre utilisation approche d'un quota de service. Pour de plus amples informations, veuillez consulter the section called "Création d'une CloudWatch alarme pour surveiller les métriques d'utilisation des ressources de Fargate".

AWS Les nouveaux comptes peuvent avoir des quotas initiaux plus faibles qui peuvent augmenter au fil du temps. Fargate surveille en permanence l'utilisation du compte dans AWS chaque région, puis augmente automatiquement les quotas en fonction de l'utilisation. Vous pouvez également demander une augmentation du quota pour les valeurs désignées comme ajustables. Pour de plus amples informations, consultez Demande d'augmentation de quota dans le Guide de l'utilisateur Service Quotas.

AWS recommande d'utiliser le AWS Management Console pour consulter vos quotas actuels. Pour de plus amples informations, veuillez consulter the section called "Consultez les quotas de service EKS dans le AWS Management Console".

Pour consulter les quotas de service AWS Fargate par défaut sur EKS, consultez la section Quotas de service Fargate dans le manuel de référence général. AWS

> **Note**
>
> Fargate applique en outre des quotas de taux de lancement pour les tâches Amazon ECS et les pods Amazon EKS. Pour plus d'informations, consultez Quotas de limitation AWS Fargate dans le Guide Amazon ECS.

Sécurité dans Amazon EKS

La sécurité du cloud AWS est la priorité absolue. En tant que AWS client, vous bénéficiez d'un centre de données et d'une architecture réseau conçus pour répondre aux exigences des entreprises les plus sensibles en matière de sécurité.

La sécurité est une responsabilité partagée entre vous AWS et vous. Le modèle de responsabilité partagée décrit cette notion par les termes sécurité du cloud et sécurité dans le cloud :

- Sécurité du cloud : AWS est chargée de protéger l'infrastructure qui exécute les AWS services dans le AWS cloud. Pour Amazon EKS, AWS est responsable du plan de contrôle Kubernetes, qui inclut les nœuds et la base de données du plan de contrôle. `etcd` Des auditeurs tiers testent et vérifient régulièrement l'efficacité de notre sécurité dans le cadre des programmes de conformitéAWS. Pour en savoir plus sur les programmes de conformité qui s'appliquent à Amazon EKS, consultez Services AWS concernés par le programme de conformité.
- Sécurité dans le cloud : votre responsabilité englobe les domaines suivants :
  - La configuration de sécurité du plan de données, y compris la configuration des groupes de sécurité qui autorisent le trafic à transmettre à partir du plan de contrôle Amazon EKS dans le VPC client.
  - La configuration des nœuds et les conteneurs eux-mêmes
  - Le système d'exploitation du nœud (y compris les mises à jour et les correctifs de sécurité)
  - D'autres logiciels d'application connexes :
    - Configuration et gestion des contrôles réseau, tels que les règles de pare-feu
    - La gestion de l'identité au niveau de la plateforme et la gestion des accès, avec ou en complément de l'IAM
  - La sensibilité de vos données, les exigences de votre entreprise,et la législation et la réglementation applicables

Amazon EKS est certifié par plusieurs programmes de conformité pour les applications réglementées et sensibles. Amazon EKS est conforme aux normes SOC, PCI, ISO, FedRAMP-Moderate, IRAP, C5, K-ISMS, ENS High, OSPAR, HITRUST CSF et est un service éligible HIPAA. Pour de plus amples informations, veuillez consulter Gestion de l'accès.

Cette documentation vous aide à comprendre comment appliquer le modèle de responsabilité partagée lors de l'utilisation d'Amazon EKS. Les rubriques suivantes vous montrent comment

configurer Amazon EKS pour répondre à vos objectifs de sécurité et de conformité. Vous apprendrez également à utiliser d'autres AWS services qui vous aident à surveiller et à sécuriser vos ressources Amazon EKS.

> **Note**
>
> Les conteneurs Linux sont constitués de groupes de contrôle (cgroups) et d'espaces de noms qui aident à limiter les accès auxquels un conteneur peut accéder, mais tous les conteneurs partagent le même noyau Linux que l'instance Amazon EC2 hôte. L'exécution d'un conteneur en tant qu'utilisateur racine (UID 0) ou l'octroi d'un accès à un conteneur aux ressources hôtes ou aux espaces de noms tels que le réseau hôte ou l'espace de noms PID hôte sont fortement déconseillés, car cela réduit l'efficacité de l'isolation fournie par les conteneurs.

Rubriques

- Sécuriser les clusters Amazon EKS avec les bonnes pratiques
- Analyse des vulnérabilités dans Amazon EKS
- Validation de la conformité des clusters Amazon EKS
- Considérations relatives à la sécurité pour Amazon Elastic Kubernetes Service
- Considérations de sécurité pour Kubernetes
- Considérations de sécurité relatives au mode automatique Amazon EKS
- Considérations relatives à la sécurité relatives aux fonctionnalités EKS
- Gestion des identités et des accès pour Amazon EKS

Sécuriser les clusters Amazon EKS avec les bonnes pratiques

Les bonnes pratiques de sécurité Amazon EKS se trouvent dans la section Bonnes pratiques en matière de sécurité du Guide des bonnes pratiques Amazon EKS.

Analyse des vulnérabilités dans Amazon EKS

La sécurité est une considération cruciale pour la configuration et la maintenance des clusters et des applications Kubernetes. Vous trouverez ci-dessous la liste des ressources vous permettant d'analyser la configuration de sécurité de vos clusters EKS, les outils pour vérifier les vulnérabilités, ainsi que les intégrations avec les services AWS qui peuvent effectuer cette analyse pour vous.

Référence Center for Internet Security (CIS) pour Amazon EKS

La Référence Kubernetes du Center for Internet Security (CIS) fournit des conseils sur les configurations de sécurité des nœuds Amazon EKS. Le benchmark :

- S'applique aux nœuds Amazon EC2 (gérés et autogérés) dans lesquels vous êtes responsable des configurations de sécurité des composants Kubernetes.
- Fournit un moyen standard approuvé par la communauté de vous assurer que vous avez configuré votre cluster et vos nœuds Kubernetes en toute sécurité lors de l'utilisation d'Amazon EKS.
- Se compose de quatre sections : configuration de journalisation du plan de contrôle, configurations de sécurité des nœuds, politiques et services managés.
- Prend en charge toutes les versions de Kubernetes actuellement disponibles dans Amazon EKS et peut être exécuté à l'aide de kube-bench, un outil open source standard pour vérifier la configuration à l'aide du benchmark CIS sur les clusters Kubernetes.

Pour en savoir plus, consultez Présentation de la norme CIS Amazon EKS.

Pour un pipeline `aws-sample` automatisé permettant de mettre à jour votre groupe de nœuds avec une image AMI conforme à la référence CIS, consultez Pipeline de renforcement des AMI optimisées EKS.

Versions de plateforme Amazon EKS

Les versions de la plateforme Amazon EKS représentent les fonctionnalités du plan de contrôle du cluster, y compris les indicateurs activés du serveur d'API Kubernetes ainsi que la version actuelle du correctif Kubernetes. Les nouveaux clusters sont déployés avec la dernière version de la plateforme. Pour plus de détails, consultez Versions de plateforme EKS.

Vous pouvez mettre à jour un cluster Amazon EKS vers de nouvelles versions Kubernetes. À mesure que de nouvelles versions Kubernetes deviennent disponibles dans Amazon EKS, nous vous recommandons de mettre à jour de manière proactive vos clusters pour utiliser la dernière version disponible. Pour plus d'informations sur les versions de Kubernetes prises en charge dans EKS, consultez Versions prises en charge par Amazon EKS.

Liste des vulnérabilités du système d'exploitation

Liste des vulnérabilités AL2023

Suivez les évènements de sécurité et de confidentialité pour Amazon Linux 2023 via le centre de sécurité Amazon Linux ou souscrivez au flux RSS associé. Les événements de sécurité et de confidentialité incluent une présentation du problème, des packages et des instructions relatives à la mise à jour de vos instances pour résoudre le problème.

Liste des vulnérabilités Amazon Linux 2

Suivez les évènements de sécurité et de confidentialité pour Amazon Linux 2 via le centre de sécurité Amazon Linux ou souscrivez au flux RSS associé. Les événements de sécurité et de confidentialité incluent une présentation du problème, des packages et des instructions relatives à la mise à jour de vos instances pour résoudre le problème.

Détection de nœuds avec Amazon Inspector

Vous pouvez utiliser Amazon Inspector pour vérifier l'accessibilité réseau indésirable de vos nœuds et les vulnérabilités de ces instances Amazon EC2.

Détection de clusters et de nœuds avec Amazon GuardDuty

Amazon GuardDuty est un service de détection des menaces qui vous aide à protéger vos comptes, vos conteneurs, vos charges de travail et les données de votre environnement AWS. Parmi d'autres fonctionnalités, GuardDuty offre les deux fonctionnalités suivantes qui détectent les menaces potentielles pour vos clusters EKS : Protection EKS et Surveillance de l'exécution.

Pour de plus amples informations, consultez the section called "Amazon GuardDuty".

Validation de la conformité des clusters Amazon EKS

Pour savoir si un service AWS est inclus dans le périmètre d'un programme de conformité spécifique, consultez Services AWS inclus dans le périmètre par programme de conformité, puis sélectionnez le programme de conformité qui vous intéresse. Pour obtenir des informations générales, consultez Programmes de conformité AWS.

Vous pouvez télécharger les rapports d'audit de tiers à l'aide de l'artefact AWS. Pour plus d'informations, consultez Téléchargement des rapports dans AWS Artifact.

Votre responsabilité en matière de conformité lorsque vous utilisez des services AWS dépend de la sensibilité de vos données, des objectifs de conformité de votre entreprise, ainsi que des lois et réglementations applicables. AWS fournit les ressources suivantes pour vous aider à respecter ces exigences de conformité :

- Conformité et gouvernance de la sécurité : ces guides de mise en œuvre de solutions traitent des considérations architecturales et fournissent les étapes à suivre afin de déployer des fonctionnalités de sécurité et de conformité.
- Référence des services éligibles HIPAA : liste les services éligibles HIPAA. Tous les services AWS ne sont pas admissibles à la norme HIPAA.
- Ressources de conformité AWS : cet ensemble de manuels et de guides peut s'appliquer à votre secteur et à votre emplacement.
- AWSGuides de conformité destinés aux clients — Comprenez le modèle de responsabilité partagée sous l'angle de la conformité. Les guides résument les meilleures pratiques pour sécuriser les services AWS et établissent une correspondance entre ces pratiques et les contrôles de sécurité définis par plusieurs cadres de référence, notamment l'Institut national de normalisation et de technologie (NIST), le Conseil de normes de sécurité PCI (Payment Card Industry) et l'Organisation internationale de normalisation (ISO).
- Évaluation des ressources à l'aide de règles dans le Guide du développeur AWS Config : le service AWS Config évalue dans quelle mesure vos configurations de ressources sont conformes aux pratiques internes, aux lignes directrices de l'industrie et aux réglementations.
- AWS Security Hub : ce service AWS fournit une vue complète de votre état de sécurité au sein d'AWS. Security Hub utilise des contrôles de sécurité pour évaluer vos ressources AWS et vérifier votre conformité par rapport aux normes et aux bonnes pratiques du secteur de la sécurité. Pour obtenir la liste des services et des contrôles pris en charge, consultez Référence des contrôles Security Hub.
- Amazon GuardDuty : ce service AWS permet de détecter les menaces potentielles pesant sur vos comptes AWS, vos charges de travail, vos conteneurs et vos données, en surveillant votre environnement à la recherche d'activités suspectes ou malveillantes. GuardDuty peut vous aider à satisfaire diverses exigences de conformité, comme la conformité à la norme PCI DSS, en répondant aux exigences de détection d'intrusion imposées par certains frameworks de conformité.
- AWS Audit Manager – Ce service AWS vous aide à auditer en permanence votre utilisation de AWS afin de simplifier la gestion des risques et la conformité aux réglementations et aux normes du secteur.

Considérations relatives à la sécurité pour Amazon Elastic Kubernetes Service

Voici quelques considérations relatives à la sécurité du cloud, car elles affectent Amazon EKS.

Rubriques

- Sécurité de l'infrastructure dans Amazon EKS
- Comprendre la résilience dans les clusters Amazon EKS
- Prévention des conflits entre services dans Amazon EKS

Sécurité de l'infrastructure dans Amazon EKS

En tant que service géré, Amazon Elastic Kubernetes Service bénéficie de la sécurité offerte par le réseau mondial AWS. Pour plus d'informations sur les services de sécurité AWS et la manière dont AWS protège l'infrastructure, consultez la section Sécurité du cloud AWS. Pour concevoir votre environnement AWS en utilisant les meilleures pratiques en matière de sécurité de l'infrastructure, consultez la section Protection de l'infrastructure dans le Security Pillar AWS Well-Architected Framework (Pilier de sécurité de l'infrastructure Well-Architected Framework).

Vous utilisez les appels d'API publiés par AWS pour accéder à Amazon EKS via le réseau. Les clients doivent prendre en charge les éléments suivants :

- Protocole TLS (Transport Layer Security). Nous exigeons TLS 1.2 et recommandons TLS 1.3.
- Ses suites de chiffrement PFS (Perfect Forward Secrecy) comme DHE (Ephemeral Diffie-Hellman) ou ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). La plupart des systèmes modernes tels que Java 7 et les versions ultérieures prennent en charge ces modes.

En outre, les demandes doivent être signées à l'aide d'un ID de clé d'accès et d'une clé d'accès secrète associée à un principal IAM. Vous pouvez également utiliser le Service de jetons de sécurité AWS (AWS STS) pour générer des informations d'identification de sécurité temporaires afin de signer les demandes.

Lorsque vous créez un cluster Amazon EKS, vous spécifiez les sous-réseaux VPC à utiliser par votre cluster. Amazon EKS nécessite des sous-réseaux dans au moins deux zones de disponibilité. Nous recommandons un VPC avec des sous-réseaux publics et privés afin que Kubernetes puisse créer

des équilibreurs de charge publics dans les sous-réseaux publics qui équilibrent le trafic vers des pods s'exécutant sur des nœuds qui se trouvent dans des sous-réseaux privés.

Pour plus d'informations sur les considérations VPC, consultez the section called "Exigences requises pour le VPC et les sous-réseaux".

Si vous créez votre VPC et vos groupes de nœuds à l'aide des modèles CloudFormation AWS fournis dans le Guide de démarrage Amazon EKS, votre plan de contrôle et vos groupes de sécurité de nœuds sont configurés avec nos paramètres recommandés.

Pour plus d'informations sur les considérations des groupes de sécurité, consultez the section called "Exigences de groupe de sécurité".

Lorsque vous créez un cluster, Amazon EKS crée un point de terminaison pour le serveur d'API Kubernetes géré que vous utilisez pour communiquer avec votre cluster (à l'aide d'outils de gestion Kubernetes comme `kubectl`). Par défaut, ce point de terminaison du serveur d'API est public sur Internet, et l'accès au serveur d'API est sécurisé à l'aide d'une combinaison de la gestion des identités et des accès AWS (IAM) et du contrôle d'accès basé sur les rôles (RBAC) natif de Kubernetes.

Vous pouvez activer l'accès privé au serveur d'API Kubernetes pour que toutes les communications entre vos nœuds et le serveur d'API restent au sein de votre VPC. Vous pouvez limiter les adresses IP qui peuvent accéder à votre serveur API à partir d'Internet, ou désactiver complètement l'accès Internet au serveur d'API.

Pour plus d'informations sur la modification de l'accès au point de terminaison de cluster, consultez the section called "Modification de l'accès au point de terminaison de cluster".

Vous pouvez mettre en œuvre les politiques réseau Kubernetes avec Amazon VPC CNI ou des outils tiers tels que Project Calico. Pour plus d'informations sur l'utilisation de l'Amazon VPC CNI pour les stratégies réseau, consultez the section called "Politiques Kubernetes". Project Calico tiers est un projet open source tierce. Pour plus d'informations, consultez la documentation Project Calico.

Accéder à Amazon EKS à l'aide d'AWS PrivateLink

Vous pouvez utiliser AWS PrivateLink pour créer une connexion privée entre votre VPC et Amazon Elastic Kubernetes Service. Vous pouvez accéder à Amazon EKS comme s'il se trouvait dans votre VPC, sans utiliser de passerelle Internet, de dispositif NAT, de connexion VPN ou de connexion AWS Direct Connect. Les instances de votre VPC n'ont pas besoin d'adresses IP publiques pour accéder à Amazon EKS.

Vous établissez cette connexion privée en créant un point de terminaison d'interface à technologie AWS PrivateLink. Nous créons une interface réseau de point de terminaison dans chaque sous-réseau que vous activez pour le point de terminaison d'interface. Il s'agit d'interfaces réseau gérées par le demandeur qui servent de point d'entrée pour le trafic destiné à Amazon EKS.

Pour plus d'informations, consultez Accéder aux services AWS via AWS PrivateLink dans le Guide AWS PrivateLink.

Avant de commencer

Avant de commencer, assurez-vous d'avoir effectué les tâches suivantes :

- Consultez la section Accéder à un service AWS à l'aide d'un point de terminaison d'un VPC d'interface dans le Guide AWS PrivateLink

Considérations

- Assistance et limitations : les points de terminaison d'interface Amazon EKS permettent un accès sécurisé à toutes les actions de l'API Amazon EKS à partir de votre VPC, mais ils sont soumis à certaines limitations : ils ne prennent pas en charge l'accès aux API Kubernetes, car celles-ci disposent d'un point de terminaison privé distinct. Vous ne pouvez pas configurer Amazon EKS pour qu'il soit accessible uniquement via le point de terminaison d'interface.
- Tarification : l'utilisation de points de terminaison d'interface pour Amazon EKS entraîne des frais AWS PrivateLink standard : frais horaires pour chaque point de terminaison provisionné dans chaque zone de disponibilité, frais de traitement des données pour le trafic passant par le point de terminaison. Pour en savoir plus, consultez Tarification AWS PrivateLink.
- Sécurité et contrôle d'accès : nous vous recommandons de renforcer la sécurité et de contrôler l'accès à l'aide de ces configurations supplémentaires : utilisez les politiques de point de terminaison d'un VPC pour contrôler l'accès à Amazon EKS via le point de terminaison d'interface, associez des groupes de sécurité aux interfaces réseau des points de terminaison pour gérer le trafic, utilisez les journaux de flux VPC pour capturer et surveiller le trafic IP vers et depuis les points de terminaison d'interface, avec des journaux pouvant être diffusés sur Amazon CloudWatch ou Amazon S3. Pour en savoir plus, consultez Contrôle de l'accès aux points de terminaison d'un VPC à l'aide des politiques de point de terminaison et Journalisation du trafic IP à l'aide des journaux de flux VPC.
- Options de connectivité : les points de terminaison d'interface offrent des options de connectivité flexibles à l'aide de l'accès sur site (connectez votre centre de données sur site à un VPC avec le

point de terminaison d'interface à l'aide d'AWS Direct Connect ou d'AWS Site-to-Site VPN) ou via la connectivité inter-VPC (utilisez AWS Transit Gateway ou le peering VPC pour connecter d'autres VPC au VPC avec le point de terminaison d'interface, en conservant le trafic au sein du réseau AWS). .

- Prise en charge des versions IP : les points de terminaison créés avant août 2024 prennent uniquement en charge IPv4 à l'aide de eks.region.amazonaws.com. Les nouveaux points de terminaison créés après août 2024 prennent en charge la double pile IPv4 et IPv6 (par exemple, eks.region.amazonaws.com, eks.region.api.aws).
- Disponibilité régionale : AWS PrivateLink pour l'API EKS n'est pas disponible dans les régions Asie-Pacifique (Malaisie) (ap-southeast-5), Asie-Pacifique (Thaïlande) (ap-southeast-7), Mexique (Centre) (mx-central-1) et Asie-Pacifique (Taipei) (ap-east-2). AWS PrivateLink pour eks-auth (Identité du pod Amazon EKS) est disponible dans la région Asie-Pacifique (Malaisie) (ap-southeast-5).

Créer un point de terminaison d'interface pour Amazon EKS

Vous pouvez créer un point de terminaison d'un VPC pour Amazon EKS à l'aide de la console Amazon VPC ou de l'interface de ligne de commande AWS (AWS CLI). Pour plus d'informations, consultez Créer un point de terminaison d'un VPC dans le Guide AWS PrivateLink.

Créez un point de terminaison d'interface pour Amazon EKS à l'aide des noms de service suivants :

API EKS

- com.amazonaws.region-code.eks
- com.amazonaws.code-région.eks-fips (pour les points de terminaison conformes à la norme FIPS)

API EKS Auth (Identité du pod EKS)

- com.amazonaws.region-code.eks-auth

Fonctionnalité DNS privée pour les points de terminaison d'interface Amazon EKS

La fonctionnalité DNS privée, activée par défaut pour les points de terminaison d'interface d'Amazon EKS et d'autres services AWS, facilite les demandes API sécurisées et privées à l'aide des noms DNS régionaux par défaut. Cette fonctionnalité garantit que les appels d'API sont routés via le point de terminaison d'interface sur le réseau AWS privé, ce qui améliore la sécurité et les performances.

La fonctionnalité DNS privés s'active automatiquement lorsque vous créez un point de terminaison d'interface pour Amazon EKS ou d'autres services AWS. Pour l'activer, vous devez configurer correctement votre VPC en définissant des attributs spécifiques :

- enableDnsHostnames : permet aux instances au sein du VPC d'avoir des noms d'hôte DNS.
- enableDnsSupport : active la résolution DNS dans l'ensemble du VPC.

Pour obtenir des instructions détaillées sur la vérification ou la modification de ces paramètres, consultez Afficher et mettre à jour les attributs DNS de votre VPC.

Noms DNS et types d'adresses IP

Lorsque la fonctionnalité DNS privés est activée, vous pouvez utiliser des noms DNS spécifiques pour vous connecter à Amazon EKS. Ces options évoluent au fil du temps :

- eks.region.amazonaws.com : le nom DNS traditionnel, qui ne résout que les adresses IPv4 avant août 2024. Pour les points de terminaison existants mis à jour vers la double pile, ce nom résout à la fois les adresses IPv4 et IPv6.
- eks.region.api.aws : disponible pour les nouveaux points de terminaison créés après août 2024, ce nom DNS à double pile résout à la fois les adresses IPv4 et IPv6.

Après août 2024, les nouveaux points de terminaison d'interface seront fournis avec deux noms DNS, et vous pourrez opter pour le type d'adresse IP double pile. Pour les points de terminaison existants, la mise à jour vers la double pile modifie eks.region.amazonaws.com afin de prendre en charge à la fois IPv4 et IPv6.

Utilisation de la fonctionnalité DNS privés

Une fois configurée, la fonctionnalité DNS privés peut être intégrée à vos flux de travail, offrant les capacités suivantes :

- Requêtes API : utilisez les noms DNS régionaux par défaut, `eks.region.amazonaws.com` ou `eks.region.api.aws`, en fonction de la configuration de votre point de terminaison pour effectuer des requêtes API vers Amazon EKS.
- Compatibilité des applications : vos applications existantes qui appellent les API EKS ne nécessitent aucune modification pour tirer parti de cette fonctionnalité.

- AWS CLI avec double pile : pour utiliser les points de terminaison à double pile avec l'AWS CLI, consultez la configuration Points de terminaison à double pile et FIPS dans le Guide de référence des SDK et outils AWS.
- Routage automatique : tout appel vers le point de terminaison de service par défaut d'Amazon EKS est automatiquement dirigé vers le point de terminaison de l'interface, garantissant ainsi une connectivité privée et sécurisée.

Comprendre la résilience dans les clusters Amazon EKS

L'infrastructure mondiale AWS s'articule autour de régions et de zones de disponibilité AWS. AWS Les régions fournissent plusieurs zones de disponibilité physiquement séparées et isolées, reliées par un réseau à latence faible, à haut débit et hautement redondant. Avec les zones de disponibilité, vous pouvez concevoir et exploiter des applications et des bases de données qui basculent automatiquement d'une zone de disponibilité à l'autre sans interruption. Les zones de disponibilité sont plus hautement disponibles, tolérantes aux pannes et évolutives que les infrastructures traditionnelles à un ou plusieurs centres de données.

Amazon EKS exécute et met à l'échelle des instances de plan de contrôle Kubernetes sur plusieurs zones de disponibilité AWS, ce qui permet d'assurer une haute disponibilité. Amazon EKS met automatiquement à l'échelle les instances du plan de contrôle en fonction de la charge, détecte et remplace les instances du plan de contrôle non saines, et applique automatiquement des correctifs au plan de contrôle. Après que vous ayez lancé une mise à jour de version, Amazon EKS met à jour votre plan de contrôle pour vous, en maintenant la haute disponibilité du plan de contrôle pendant la mise à jour.

Ce plan de contrôle se compose d'au moins deux instances de serveur API et trois instances `etcd` qui s'exécutent sur trois zones de disponibilité au sein d'une région AWS. Amazon EKS :

- Surveille activement la charge sur les instances de plan de contrôle et les met automatiquement à l'échelle pour garantir des performances élevées.
- Détecte et remplace automatiquement les instances de plan de contrôle défaillantes, en les redémarrant dans les zones de disponibilité de la région AWS si nécessaire.
- Exploite l'architecture des régions AWS afin de maintenir une haute disponibilité. De ce fait, Amazon EKS peut offrir un accord de niveau de service (SLA) pour la disponibilité des points de terminaison de serveur d'API.

Pour plus d'informations sur les régions et les zones de disponibilité AWS, consultez Infrastructure mondiale AWS.

Prévention des conflits entre services dans Amazon EKS

Le problème de député confus est un problème de sécurité dans lequel une entité qui n'est pas autorisée à effectuer une action peut contraindre une entité plus privilégiée à le faire. En AWS, l'usurpation d'identité interservices peut entraîner la confusion des adjoints. L'usurpation d'identité entre services peut se produire lorsqu'un service (le service appelant) appelle un autre service (le service appelé). Le service appelant peut être manipulé et ses autorisations utilisées pour agir sur les ressources d'un autre client auxquelles on ne serait pas autorisé d'accéder autrement. Pour éviter cela, AWS fournit des outils qui vous aident à protéger vos données pour tous les services auprès des principaux fournisseurs de services qui ont obtenu l'accès aux ressources de votre compte.

Nous vous recommandons d'utiliser les clés de contexte de condition globale `aws:SourceArn`, `aws:SourceAccount` dans les politiques de ressources afin de limiter les autorisations accordées par Amazon Elastic Kubernetes Service (Amazon EKS) à un autre service pour la ressource.

`aws:SourceArn`

Utilisez `aws:SourceArn` pour associer une seule ressource à l'accès interservice.

`aws:SourceAccount`

Utilisez `aws:SourceAccount` pour permettre à n'importe quelle ressource de ce compte d'être associée à l'utilisation interservice.

Le moyen le plus efficace de se protéger contre le problème de député confus consiste à utiliser la clé de contexte de condition globale `aws:SourceArn` avec l'ARN complet de la ressource. Si vous ne connaissez pas l'ARN complet de la ressource ou si vous spécifiez plusieurs ressources, utilisez la clé de condition de contexte global `aws:SourceArn` avec des caractères génériques (*) pour les parties inconnues de l'ARN. Par exemple, `arn:aws:<servicename>:*:<123456789012>:*`.

Si la valeur `aws:SourceArn` ne contient pas l'ID du compte, tel qu'un ARN de compartiment Amazon S3, vous devez utiliser à la fois `aws:SourceAccount` et `aws:SourceArn` pour limiter les autorisations.

Prévention des conflits entre services dans les rôles de cluster Amazon EKS

Un rôle IAM du cluster Amazon EKS est requis pour chaque cluster. Les clusters Kubernetes gérés par Amazon EKS utilisent ce rôle pour gérer les nœuds, tandis que l'ancien fournisseur de cloud utilise ce rôle pour créer des équilibreurs de charge avec Elastic Load Balancing pour les services. Ces actions de cluster ne peuvent affecter que le même compte. Nous vous recommandons donc de limiter chaque rôle de cluster à ce cluster et à ce compte. Il s'agit d'une application spécifique de la AWS recommandation de respecter le principe du moindre privilège dans votre compte.

Format source ARN

La valeur de `aws:SourceArn` doit être l'ARN d'un cluster EKS au format `arn:aws:eks:`*`region`*`:`*`account`*`:cluster/`*`cluster-name`*. Par exemple, `arn:aws: eks:us-west-2:123456789012:cluster/my-cluster`.

Format de stratégie de confiance pour les rôles de cluster EKS

L'exemple suivant montre comment vous pouvez utiliser les clés de contexte de condition globale `aws:SourceArn` et `aws:SourceAccount` dans Amazon EKS pour éviter le problème de l'adjoint confus.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "ArnLike": {
          "aws:SourceArn": "arn:aws:eks:us-west-2:123456789012:cluster/my-cluster"
          },
        "StringEquals": {
            "aws:SourceAccount": "123456789012"
        }
      }
    }
  ]
}
```

Considérations de sécurité pour Kubernetes

Voici quelques considérations relatives à la sécurité dans le cloud, car elles affectent Kubernetes dans les clusters Amazon EKS. Pour une analyse approfondie des contrôles et des pratiques de sécurité dans Kubernetes, consultez Sécurité native cloud et Kubernetes dans la documentation Kubernetes.

Rubriques

- Sécurisation des charges de travail grâce aux certificats Kubernetes
- Comprendre les rôles RBAC et les utilisateurs créés par Amazon EKS
- Chiffrer les secrets Kubernetes avec KMS sur les clusters existants
- Utilisation des secrets AWS Secrets Manager avec Amazon EKS Pods
- Chiffrement d'enveloppe par défaut pour toutes les données de l'API Kubernetes

Sécurisation des charges de travail grâce aux certificats Kubernetes

L'API Kubernetes Certificates automatise le provisionnement des informations d'identification X.509. Elle dispose d'une interface de ligne de commande permettant aux clients de l'API Kubernetes de demander et d'obtenir des certificats X.509 auprès d'une autorité de certification (CA). Vous pouvez utiliser la ressource `CertificateSigningRequest` (CSR) pour demander à un signataire désigné de signer le certificat. Vos demandes sont approuvées ou refusées avant d'être signées. Kubernetes prend en charge à la fois les signataires intégrés et les signataires personnalisés, avec des comportements clairement définis. De cette façon, les clients peuvent prédire ce qu'il advient de leurs CSR. Pour en savoir plus sur la signature des certificats, consultez les demandes de signature.

L'un des signataires intégrés est `kubernetes.io/legacy-unknown`. L'API `v1beta1` de la ressource CSR a honoré ce signataire d'héritage inconnu. Cependant, l'API `v1` stable de CSR ne permet pas de définir `signerName` sur `kubernetes.io/legacy-unknown`.

Si vous souhaitez utiliser Amazon EKS CA pour générer des certificats sur vos clusters, vous devez utiliser un signataire personnalisé. Pour utiliser la version de l'API CSR `v1` et générer un nouveau certificat, vous devez migrer tous les manifestes et clients API existants. Les certificats existants créés avec l'API `v1beta1` existante sont valides et fonctionnent jusqu'à l'expiration du certificat. Cela inclut les éléments suivants :

- Distribution d'approbation : aucune. Il n'existe pas d'approbation ou de distribution standard pour ce signataire dans un cluster Kubernetes.

- Sujets autorisés : tous
- Extensions x509 autorisées : honore SubjectAltName et les extensions d'utilisation des clés et rejette les autres extensions
- Utilisations de clés autorisées : ne doit pas inclure les utilisations au-delà de [« chiffrement de clé », « signature numérique », « authentification du serveur »]

> **Note**
>
> La signature de certificat client n'est pas prise en charge.

- Expiration/durée de vie du certificat : 1 an (par défaut et maximum)
- Bit CA autorisé/interdit : non autorisé

Exemple de génération CSR avec SignerName

Ces étapes montrent comment générer un certificat de service pour un nom DNS `myserver.default.svc` en utilisant `signerName: beta.eks.amazonaws.com/app-serving`. Utilisez-le comme guide pour votre propre environnement.

1. Exécutez la commande `openssl genrsa -out myserver.key 2048` pour générer une clé privée RSA.

```
openssl genrsa -out myserver.key 2048
```

2. Utilisez la commande suivante pour générer une demande de certificat.

```
openssl req -new -key myserver.key -out myserver.csr -subj "/CN=myserver.default.svc"
```

3. Générez une valeur `base64` pour la demande CSR et stockez-la dans une variable, car vous en aurez besoin lors d'une étape ultérieure.

```
base_64=$(cat myserver.csr | base64 -w 0 | tr -d "
")
```

4. Pour créer un fichier nommé `mycsr.yaml`, exécutez la commande suivante. Dans l'exemple suivant, `beta.eks.amazonaws.com/app-serving` est le `signerName`.

```
cat >mycsr.yaml <<EOF
apiVersion: certificates.k8s.io/v1
```

```
kind: CertificateSigningRequest
metadata:
  name: myserver
spec:
  request: $base_64
  signerName: beta.eks.amazonaws.com/app-serving
  usages:
    - digital signature
    - key encipherment
    - server auth
EOF
```

5. Envoyez la CSR.

```
kubectl apply -f mycsr.yaml
```

6. Approuvez le certificat de service.

```
kubectl certificate approve myserver
```

7. Vérifiez que le certificat a été émis.

```
kubectl get csr myserver
```

L'exemple qui suit illustre un résultat.

```
NAME       AGE     SIGNERNAME                              REQUESTOR          CONDITION
myserver   3m20s   beta.eks.amazonaws.com/app-serving   kubernetes-admin
 Approved,Issued
```

8. Exportez le certificat émis.

```
kubectl get csr myserver -o jsonpath='{.status.certificate}'| base64 -d >
 myserver.crt
```

Comprendre les rôles RBAC et les utilisateurs créés par Amazon EKS

Lorsque vous créez un cluster Kubernetes, plusieurs identités Kubernetes par défaut sont créées sur ce cluster afin d'assurer le bon fonctionnement de Kubernetes. Amazon EKS crée des identités Kubernetes pour chacun de ses composants par défaut. Les identités fournissent un contrôle

d'autorisation basé sur les rôles (RBAC) Kubernetes pour les composants du cluster. Pour plus d'informations, consultez Utilisation de l'autorisation RBAC dans la documentation Kubernetes.

Lorsque vous installez des modules complémentaires optionnels sur votre cluster, des identités Kubernetes supplémentaires peuvent être ajoutées à votre cluster. Pour plus d'informations sur les identités non abordées dans cette rubrique, consultez la documentation du module complémentaire.

Vous pouvez consulter la liste des identités Kubernetes créées par Amazon EKS sur votre cluster à l'aide de l'outil de ligne de `kubectl` commande AWS Management Console or. Toutes les identités des utilisateurs apparaissent dans les journaux `kube` d'audit mis à votre disposition via Amazon CloudWatch.

AWS Management Console

Prérequis

Le principal IAM que vous utilisez doit disposer des autorisations décrites dans la section Autorisations requises.

Pour consulter les identités créées par Amazon EKS à l'aide du AWS Management Console

1. Ouvrez la console Amazon EKS.
2. Dans la liste Clusters, sélectionnez le cluster qui contient les identités que vous souhaitez afficher.
3. Sélectionnez l'onglet Ressources.
4. Sous Resource types (Types de ressources), sélectionnez Authorization (Autorisation).
5. Choisissez, ClusterRolesClusterRoleBindings, Rôles ou RoleBindings. Toutes les ressources précédées de eks sont créées par Amazon EKS. Les ressources d'identité supplémentaires créées par Amazon EKS sont les suivantes :
   - Le ClusterRoleet ClusterRoleBindingnommé aws-node. Les ressources aws-node prennent en charge le plug-in Amazon VPC CNI pour Kubernetes, qu'Amazon EKS installe sur tous les clusters.
   - Un ClusterRolenom vpc-resource-controller-roleet un ClusterRoleBindingnom vpc-resource-controller-rolebinding. Ces ressources prennent en charge le contrôleur de ressources Amazon VPC, qu'Amazon EKS installe sur tous les clusters.

   Outre les ressources que vous voyez dans la console, les identités utilisateur spéciales suivantes existent sur votre cluster, bien qu'elles ne soient pas visibles dans la configuration du cluster :

- **`eks:cluster-bootstrap`** : utilisé pour les opérations `kubectl` pendant le démarrage du cluster.
- **`eks:support-engineer`** : utilisée pour les opérations de gestion des clusters.

6. Choisissez une ressource spécifique pour afficher les détails la concernant. Par défaut, les informations s'affichent en mode Structured view (Vue structurée). Dans le coin supérieur droit de la page de détails, vous pouvez sélectionner Raw view (Vue brute) pour afficher toutes les informations relative à la ressource.

Kubectl

Prérequis

L'entité que vous utilisez (AWS Identity and Access Management (IAM) ou OpenID Connect (OIDC)) pour répertorier les ressources Kubernetes du cluster doit être authentifiée par IAM ou par votre fournisseur d'identité OIDC. L'entité doit disposer des autorisations nécessaires pour utiliser Kubernetes `get` et les verbes `list` pour les ressources `Role`, `ClusterRole`, `RoleBinding`, et `ClusterRoleBinding` de votre cluster avec lesquelles vous souhaitez que l'entité fonctionne. Pour plus d'informations sur la façon dont vous pouvez autoriser votre cluster à accéder aux entités IAM, consultez the section called "Accès à l'API Kubernetes". Pour plus d'informations sur l'octroi à des entités authentifiées par votre propre fournisseur OIDC d'un accès à votre cluster, consultez the section called "Association d'un fournisseur OIDC externe".

Pour consulter les identités créées par Amazon EKS à l'aide de **`kubectl`**

Exécutez la commande correspondant au type de ressource que vous souhaitez consulter. Toutes les ressources renvoyées qui sont précédées de eks sont créées par Amazon EKS. Outre les ressources renvoyées dans la sortie des commandes, les identités d'utilisateur spéciales suivantes sont disponibles sur votre cluster, même si elles n'apparaissent pas dans la configuration du cluster :

- **`eks:cluster-bootstrap`** : utilisé pour les opérations `kubectl` pendant le démarrage du cluster.
- **`eks:support-engineer`** : utilisée pour les opérations de gestion des clusters.

ClusterRoles— `ClusterRoles` sont limités à votre cluster, de sorte que toute autorisation accordée à un rôle s'applique aux ressources de n'importe quel espace de noms Kubernetes du cluster.

La commande suivante renvoie tous les `ClusterRoles` Kubernetes créés par Amazon EKS sur votre cluster.

```
kubectl get clusterroles | grep eks
```

Outre les `ClusterRoles` renvoyés dans la sortie, les `ClusterRoles` suivants sont disponibles.

- **`aws-node`** : ce `ClusterRole` prend en charge le plug-in Amazon VPC CNI pour Kubernetes, qu'Amazon EKS installe sur tous les clusters.
- **`vpc-resource-controller-role`** : ce `ClusterRole` prend en charge le contrôleur de ressources Amazon VPC, qu'Amazon EKS installe sur tous les clusters.

Pour voir la spécification de a`ClusterRole`, remplacez *`eks:k8s-metrics`* la commande suivante par un `ClusterRole` renvoyé dans la sortie de la commande précédente. L'exemple suivant renvoie la spécification du *`eks:k8s-metrics`*`ClusterRole`.

```
kubectl describe clusterrole eks:k8s-metrics
```

L'exemple qui suit illustre un résultat.

```
Name:         eks:k8s-metrics
Labels:       <none>
Annotations:  <none>
PolicyRule:
  Resources         Non-Resource URLs  Resource Names  Verbs
  ---------         -----------------  --------------  -----
                    [/metrics]         []              [get]
  endpoints         []                 []              [list]
  nodes             []                 []              [list]
  pods              []                 []              [list]
  deployments.apps  []                 []              [list]
```

ClusterRoleBindings— `ClusterRoleBindings` sont limités à votre cluster.

La commande suivante renvoie tous les Kubernetes `ClusterRoleBindings` créés par Amazon EKS sur votre cluster.

```
kubectl get clusterrolebindings | grep eks
```

En plus des `ClusterRoleBindings` renvoyés dans la sortie, les `ClusterRoleBindings` suivants sont disponibles.

- **`aws-node`** : ce `ClusterRoleBinding` prend en charge le plug-in Amazon VPC CNI pour Kubernetes, qu'Amazon EKS installe sur tous les clusters.
- **`vpc-resource-controller-rolebinding`** : ce `ClusterRoleBinding` prend en charge le contrôleur de ressources Amazon VPC, qu'Amazon EKS installe sur tous les clusters.

Pour voir la spécification de a`ClusterRoleBinding`, remplacez *`eks:k8s-metrics`* la commande suivante par un `ClusterRoleBinding` renvoyé dans la sortie de la commande précédente. L'exemple suivant renvoie la spécification du *`eks:k8s-metrics`*`ClusterRoleBinding`.

```
kubectl describe clusterrolebinding eks:k8s-metrics
```

L'exemple qui suit illustre un résultat.

```
Name:         eks:k8s-metrics
Labels:       <none>
Annotations:  <none>
Role:
  Kind:  ClusterRole
  Name:  eks:k8s-metrics
Subjects:
  Kind  Name             Namespace
  ----  ----             ---------
  User  eks:k8s-metrics
```

Rôles : les `Roles` sont limités à un espace de noms Kubernetes. Tous les `Roles` créés par Amazon EKS sont limités à l'espace de noms `kube-system`.

La commande suivante renvoie tous les Kubernetes `Roles` créés par Amazon EKS sur votre cluster.

```
kubectl get roles -n kube-system | grep eks
```

Pour voir la spécification de a`Role`, remplacez *`eks:k8s-metrics`* la commande suivante par le nom de a `Role` renvoyé dans le résultat de la commande précédente. L'exemple suivant renvoie la spécification du *`eks:k8s-metrics`*`Role`.

```
kubectl describe role eks:k8s-metrics -n kube-system
```

L'exemple qui suit illustre un résultat.

```
Name:         eks:k8s-metrics
Labels:       <none>
Annotations:  <none>
PolicyRule:
  Resources         Non-Resource URLs  Resource Names              Verbs
  ---------         -----------------  --------------              -----
  daemonsets.apps   []                 [aws-node]                  [get]
  deployments.apps  []                 [vpc-resource-controller]  [get]
```

RoleBindings— `RoleBindings` sont limités à un espace de noms Kubernetes. Tous les `RoleBindings` créés par Amazon EKS sont limités à l'espace de noms `kube-system`.

La commande suivante renvoie tous les Kubernetes `RoleBindings` créés par Amazon EKS sur votre cluster.

```
kubectl get rolebindings -n kube-system | grep eks
```

Pour voir la spécification de a`RoleBinding`, remplacez *`eks:k8s-metrics`* la commande suivante par un `RoleBinding` renvoyé dans la sortie de la commande précédente. L'exemple suivant renvoie la spécification du *`eks:k8s-metrics`*`RoleBinding`.

```
kubectl describe rolebinding eks:k8s-metrics -n kube-system
```

L'exemple qui suit illustre un résultat.

```
Name:         eks:k8s-metrics
Labels:       <none>
Annotations:  <none>
Role:
  Kind:  Role
  Name:  eks:k8s-metrics
Subjects:
  Kind  Name             Namespace
  ----  ----             ---------
  User  eks:k8s-metrics
```

Chiffrer les secrets Kubernetes avec KMS sur les clusters existants

> **Important**
>
> Cette procédure s'applique uniquement aux clusters EKS exécutant Kubernetes version 1.27 ou antérieure. Si vous utilisez Kubernetes version 1.28 ou supérieure, vos secrets Kubernetes sont protégés par défaut par un chiffrement par enveloppe. Pour de plus amples informations, veuillez consulter the section called "Chiffrement d'enveloppe par défaut pour toutes les données de l'API Kubernetes".

Si vous activez le chiffrement des secrets, les secrets Kubernetes sont chiffrés à l'aide de la clé AWS KMS que vous sélectionnez. La clé KMS doit répondre aux conditions suivantes :

- Symétrique
- Peut chiffrer et déchiffrer des données
- Créé dans la même AWS région que le cluster
- Si la clé KMS a été créée dans un autre compte, le principal IAM doit avoir accès à la clé KMS.

Pour plus d'informations, consultez la section Autoriser les principaux IAM d'autres comptes à utiliser une clé KMS dans le Guide du développeur du service de gestion des AWS clés.

> **Warning**
>
> Vous ne pouvez pas désactiver le chiffrement des secrets après l'avoir activé. Cette action est irréversible.

eksctl

Cette procédure s'applique uniquement aux clusters EKS exécutant Kubernetes version 1.27 ou antérieure. Pour de plus amples informations, veuillez consulter the section called "Chiffrement d'enveloppe par défaut pour toutes les données de l'API Kubernetes".

Vous pouvez activer le chiffrement de deux manières :

- Ajoutez le chiffrement à votre cluster à l'aide d'une seule commande.

Pour re-chiffrer automatiquement vos secrets, exécutez la commande suivante.

```
eksctl utils enable-secrets-encryption \
    --cluster my-cluster \
    --key-arn arn:aws: kms:region-code:account:key/key
```

Pour vous désabonner du chiffrement automatique de vos secrets, exécutez la commande suivante.

```
eksctl utils enable-secrets-encryption
    --cluster my-cluster \
    --key-arn arn:aws: kms:region-code:account:key/key \
    --encrypt-existing-secrets=false
```

- Ajoutez le chiffrement à votre cluster à l'aide d'un fichier `kms-cluster.yaml`.

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-cluster
  region: region-code

secretsEncryption:
  keyARN: arn:aws: kms:region-code:account:key/key
```

Pour re-chiffrer automatiquement vos secrets, exécutez la commande suivante.

```
eksctl utils enable-secrets-encryption -f kms-cluster.yaml
```

Pour vous désabonner du chiffrement automatique de vos secrets, exécutez la commande suivante.

```
eksctl utils enable-secrets-encryption -f kms-cluster.yaml --encrypt-existing-
secrets=false
```

AWS Management Console

a. Cette procédure s'applique uniquement aux clusters EKS exécutant Kubernetes version 1.27 ou antérieure. Pour de plus amples informations, veuillez consulter the section called "Chiffrement d'enveloppe par défaut pour toutes les données de l'API Kubernetes".

b. Ouvrez la console Amazon EKS.

c. Choisissez le cluster auquel vous souhaitez ajouter le chiffrement KMS.

d. Cliquez sur l'onglet Présentation (cette option est sélectionnée par défaut).

e. Faites défiler la page jusqu'à Secrets encryption (Chiffrement des secrets) et sélectionnez Enable (Activer).

f. Sélectionnez une clé dans la liste déroulant et appuyez sur le bouton Enable (Activer). Si aucune clé n'est répertoriée, vous devez d'abord en créer une. Pour plus d'informations, consultez Création de clés.

g. Appuyez sur le bouton Confirm (Confirmer) pour utiliser la touche choisie.

AWS CLI

a. Cette procédure s'applique uniquement aux clusters EKS exécutant Kubernetes version 1.27 ou antérieure. Pour de plus amples informations, veuillez consulter the section called "Chiffrement d'enveloppe par défaut pour toutes les données de l'API Kubernetes".

b. Associez la configuration de chiffrement des secrets à votre cluster à l'aide de la commande AWS CLI suivante. Remplacez les exemples de valeurs par les vôtres.

```
aws eks associate-encryption-config \
    --cluster-name my-cluster \
    --encryption-config '[{"resources":["secrets"],"provider":
{"keyArn":"arn:aws: kms:region-code:account:key/key"}}]'
```

L'exemple qui suit illustre un résultat.

```
{
  "update": {
    "id": "3141b835-8103-423a-8e68-12c2521ffa4d",
    "status": "InProgress",
    "type": "AssociateEncryptionConfig",
    "params": [
      {
        "type": "EncryptionConfig",
```

```
        "value": "[{\"resources\":[\"secrets\"],\"provider\":{\"keyArn\":
\"arn:aws: kms:region-code:account:key/key\"}}]"
      }
    ],
    "createdAt": 1613754188.734,
    "errors": []
  }
}
```

c. Vous pouvez contrôler l'état de votre mise à jour de chiffrement à l'aide de la commande suivante. Utilisez le `cluster name` et le `update ID` spécifiques qui ont été renvoyés dans la sortie précédente. Lorsqu'un état `Successful` s'affiche, la mise à jour est terminée.

```
aws eks describe-update \
    --region region-code \
    --name my-cluster \
    --update-id 3141b835-8103-423a-8e68-12c2521ffa4d
```

L'exemple qui suit illustre un résultat.

```
{
  "update": {
    "id": "3141b835-8103-423a-8e68-12c2521ffa4d",
    "status": "Successful",
    "type": "AssociateEncryptionConfig",
    "params": [
      {
        "type": "EncryptionConfig",
        "value": "[{\"resources\":[\"secrets\"],\"provider\":{\"keyArn\":
\"arn:aws: kms:region-code:account:key/key\"}}]"
      }
    ],
    "createdAt": 1613754188.734>,
    "errors": []
  }
}
```

d. Pour vérifier que le chiffrement est activé dans votre cluster, exécutez la commande `describe-cluster`. La réponse contient une chaîne `EncryptionConfig`.

```
aws eks describe-cluster --region region-code --name my-cluster
```

Une fois que vous avez activé le chiffrement sur votre cluster, vous devez chiffrer tous les secrets existants avec la nouvelle clé :

> **Note**
>
> Si vous utilisez `eksctl`, l'exécution de la commande suivante n'est nécessaire que si vous vous désabonnez du chiffrement automatique de vos secrets.

```
kubectl get secrets --all-namespaces -o json | kubectl annotate --overwrite -f - kms-encryption-timestamp="time value"
```

> **Warning**
>
> Si vous activez le chiffrement de secrets pour un cluster existant et que la clé KMS que vous utilisez est supprimée, il n'y a pas de chemin vers la récupération pour le cluster. Si vous supprimez la clé KMS, vous mettez définitivement le cluster dans un état dégradé. Pour plus d'informations, consultez la section Suppression de clés AWS KMS.

> **Note**
>
> Par défaut, la `create-key` commande crée une clé KMS de chiffrement symétrique avec une politique de clé qui donne à l'administrateur racine du compte un accès aux actions et aux ressources AWS KMS. Si vous souhaitez limiter les autorisations, assurez-vous que les actions `kms:DescribeKey` et `kms:CreateGrant` sont autorisées sur la stratégie pour le principal qui appellera l'API `create-cluster`.
>
> Pour les clusters utilisant le chiffrement d'enveloppe KMS, des autorisations `kms:CreateGrant` sont requises. La condition n'`kms:GrantIsForAWSResource`est pas prise en charge pour l' CreateCluster action et ne doit pas être utilisée dans les politiques KMS pour contrôler `kms:CreateGrant` les autorisations accordées aux utilisateurs CreateCluster.

Utilisation des secrets AWS Secrets Manager avec Amazon EKS Pods

Pour afficher les secrets du gestionnaire de secrets et les paramètres du magasin de paramètres sous forme de fichiers montés dans les pods Amazon EKS, vous pouvez utiliser le fournisseur de secrets et de configuration AWS (ASCP) pour le pilote CSI du magasin de secrets Kubernetes.

Avec l'ASCP, vous pouvez stocker et gérer vos secrets dans Secrets Manager, puis les récupérer via vos applications exécutées sur Amazon EKS. Vous pouvez utiliser des rôles et des stratégies IAM pour limiter l'accès à vos secrets à des pods Kubernetes spécifiques dans un cluster. L'ASCP récupère l'identité du pod et l'échange contre un rôle IAM. ASCP assume le rôle IAM du pod, puis il peut récupérer les secrets de Secrets Manager autorisés pour ce rôle.

Si vous utilisez la rotation automatique de Secrets Manager pour vos secrets, vous pouvez également utiliser la fonction de réconciliation de rotation du pilote CSI de Secrets Store pour vous assurer que vous récupérez le dernier secret de Secrets Manager.

> **Note**
>
> Les groupes de nœuds AWS Fargate (Fargate) ne sont pas pris en charge.

Pour plus d'informations, consultez Utilisation des secrets de Secrets Manager dans Amazon EKS dans le Guide de l'utilisateur AWS Secrets Manager.

Chiffrement d'enveloppe par défaut pour toutes les données de l'API Kubernetes

Amazon Elastic Kubernetes Service (Amazon EKS) fournit un chiffrement par défaut pour toutes les données API Kubernetes dans les clusters EKS exécutant la version 1.28 ou supérieure de Kubernetes.

Le chiffrement des enveloppes protège les données que vous stockez avec le serveur d'API Kubernetes. Par exemple, le chiffrement d'enveloppe s'applique à la configuration de votre cluster Kubernetes, par exemple `ConfigMaps`. Le chiffrement des enveloppes ne s'applique pas aux données des nœuds ou des volumes EBS. EKS prenait déjà en charge le chiffrement des secrets Kubernetes, et désormais, ce chiffrement par enveloppe s'étend à toutes les données API Kubernetes.

Cela fournit une expérience gérée par défaut qui s'implémente defense-in-depth pour vos applications Kubernetes et ne nécessite aucune action de votre part.

Amazon EKS utilise le service de gestion des AWS clés (KMS) avec le fournisseur Kubernetes KMS v2 pour cette couche de sécurité supplémentaire avec une clé appartenant à Amazon Web Services et la possibilité pour vous d'apporter votre propre clé gérée par le client (CMK) auprès de KMS. AWS

Comprendre le chiffrement d'enveloppe

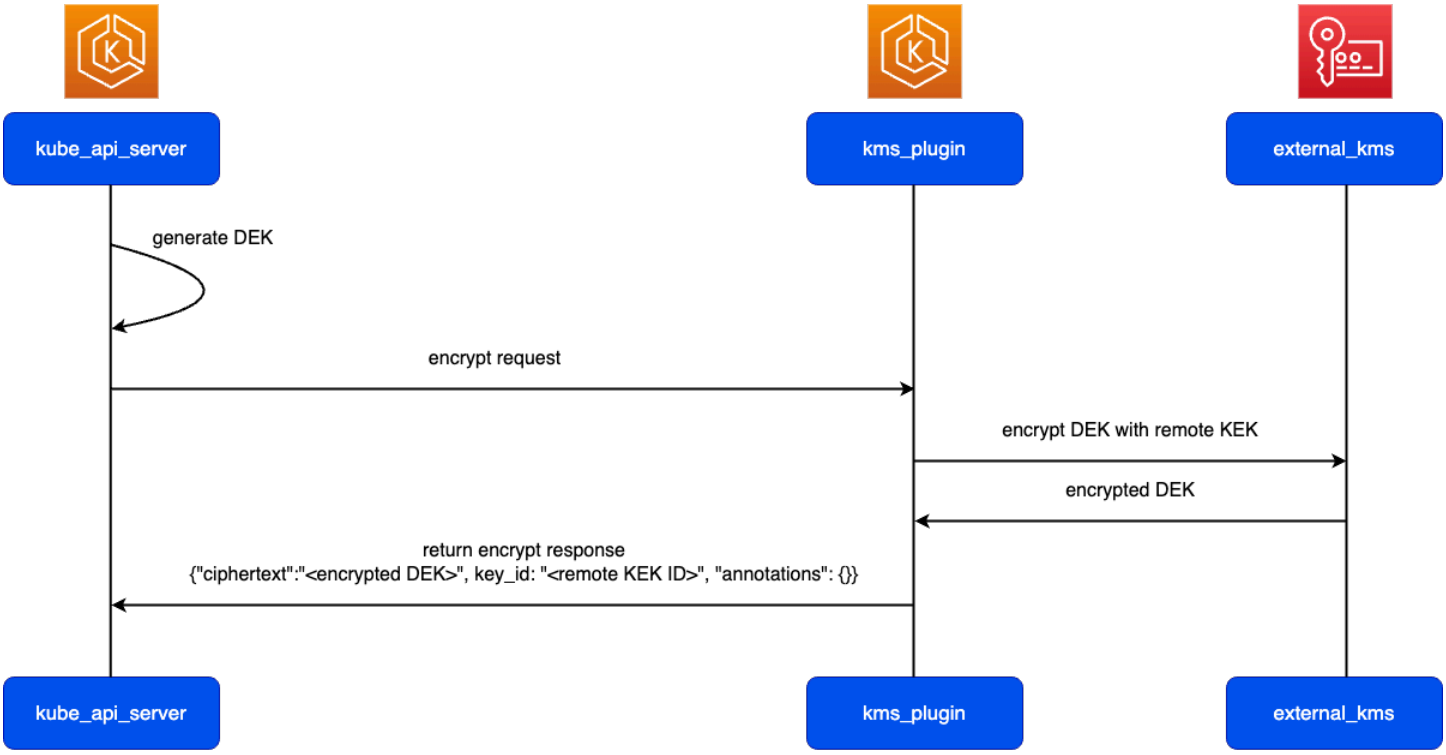
Le chiffrement d'enveloppe consiste à chiffrer des données en texte brut à l'aide d'une clé de chiffrement des données (DEK) avant leur envoi à la banque de données (etcd), puis à chiffrer la DEK avec une clé KMS racine stockée dans un système KMS (KMS) distant géré de manière centralisée (KMS).AWS Il s'agit d'une defense-in-depth stratégie car elle protège les données à l'aide d'une clé de chiffrement (DEK), puis ajoute une autre couche de sécurité en protégeant cette DEK avec une clé de chiffrement séparée et stockée de manière sécurisée appelée clé de chiffrement (KEK).

Comment Amazon EKS active le chiffrement des enveloppes par défaut avec KMS v2 et AWS KMS

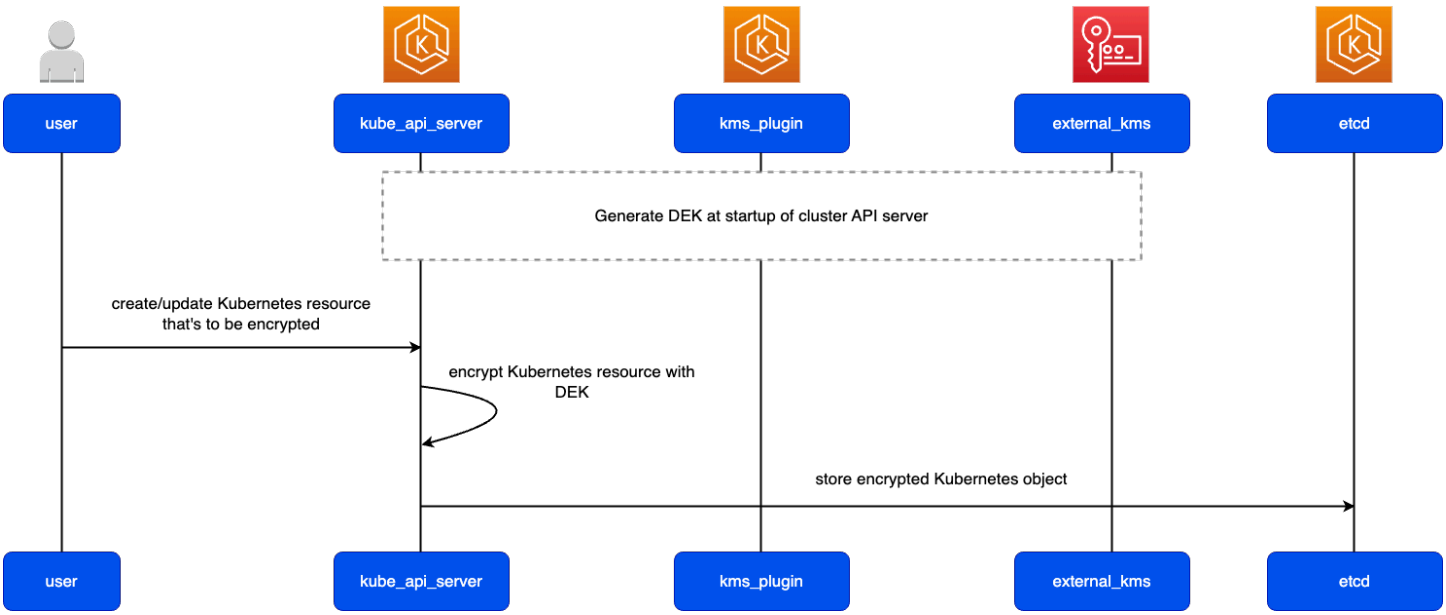
Amazon EKS utilise KMS v2 pour mettre en œuvre le chiffrement par défaut de toutes les données API dans le plan de contrôle Kubernetes géré avant leur conservation dans la base de données etcd. Au démarrage, le serveur API du cluster génère une clé de chiffrement des données (DEK) à partir d'une graine secrète combinée à des données générées aléatoirement. Au démarrage également, le serveur d'API appelle le plug-in KMS pour chiffrer le seed DEK à l'aide d'une clé de chiffrement par clé distante (KEK) fournie par AWS KMS. Il s'agit d'un appel ponctuel exécuté au démarrage du serveur API et lors de la rotation du KEK. Le serveur API met ensuite en cache la graine DEK chiffrée. Ensuite, le serveur API utilise la graine DEK mise en cache pour générer d'autres données à usage unique DEKs basées sur une fonction de dérivation de clés (KDF). Chacun de ces éléments générés n' DEKs est ensuite utilisé qu'une seule fois pour chiffrer une seule ressource Kubernetes avant qu'elle ne soit stockée dans etcd. Grâce à l'utilisation d'une graine DEK cryptée mise en cache dans KMS v2, le processus de cryptage des ressources Kubernetes dans le serveur API est à la fois plus performant et plus rentable.

Par défaut, ce KEK appartient à AWS KMS AWS, mais vous pouvez éventuellement apporter le vôtre.

Le schéma ci-dessous décrit la génération et le chiffrement d'un DEK au démarrage du serveur d'API.



Le schéma de haut niveau ci-dessous décrit le chiffrement d'une ressource Kubernetes avant son stockage dans etcd.



Questions fréquentes (FAQ)

Comment le chiffrement d'enveloppe par défaut améliore-t-il le niveau de sécurité de mon cluster EKS ?

Cette fonctionnalité réduit la surface et la durée pendant lesquelles les métadonnées et le contenu client ne sont pas cryptés. Avec le chiffrement par défaut des enveloppes, les métadonnées et le contenu client ne sont temporairement déchiffrés que dans la mémoire du kube-apiserver avant d'être stockés dans etcd. La mémoire du kube-apiserver est sécurisée via le système Nitro. Amazon EKS utilise uniquement des instances EC2 basées sur Nitro pour le plan de contrôle Kubernetes géré. Ces instances sont dotées de dispositifs de contrôle de sécurité qui empêchent tout système ou toute personne d'accéder à leur mémoire.

Quelle version de Kubernetes dois-je exécuter pour bénéficier de cette fonctionnalité ?

Pour que le chiffrement par défaut des enveloppes soit activé, votre cluster Amazon EKS doit exécuter Kubernetes version 1.28 ou ultérieure.

Mes données sont-elles toujours sécurisées si j'utilise une version de cluster Kubernetes qui ne prend pas en charge cette fonctionnalité ?

Oui. Chez AWS, la sécurité est notre priorité absolue. Nous basons toute notre transformation numérique et notre innovation sur les pratiques opérationnelles les plus strictes en matière de sécurité, et nous nous engageons à relever encore davantage la barre.

Toutes les données stockées dans etcd sont chiffrées au niveau du disque pour chaque cluster EKS, quelle que soit la version de Kubernetes utilisée. EKS utilise des clés racines qui génèrent des clés de chiffrement de volume gérées par le service EKS. De plus, chaque cluster Amazon EKS est exécuté dans un VPC isolé à l'aide de machines virtuelles spécifiques au cluster. Grâce à cette architecture et à nos pratiques en matière de sécurité opérationnelle, Amazon EKS a obtenu plusieurs certifications et normes de conformité, notamment SOC 1, 2 et 3, PCI-DSS, ISO et HIPAA. Ces notes et normes de conformité sont maintenues pour tous les clusters EKS, avec ou sans chiffrement d'enveloppe par défaut.

Comment fonctionne le chiffrement des enveloppes dans Amazon EKS ?

Au démarrage, le serveur API du cluster génère une clé de chiffrement des données (DEK) à partir d'une graine secrète combinée à des données générées aléatoirement. Au démarrage également, le serveur d'API appelle le plug-in KMS pour chiffrer le DEK à l'aide d'une clé de chiffrement à distance

(KEK) fournie par AWS KMS. Il s’agit d’un appel ponctuel exécuté au démarrage du serveur API et lors de la rotation du KEK. Le serveur API met ensuite en cache la graine DEK chiffrée. Ensuite, le serveur API utilise la graine DEK mise en cache pour générer d'autres données à usage unique DEKs basées sur une fonction de dérivation de clés (KDF). Chacun de ces éléments générés n' DEKs est ensuite utilisé qu'une seule fois pour chiffrer une seule ressource Kubernetes avant qu'elle ne soit stockée dans etcd.

Il est important de noter que des appels supplémentaires sont effectués depuis le serveur API pour vérifier le bon fonctionnement et le fonctionnement normal de l'intégration AWS KMS. Ces bilans de santé supplémentaires sont visibles dans votre AWS CloudTrail.

Dois-je effectuer une action ou modifier des autorisations pour que cette fonctionnalité fonctionne dans mon cluster EKS ?

Non, vous n’avez aucune démarche à effectuer. Le chiffrement des enveloppes dans Amazon EKS est désormais une configuration par défaut activée dans tous les clusters exécutant Kubernetes version 1.28 ou supérieure. L'intégration AWS KMS est établie par le serveur d'API Kubernetes géré par. AWS Cela signifie que vous n’avez pas besoin de configurer d’autorisations pour commencer à utiliser le chiffrement KMS pour votre cluster.

Comment savoir si le chiffrement d’enveloppe par défaut est activé sur mon cluster ?

Si vous migrez pour utiliser votre propre CMK, vous verrez alors l’ARN de la clé KMS associée à votre cluster. En outre, vous pouvez consulter les journaux AWS CloudTrail d'événements associés à l'utilisation de la clé CMK de votre cluster.

Si votre cluster utilise une clé AWS détenue, celle-ci sera détaillée dans la console EKS (à l'exception de l'ARN de la clé).

Puis-je AWS accéder à la clé AWS détenue utilisée pour le chiffrement des enveloppes par défaut dans Amazon EKS ?

Non AWS dispose de contrôles de sécurité stricts dans Amazon EKS qui empêchent toute personne d'accéder aux clés de chiffrement en texte brut utilisées pour sécuriser les données de la base de données etcd. Ces mesures de sécurité sont également appliquées à la clé KMS AWS détenue.

Le chiffrement d’enveloppe par défaut est-il activé dans mon cluster EKS existant ?

Si vous utilisez un cluster Amazon EKS avec la version 1.28 ou supérieure de Kubernetes, le chiffrement par enveloppe de toutes les données API Kubernetes est activé. Pour les clusters

existants, Amazon EKS utilise le `eks:kms-storage-migrator` RBAC ClusterRole pour faire migrer les données qui n'étaient pas auparavant chiffrées par enveloppe dans etcd vers ce nouvel état de chiffrement.

Qu'est-ce que cela signifie si j'ai déjà activé le chiffrement par enveloppe pour les secrets dans mon cluster EKS ?

Si vous disposez déjà d'une clé gérée par le client (CMK) dans KMS qui a été utilisée pour crypter vos secrets Kubernetes, cette même clé sera utilisée comme KEK pour le cryptage de tous les types de données API Kubernetes dans votre cluster.

Y a-t-il un coût supplémentaire pour l'exécution d'un cluster EKS avec le chiffrement par défaut des enveloppes ?

Il n'y a pas de coût supplémentaire associé au plan de contrôle Kubernetes géré si vous utilisez une clé appartenant à Amazon Web Services pour le chiffrement par défaut des enveloppes. Par défaut, chaque cluster EKS exécutant Kubernetes version 1.28 ou ultérieure utilise une clé appartenant à Amazon Web Service. Toutefois, si vous utilisez votre propre clé AWS KMS, la tarification KMS normale s'appliquera.

Combien coûte l'utilisation de ma propre clé AWS KMS pour chiffrer les données de l'API Kubernetes dans mon cluster ?

Vous payez 1$ par mois pour stocker toute clé personnalisée que vous créez ou importez dans KMS. KMS facture les demandes de chiffrement et de déchiffrement. Il existe un forfait gratuit de 20 000 requêtes par mois et par compte. Au-delà de ce forfait gratuit, vous payez 0,03 $ par tranche de 10 000 requêtes supplémentaires par mois. Cela s'applique à toutes les utilisations de KMS pour un compte, de sorte que le coût d'utilisation de votre propre clé AWS KMS sur votre cluster sera affecté par l'utilisation de cette clé sur d'autres clusters ou AWS ressources de votre compte.

Mes frais KMS seront-ils plus élevés maintenant que ma clé gérée par le client (CMK) est utilisée pour crypter toutes les données API Kubernetes et pas seulement les secrets ?

Non Notre mise en œuvre avec KMS v2 réduit considérablement le nombre d'appels passés à AWS KMS. Cela permettra à son tour de réduire les coûts associés à votre CMK, indépendamment des données Kubernetes supplémentaires cryptées ou décryptées dans votre cluster EKS.

Comme indiqué ci-dessus, la clé DEK générée utilisée pour le chiffrement des ressources Kubernetes est stockée localement dans le cache du serveur API Kubernetes après avoir été chiffrée

avec la clé KEK distante. Si la graine DEK chiffrée ne se trouve pas dans le cache du serveur API, le serveur API appellera AWS KMS pour chiffrer la graine DEK. Le serveur API met ensuite en cache la graine DEK chiffrée pour une utilisation future dans le cluster sans appeler KMS. De même, pour les demandes de déchiffrement, le serveur API appellera AWS KMS pour la première demande de déchiffrement, après quoi la graine DEK déchiffrée sera mise en cache et utilisée pour les futures opérations de déchiffrement.

Pour plus d'informations, consultez KEP-3299 : Améliorations de KMS v2 dans les améliorations de Kubernetes sur. GitHub

Puis-je utiliser la même clé CMK pour plusieurs clusters Amazon EKS ?

Oui. Pour réutiliser une clé, vous pouvez la lier à un cluster dans la même région en associant l'ARN au cluster lors de sa création. Toutefois, si vous utilisez la même CMK pour plusieurs clusters EKS, vous devez mettre en place les mesures nécessaires pour empêcher la désactivation arbitraire de la CMK. Sinon, une CMK désactivée associée à plusieurs clusters EKS aura un impact plus large sur les clusters en fonction de cette clé.

Qu'arrive-t-il à mon cluster EKS si ma clé CMK devient indisponible après l'activation du chiffrement d'enveloppe par défaut ?

Si vous désactivez une clé KMS, celle-ci ne peut plus être utilisée dans aucune opération cryptographique. Sans accès à une CMK existante, le serveur API ne pourra pas chiffrer et conserver les objets Kubernetes nouvellement créés, ni déchiffrer les objets Kubernetes précédemment chiffrés stockés dans etcd. Si la clé CMK est désactivée, le cluster sera immédiatement placé dans un unhealthy/degraded état tel que nous ne serons pas en mesure de respecter notre engagement de service tant que vous n'aurez pas réactivé la clé CMK associée.

Lorsqu'une CMK est désactivée, vous recevrez des notifications concernant la dégradation de l'état de votre cluster EKS et la nécessité de réactiver votre CMK dans les 30 jours suivant sa désactivation afin de garantir la restauration réussie de vos ressources du plan de contrôle Kubernetes.

Comment protéger mon cluster EKS de l'impact d'une clé disabled/deleted CMK ?

Pour protéger vos clusters EKS contre un tel événement, vos administrateurs clés doivent gérer l'accès aux opérations clés KMS à l'aide de politiques IAM basées sur le principe du moindre privilège afin de réduire le risque de désactivation ou de suppression arbitraire des clés associées aux clusters EKS. De plus, vous pouvez configurer une CloudWatch alarme pour être informé de l'état de votre CMK.

Mon cluster EKS sera-t-il restauré si je réactive le CMK ?

Pour garantir la réussite de la restauration de votre cluster EKS, nous vous recommandons vivement de réactiver votre CMK dans les 30 jours suivant sa désactivation. Cependant, le succès de la restauration de votre cluster EKS dépendra également de la question de savoir s'il subit ou non des modifications perturbatrices de l'API en raison d'une mise à niveau automatique de Kubernetes qui peut avoir lieu alors que le cluster est dans un état. unhealthy/degraded

Pourquoi mon cluster EKS est-il placé dans un unhealthy/degraded état après avoir désactivé le CMK ?

Le serveur API du plan de contrôle EKS utilise une clé DEK qui est cryptée et mise en cache dans la mémoire du serveur API pour chiffrer tous les objets pendant les create/update opérations avant qu'ils ne soient stockés dans etcd. Lorsqu'un objet existant est récupéré depuis etcd, le serveur API utilise la même clé DEK mise en cache et déchiffre l'objet de ressource Kubernetes. Si vous désactivez la CMK, le serveur API ne subira aucun impact immédiat grâce à la clé DEK mise en cache dans la mémoire du serveur API. Toutefois, lorsque l'instance du serveur d'API est redémarrée, elle ne possède pas de DEK en cache et devra appeler AWS KMS pour les opérations de chiffrement et de déchiffrement. Sans CMK, ce processus échouera avec un code d'erreur KMS_KEY_DISABLED, empêchant le serveur API de démarrer correctement.

Qu'arrive-t-il à mon cluster EKS si je supprime mon CMK ?

La suppression de la clé CMK associée à votre cluster EKS dégradera son état de manière irréversible. Sans la clé CMK de votre cluster, le serveur API ne sera plus en mesure de chiffrer et de conserver les nouveaux objets Kubernetes, ni de déchiffrer les objets Kubernetes précédemment chiffrés stockés dans la base de données etcd. Vous ne devez supprimer une clé CMK pour votre cluster EKS que lorsque vous êtes certain de ne plus avoir besoin d'utiliser ce cluster.

Veuillez noter que si la CMK est introuvable (KMS_KEY_NOT_FOUND) ou si les autorisations pour la CMK associée à votre cluster sont révoquées (KMS_GRANT_REVOKED), votre cluster ne pourra pas être restauré. Pour plus d'informations sur l'état du cluster et les codes d'erreur, voir État du cluster FAQs et codes d'erreur avec chemins de résolution.

Vais-je quand même être débité pour un cluster degraded/unhealthy EKS parce que j'ai désactivé ou supprimé ma clé CMK ?

Oui. Bien que le plan de contrôle EKS ne soit pas utilisable en cas de désactivation d'une clé CMK, les ressources d'infrastructure dédiées allouées au cluster EKS AWS continueront d'être utilisées jusqu'à ce qu'il soit supprimé par le client. De plus, notre engagement de service ne s'appliquera pas

dans une telle circonstance, car ce sera une action ou une inaction volontaire de la part du client qui empêchera le bon fonctionnement et l'état normal de votre cluster EKS.

Mon cluster EKS peut-il être automatiquement mis à niveau lorsqu'il est dans un unhealthy/degraded état en raison d'une clé CMK désactivée ?

Oui. Toutefois, si votre cluster dispose d'une CMK désactivée, vous disposerez d'un délai de 30 jours pour la réactiver. Au cours de cette période de 30 jours, votre cluster Kubernetes ne sera pas automatiquement mis à niveau. Toutefois, si ce délai expire et que vous n'avez pas réactivé la CMK, le cluster sera automatiquement mis à niveau vers la version suivante (n+1) bénéficiant d'un support standard, conformément au cycle de vie des versions Kubernetes dans EKS.

Nous vous recommandons vivement de réactiver rapidement une CMK désactivée dès que vous constatez qu'un cluster est affecté. Il est important de noter que, bien qu'EKS mette automatiquement à niveau ces clusters concernés, rien ne garantit qu'ils seront restaurés avec succès, en particulier si le cluster subit plusieurs mises à niveau automatiques, car cela peut entraîner des modifications de l'API Kubernetes et un comportement inattendu dans le processus de démarrage du serveur API.

Puis-je utiliser un alias de clé KMS ?

Oui. Amazon EKS prend en charge l'utilisation des alias de clés KMS. Un alias est un nom convivial attribué à une clé KMS Amazon Web Service. Par exemple, un alias vous permet de faire référence à une clé KMS en tant que my-key au lieu de **`1234abcd-12ab-34cd-56ef-1234567890ab`** .

Puis-je toujours sauvegarder et restaurer les ressources de mon cluster à l'aide de ma propre solution de sauvegarde Kubernetes ?

Oui. Vous pouvez utiliser une solution de sauvegarde Kubernetes (comme Velero) pour la reprise après sinistre, la migration des données et la protection des données du cluster Kubernetes. Si vous utilisez une solution de sauvegarde Kubernetes qui accède aux ressources du cluster via le serveur API, toutes les données récupérées par l'application seront déchiffrées avant d'atteindre le client. Cela vous permettra de récupérer les ressources du cluster dans un autre cluster Kubernetes.

Considérations de sécurité relatives au mode automatique Amazon EKS

Cette rubrique décrit l'architecture de sécurité, les mécanismes de contrôle et les meilleures pratiques appliqués dans le mode automatique Amazon EKS. À mesure que les organisations déploient des applications conteneurisées à grande échelle, le maintien d'une posture de sécurité solide devient

de plus en plus complexe. Le mode automatique EKS met en œuvre des contrôles de sécurité automatisés et s'intègre aux services de sécurité AWS pour vous aider à protéger l'infrastructure de vos clusters, ainsi que vos charges de travail et vos données. Grâce à des fonctionnalités de sécurité intégrées, telles que la gestion obligatoire du cycle de vie des nœuds et le déploiement automatisé des correctifs, le mode automatique EKS vous aide à maintenir les meilleures pratiques de sécurité tout en réduisant la charge opérationnelle.

Avant d'aborder cette rubrique, assurez-vous d'être familier avec les concepts de base du mode automatique EKS et d'avoir consulté les prérequis pour l'activer sur vos clusters. Pour des informations générales sur la sécurité d'Amazon EKS, consultez Sécurité.

Le mode automatique Amazon EKS repose sur les fondations de sécurité existantes d'Amazon EKS tout en introduisant des contrôles de sécurité automatisés supplémentaires pour les instances gérées EC2.

Sécurité et authentification de l'API

Le mode automatique Amazon EKS utilise les mécanismes de sécurité de la plateforme AWS afin de sécuriser et authentifier les appels à l'API Amazon EKS.

- L'accès à l'API Kubernetes est protégé par des entrées d'accès EKS, qui s'intègrent aux identités IAM AWS.
  - Pour de plus amples informations, consultez the section called "Entrées d'accès".
- Les clients peuvent mettre en œuvre un contrôle d'accès granulaire au point de terminaison de l'API Kubernetes grâce à la configuration des entrées d'accès EKS.

Sécurité du réseau

Le mode automatique Amazon EKS prend en charge plusieurs niveaux de sécurité réseau :

- Intégration VPC
  - Fonctionne au sein de votre Amazon Virtual Private Cloud (VPC)
  - Prend en charge les configurations VPC personnalisées et les configurations de sous-réseaux
  - Permet un réseau privé entre les composants du cluster
  - Pour plus d'informations, consultez la section Gestion des responsabilités en matière de sécurité pour Amazon Virtual Private Cloud
- Politiques réseau

- Prise en charge native des politiques réseau Kubernetes
- Possibilité de définir des règles de trafic réseau granulaires
- Pour plus d'informations, consultez the section called "Politiques Kubernetes".

Sécurité des instances EC2 gérées

Le mode automatique Amazon EKS exploite des instances EC2 gérées avec les contrôles de sécurité suivants :

Sécurité EC2

- Les instances gérées EC2 conservent les fonctionnalités de sécurité d'Amazon EC2.
- Pour plus d'informations sur les instances gérées EC2, consultez la section Sécurité dans Amazon EC2.

Gestion du cycle de vie des instances

Les instances gérées EC2 exploitées par le mode automatique EKS ont une durée de vie maximale de 21 jours. Le mode automatique Amazon EKS met automatiquement fin aux instances qui dépassent cette durée. Cette limite de cycle de vie permet d'éviter toute dérive de configuration et de maintenir une posture de sécurité solide.

Protection des données

- Le stockage des instances Amazon EC2 est chiffré. Il s'agit d'un stockage directement connecté à l'instance. Pour de plus amples informations, consultez la section Protection des données dans Amazon EC2.
- Le mode automatique EKS gère les volumes rattachés aux instances EC2 au moment de leur création, y compris les volumes racine et les volumes de données. Le mode automatique EKS ne gère pas entièrement les volumes EBS créés via les fonctionnalités de stockage persistant de Kubernetes.

Gestion des correctifs

- Le mode automatique Amazon EKS applique automatiquement les correctifs aux instances gérées.
- Les correctifs incluent :

- Mises à jour du système d'exploitation
- Correctifs de sécurité
- Composants du mode automatique Amazon EKS

> **Note**
>
> Les clients retiennent la responsabilité de la sécurisation et de la mise à jour des charges de travail exécutées sur ces instances.

Contrôles d'accès

- L'accès direct aux instances est restreint :
  - L'accès SSH n'est pas disponible.
  - L'accès via le gestionnaire de session (SSM) d'AWS Systems Manager n'est pas disponible.
- Les opérations de gestion sont effectuées via l'API Amazon EKS et l'API Kubernetes.

Gestion automatisée des ressources

Le mode automatique Amazon EKS ne gère pas entièrement les volumes Amazon Elastic Block Store (Amazon EBS) créés à l'aide des fonctionnalités de stockage persistant Kubernetes. Le mode automatique EKS ne gère pas non plus les équilibreurs de charge Elastic Load Balancer (ELB). Cependant, le mode automatique Amazon EKS automatise les tâches courantes liées à ces ressources.

Sécurité du stockage

- AWS recommande d'activer le chiffrement pour les volumes EBS provisionnés à l'aide des fonctionnalités de stockage persistant de Kubernetes. Pour de plus amples informations, consultez the section called "Créez StorageClass".
- Chiffrement au repos à l'aide d'AWS KMS
- Vous pouvez configurer votre compte AWS pour imposer le chiffrement des nouveaux volumes EBS et des copies d'instantané que vous créez. Pour plus d'informations, consultez Activation du chiffrement Amazon EBS par défaut dans le Guide de l'utilisateur Amazon EBS.
- Pour plus d'informations, consultez Sécurité dans Amazon EBS.

Sécurité des équilibreurs de charge

- Configuration automatisée des équilibreurs de charge Elastic Load Balancer
- Gestion des certificats SSL/TLS par intégration à AWS Certificate Manager
- Automatisation des groupes de sécurité pour le contrôle d'accès aux équilibreurs de charge
- Pour plus d'informations, consultez Sécurité dans Elastic Load Balancing.

Bonnes pratiques de sécurité

La section suivante décrit les bonnes pratiques en matière de sécurité pour le mode automatique Amazon EKS.

- Passez régulièrement en revue les politiques IAM AWS et les entrées d'accès EKS.
- Mettez en œuvre le principe du moindre privilège pour les charges de travail.
- Surveillez l'activité du cluster via AWS CloudTrail et Amazon CloudWatch. Pour plus d'informations, consultez the section called " AWS CloudTrail" et the section called "Amazon CloudWatch".
- Utilisez AWS Security Hub pour évaluer le niveau de sécurité.
- Mettez en œuvre des normes de sécurité des pods adaptées aux charges de travail.

Considérations relatives à la sécurité relatives aux fonctionnalités EKS

Cette rubrique aborde les principales considérations de sécurité relatives aux fonctionnalités EKS, notamment la configuration des rôles IAM, les autorisations Kubernetes et les modèles architecturaux pour les déploiements multi-clusters et la gestion des ressources entre comptes. AWS

Les fonctionnalités d'EKS utilisent une combinaison de rôles IAM, d'entrées d'accès EKS et de Kubernetes RBAC pour fournir un accès sécurisé aux AWS services, aux ressources Kubernetes du cluster et des intégrations avec Secrets Manager et d'autres services. AWS CodeConnections AWS AWS

Capacité : rôle IAM

Lorsque vous créez une fonctionnalité, vous fournissez un rôle de fonctionnalité IAM qu'EKS utilise pour effectuer des actions en votre nom. Ce rôle doit :

- Être dans le même AWS compte que le cluster et la ressource de capacité
- Adopter une politique de confiance permettant au directeur du `capabilities.eks.amazonaws.com` service d'assumer le rôle
- Disposez d'autorisations IAM adaptées au type de fonctionnalité et au cas d'utilisation, en fonction de vos besoins. Pour des informations détaillées sur les autorisations IAM requises, consultez the section called " AWS CodeConnections"the section called " AWS Secrets Manager", et the section called "Configurer les autorisations "

Il est recommandé de prendre en compte l'étendue des privilèges requis pour votre cas d'utilisation spécifique et d'accorder uniquement les autorisations nécessaires pour répondre à vos exigences. Par exemple, lorsque vous utilisez la fonctionnalité EKS pour Kube Resource Orchestrator, aucune autorisation IAM peut être requise, tandis que lorsque vous utilisez la fonctionnalité EKS pour les AWS contrôleurs pour Kubernetes, vous pouvez accorder un accès complet à un ou plusieurs services. AWS

> **Important**
>
> Bien que certains cas d'utilisation puissent justifier l'utilisation de privilèges administratifs étendus, suivez le principe du moindre privilège en n'accordant que les autorisations IAM minimales requises pour votre cas d'utilisation spécifique, en limitant l'accès à des ressources spécifiques à l'aide de clés ARNs et de conditions plutôt que d'utiliser des autorisations génériques.

Pour des informations détaillées sur la création et la configuration des rôles IAM liés aux fonctionnalités, consultezthe section called "Capacité : rôle IAM".

Entrées d'accès EKS

Lorsque vous créez une fonctionnalité avec un rôle IAM, Amazon EKS crée automatiquement une entrée d'accès pour ce rôle sur votre cluster. Cette entrée d'accès accorde à la base de capacités Kubernetes les autorisations nécessaires pour fonctionner.

> **Note**
>
> Des entrées d'accès sont créées pour le cluster dans lequel la fonctionnalité est créée. Pour les déploiements d'Argo CD vers des clusters distants, vous devez créer des entrées d'accès

sur ces clusters avec les autorisations appropriées pour que l'Argo CD puisse déployer et gérer des applications.

L'entrée d'accès inclut :

- Le rôle IAM (ARN) en tant que principal
- Politiques d'entrée d'accès spécifiques aux fonctionnalités qui accordent des autorisations Kubernetes de base
- Étendue appropriée (à l'échelle du cluster ou étendue à l'espace de noms) en fonction du type de fonctionnalité

> **Note**
>
> Pour Argo CD, les autorisations étendues à l'espace de noms sont accordées à l'espace de noms spécifié dans la configuration des fonctionnalités (par défaut, c'est). `argocd`

Politiques de saisie d'accès par défaut par fonctionnalité

Chaque type de capacité accorde au rôle de capacité les autorisations requises, en définissant différentes politiques d'entrée d'accès par défaut comme suit :

kro

- `arn:aws:eks::aws:cluster-access-policy/AmazonEKSKROPolicy`(délimité par cluster)

  Accorde les autorisations nécessaires pour surveiller, gérer ResourceGraphDefinitions et créer des instances de ressources personnalisées définies par RGDs.

ACK

- `arn:aws:eks::aws:cluster-access-policy/AmazonEKSACKPolicy`(délimité par cluster)

  Accorde les autorisations nécessaires pour créer, lire, mettre à jour et supprimer des ressources personnalisées ACK dans tous les espaces de noms.

Argo CD

- `arn:aws:eks::aws:cluster-access-policy/AmazonEKSArgoCDClusterPolicy`(délimité par cluster)

  Accorde des autorisations au niveau du cluster pour qu'Argo CD découvre des ressources et gère les objets délimités au cluster.

- `arn:aws:eks::aws:cluster-access-policy/AmazonEKSArgoCDPolicy`(limité à l'espace de noms)

  Accorde des autorisations au niveau de l'espace de noms permettant à Argo CD de déployer et de gérer des applications. Limité à l'espace de noms spécifié dans la configuration des fonctionnalités (par défaut). `argocd`

Voir the section called "Vérification des stratégies d'accès" pour des informations plus détaillées.

Autorisations Kubernetes supplémentaires

Certaines fonctionnalités peuvent nécessiter des autorisations Kubernetes supplémentaires au-delà des politiques de saisie d'accès par défaut. Vous pouvez accorder ces autorisations en utilisant l'une des méthodes suivantes :

- Politiques de saisie d'accès : associez des politiques gérées supplémentaires à l'entrée d'accès
- Kubernetes RBAC : création `Role` de `ClusterRole` liaisons pour l'utilisateur Kubernetes de la fonctionnalité

Autorisations du lecteur secret ACK

Certains contrôleurs ACK doivent lire les secrets de Kubernetes pour récupérer des données sensibles telles que les mots de passe des bases de données. Les contrôleurs ACK suivants nécessitent un accès secret en lecture :

- `acm`, `acmpca`, `documentdb`, `memorydb`, `mq`, `rds`, `secretsmanager`

Pour accorder des autorisations de lecture secrète, procédez comme suit :

1. Associer la politique d'`arn:aws:eks::aws:cluster-access-policy/AmazonEKSSecretReaderPolicy`accès à l'entrée d'accès de la fonctionnalité

2. Étendre la politique à des espaces de noms spécifiques où les ressources ACK référenceront des secrets ou accorderont un accès à l'échelle du cluster

> **Important**
>
> Les autorisations de lecture secrète sont limitées aux espaces de noms que vous spécifiez lors de l'association de la politique d'entrée d'accès. Cela vous permet de limiter les secrets auxquels la fonctionnalité peut accéder.

autorisations de ressources arbitraires kro

kro a besoin d'autorisations pour créer et gérer les ressources définies dans votre ResourceGraphDefinitions. Par défaut, Kro ne peut que se surveiller et RGDs se gérer lui-même.

Pour accorder à Kro l'autorisation de créer des ressources, procédez comme suit :

Option 1 : accéder aux politiques d'entrée

Associez des politiques d'entrée d'accès prédéfinies telles que `AmazonEKSAdminPolicy` ou `AmazonEKSEditPolicy` à l'entrée d'accès de la fonctionnalité.

Option 2 : Kubernetes RBAC

Créez un `ClusterRoleBinding` qui accorde à l'utilisateur Kubernetes de la fonctionnalité les autorisations nécessaires :

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: kro-cluster-admin
subjects:
- kind: User
  name: arn:aws:sts::111122223333:assumed-role/my-kro-role/KRO
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: cluster-admin
  apiGroup: rbac.authorization.k8s.io
```

> **Note**
>
> Le nom d'utilisateur Kubernetes pour kro suit le modèle suivant : `arn:aws:sts::ACCOUNT_ID:assumed-role/ROLE_NAME/KRO`
> Le nom de session `/KRO` (en majuscules) est automatiquement défini par la fonctionnalité EKS kro.

Autorisations IAM requises par fonctionnalité

kro (Kube Resource Orchestrator)

Aucune autorisation IAM n'est requise. Vous pouvez créer un rôle de capacité sans politique attachée. Kro nécessite uniquement les autorisations RBAC de Kubernetes.

ACK (AWS contrôleurs pour Kubernetes)

Nécessite des autorisations pour gérer les AWS ressources qu'ACK créera et gérera. Vous devez définir les autorisations pour des services, des actions et des ressources spécifiques en fonction de vos besoins. Pour des informations détaillées sur la configuration des autorisations ACK, y compris les meilleures pratiques de production avec les sélecteurs de rôle IAM, consultez. the section called “Configurer les autorisations ”

Argo CD

Aucune autorisation IAM n'est requise par défaut. Des autorisations facultatives peuvent être nécessaires pour :

- AWS Secrets Manager : si vous stockez les informations d'identification du référentiel Git dans Secrets Manager
- AWS CodeConnections: en cas d'utilisation CodeConnections pour l'authentification du dépôt Git
- Amazon ECR : si vous utilisez des diagrammes Helm stockés au format OCI dans Amazon ECR

Bonnes pratiques de sécurité

Le moindre privilège de l'IAM

Accordez à vos ressources de capacité uniquement les autorisations requises pour votre cas d'utilisation. Cela ne signifie pas que vous ne pouvez pas accorder des autorisations administratives étendues à vos fonctionnalités si nécessaire. Dans de tels cas, vous devez gérer l'accès à ces ressources de manière appropriée.

Rôles liés aux capacités :

- ACK : Dans la mesure du possible, limitez les autorisations IAM aux AWS services et ressources spécifiques dont vos équipes ont besoin, en fonction des cas d'utilisation et des exigences
- Argo CD : Restreindre l'accès à des référentiels Git et à des espaces de noms Kubernetes spécifiques
- kro : nécessite un rôle de capacité pour la politique de confiance, mais aucune autorisation IAM n'est requise (utilise uniquement le cluster RBAC)

Exemple : au lieu de`"Resource": "*"`, spécifiez des modèles pour des ressources ou des groupes de ressources spécifiques.

```
"Resource": [
  "arn:aws:s3:::my-app-*",
  "arn:aws:rds:us-west-2:111122223333:db:prod-*"
]
```

Utilisez les clés de condition IAM pour restreindre davantage l'accès :

```
"Condition": {
  "StringEquals": {
    "aws:ResourceTag/Environment": "production"
  }
}
```

Pour plus d'informations sur la configuration IAM, consultez la section des considérations relatives à chaque fonctionnalité.

Isolation de l'espace de noms pour les secrets d'Argo CD

La fonctionnalité Argo CD gérée a accès à tous les secrets Kubernetes dans son espace de noms configuré (par défaut :). `argocd` Pour maintenir une posture de sécurité optimale, suivez les pratiques d'isolation des espaces de noms suivantes :

- Conservez uniquement les secrets relatifs à Argo CD dans l'espace de noms Argo CD
- Évitez de stocker des secrets d'application indépendants dans le même espace de noms qu'Argo CD
- Utilisez des espaces de noms distincts pour les secrets d'application qui ne sont pas nécessaires pour les opérations sur Argo CD

Cette isolation garantit que l'accès secret d'Argo CD est limité aux seules informations d'identification dont il a besoin pour l'authentification du référentiel Git et pour d'autres opérations spécifiques à Argo CD.

RBAC Kubernetes

Contrôlez quels utilisateurs et quels comptes de service peuvent créer et gérer des ressources de fonctionnalités. Il est recommandé de déployer des ressources de capacité dans des espaces de noms dédiés avec des politiques RBAC appropriées.

Exemple : rôle RBAC compatible avec ACK, permettant de gérer les ressources du compartiment S3 dans l'espace de `app-team` noms :

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: ack-s3-manager
  namespace: app-team
rules:
- apiGroups: ["s3.services.k8s.aws"]
  resources: ["buckets"]
  verbs: ["get", "list", "create", "update", "delete"]
```

Journaux d'audit

CloudTrail: Toutes les opérations de l'API EKS Capability (création, mise à jour, suppression) sont enregistrées AWS CloudTrail.

Activez la CloudTrail journalisation pour suivre :

- Qui a créé ou modifié les fonctionnalités
- Lorsque les configurations des capacités ont changé
- Quels rôles de capacité sont utilisés

Accès au réseau et points de terminaison VPC

Accès privé à l'API Argo CD

Vous pouvez restreindre l'accès au serveur d'API Argo CD en associant un ou plusieurs points de terminaison VPC au point de terminaison Argo CD hébergé. Cela permet une connectivité privée depuis votre VPC sans passer par l'Internet public. Le point de terminaison VPC fournit un accès à la fois à l'interface utilisateur Web d'Argo CD et à l'API Argo CD (y compris l'accès à la CLI).

> **Note**
>
> Points de terminaison VPC connectés aux points de terminaison de l'API Argo CD hébergés (à l'aide des fonctionnalités eks). *region*.amazonaws.com) ne prennent pas en charge les politiques relatives aux points de terminaison VPC.

Déploiement sur des clusters privés

La fonctionnalité Argo CD permet de déployer des applications sur des clusters EKS entièrement privés, offrant ainsi un avantage opérationnel significatif en éliminant le besoin de peering VPC ou de configurations réseau complexes. Cependant, lors de la conception de cette architecture, considérez qu'Argo CD extraira la configuration des référentiels Git (qui peuvent être publics) et l'appliquera à vos clusters privés.

Assurez-vous de :

- Utiliser des référentiels Git privés pour les charges de travail sensibles
- Mettre en œuvre des contrôles d'accès et une authentification appropriés au référentiel Git
- Vérifiez et approuvez les modifications par le biais de pull requests avant de les fusionner
- Envisagez d'utiliser les fenêtres de synchronisation d'Argo CD pour contrôler le moment où les déploiements peuvent avoir lieu

- Surveillez les journaux d'audit d'Argo CD pour détecter les modifications de configuration non autorisées

Conformité d'

Les fonctionnalités d'EKS sont entièrement gérées et possèdent les certifications de conformité d'Amazon EKS.

Pour obtenir des informations de conformité actuelles, consultez la section AWS Services concernés par programme de conformité.

Étapes suivantes

- the section called "Configurer les autorisations "- Configurer les autorisations IAM pour ACK
- the section called "Configurer les autorisations "- Configurer Kubernetes RBAC pour kro
- the section called "Configurer les autorisations "- Configurer l'intégration d'Identity Center pour Argo CD
- the section called "Fonctionnalités de résolution des problèmes"- Résoudre les problèmes de sécurité et d'autorisation

Gestion des identités et des accès pour Amazon EKS

AWS Identity and Access Management (IAM) est un AWS service qui permet à un administrateur de contrôler en toute sécurité l'accès aux ressources. AWS Des administrateurs IAM contrôlent les personnes qui peuvent être authentifiées (connectées) et autorisées (disposant d'autorisations) pour utiliser des ressources Amazon EKS. IAM est un AWS service que vous pouvez utiliser sans frais supplémentaires.

Public ciblé

La façon dont vous utilisez AWS Identity and Access Management (IAM) varie en fonction du travail que vous effectuez dans Amazon EKS.

Utilisateur du service : si vous utilisez le service Amazon EKS pour accomplir votre tâche, votre administrateur vous fournit les informations d'identification et les autorisations dont vous avez besoin. Vous pourrez avoir besoin d'autorisations supplémentaires si vous utilisez davantage

de fonctionnalités Amazon EKS. Si vous comprenez bien la gestion des accès, vous saurez demander les autorisations appropriées à votre administrateur. Si vous ne pouvez pas accéder à une fonctionnalité dans Amazon EKS, consultez the section called "Résolution des problèmes".

Administrateur de service : si vous êtes responsable des ressources Amazon EKS au sein de votre entreprise, vous disposez probablement d'un accès complet à Amazon EKS. Il vous incombe de déterminer les fonctionnalités et ressources Amazon EKS auxquelles les utilisateurs de votre service doivent avoir accès, c'est votre tâche. Vous devez ensuite soumettre les demandes à votre administrateur IAM pour modifier les autorisations des utilisateurs de votre service. Consultez les informations sur cette page pour comprendre les concepts de base d'IAM. Pour en savoir plus sur la façon dont votre entreprise peut utiliser IAM avec Amazon EKS, consultez the section called "Amazon EKS et IAM".

Administrateur IAM : si vous êtes administrateur IAM, vous voudrez peut-être en savoir plus sur la manière de rédiger des politiques pour gérer l'accès à Amazon EKS. Pour afficher des exemples de politiques basées sur l'identité Amazon EKS que vous pouvez utiliser dans IAM, consultez the section called "Politiques basées sur l'identité".

Authentification par des identités

L'authentification est la façon dont vous vous connectez à AWS l'aide de vos informations d'identification. Vous devez être authentifié (connecté à AWS) en tant qu'utilisateur root du AWS compte, en tant qu'utilisateur IAM ou en assumant un rôle IAM.

Vous pouvez vous connecter en AWS tant qu'identité fédérée en utilisant les informations d'identification fournies par le biais d'une source d'identité. AWS Les utilisateurs de l'IAM Identity Center (IAM Identity Center), l'authentification unique de votre entreprise et vos informations d'identification Google ou Facebook sont des exemples d'identités fédérées. Lorsque vous vous connectez avec une identité fédérée, votre administrateur aura précédemment configuré une fédération d'identités avec des rôles IAM. Lorsque vous accédez à AWS l'aide de la fédération, vous assumez indirectement un rôle.

Selon le type d'utilisateur que vous êtes, vous pouvez vous connecter au portail AWS Management Console ou au portail AWS d'accès. Pour plus d'informations sur la connexion à AWS, consultez Comment vous connecter à votre AWS compte dans le Guide de l'utilisateur de AWS connexion.

Si vous y accédez AWS par programmation, AWS fournit un kit de développement logiciel (SDK) et une interface de ligne de commande (CLI) pour signer cryptographiquement vos demandes à l'aide de vos informations d'identification. Si vous n'utilisez pas d' AWS outils, vous devez signer vous-

même les demandes. Pour plus d'informations sur l'utilisation de la méthode recommandée pour signer vous-même les demandes, consultez la section Signature des demandes AWS d'API dans le guide de l'utilisateur IAM.

Quelle que soit la méthode d'authentification que vous utilisez, vous devrez peut-être fournir des informations de sécurité supplémentaires. Par exemple, il vous AWS recommande d'utiliser l'authentification multifactorielle (MFA) pour renforcer la sécurité de votre compte. Pour en savoir plus, consultez Authentification multifactorielle dans le guide de l'utilisateur d' AWS IAM Identity Center et Utilisation de l'authentification multifactorielle (MFA) AWS dans le guide de l'utilisateur IAM.

AWS utilisateur root du compte

Lorsque vous créez un AWS compte, vous commencez par utiliser une seule identité de connexion qui donne un accès complet à tous les AWS services et ressources du compte. Cette identité est appelée utilisateur root du AWS compte et est accessible en vous connectant avec l'adresse e-mail et le mot de passe que vous avez utilisés pour créer le compte. Il est vivement recommandé de ne pas utiliser l'utilisateur racine pour vos tâches quotidiennes. Protégez vos informations d'identification d'utilisateur racine et utilisez-les pour effectuer les tâches que seul l'utilisateur racine peut effectuer. Pour obtenir la liste complète des tâches qui vous imposent de vous connecter en tant qu'utilisateur racine, consultez Tâches nécessitant les informations d'identification de l'utilisateur racine dans le Guide de l'utilisateur IAM.

Utilisateurs et groupes IAM

Un utilisateur IAM est une identité au sein de votre AWS compte qui possède des autorisations spécifiques pour une seule personne ou une seule application. Dans la mesure du possible, nous vous recommandons de vous appuyer sur des informations d'identification temporaires plutôt que de créer des utilisateurs IAM ayant des informations d'identification à long terme telles que des mots de passe et des clés d'accès. Toutefois, si certains cas d'utilisation spécifiques nécessitent des informations d'identification à long terme avec les utilisateurs IAM, nous vous recommandons d'effectuer une rotation des clés d'accès. Pour plus d'informations, consultez Rotation régulière des clés d'accès pour les cas d'utilisation nécessitant des informations d'identification dans le Guide de l'utilisateur IAM.

Un groupe IAM est une identité qui concerne un ensemble d'utilisateurs IAM. Vous ne pouvez pas vous connecter en tant que groupe. Vous pouvez utiliser les groupes pour spécifier des autorisations pour plusieurs utilisateurs à la fois. Les groupes permettent de gérer plus facilement les autorisations pour de grands ensembles d'utilisateurs. Par exemple, vous pouvez nommer un groupe IAMAdminset lui donner les autorisations nécessaires pour administrer les ressources IAM.

Les utilisateurs sont différents des rôles. Un utilisateur est associé de manière unique à une personne ou une application, alors qu'un rôle est conçu pour être endossé par tout utilisateur qui en a besoin. Les utilisateurs disposent d'informations d'identification permanentes, mais les rôles fournissent des informations d'identification temporaires. Pour en savoir plus, consultez Quand créer un utilisateur IAM (au lieu d'un rôle) dans le Guide de l'utilisateur IAM.

Rôles IAM

Un rôle IAM est une identité au sein de votre AWS compte dotée d'autorisations spécifiques. Le concept ressemble à celui d'utilisateur IAM, mais le rôle IAM n'est pas associé à une personne en particulier. Vous pouvez assumer temporairement un rôle IAM dans le en AWS Management Console changeant de rôle. Vous pouvez assumer un rôle en appelant une opération d' AWS API ou de AWS CLI ou en utilisant une URL personnalisée. Pour plus d'informations sur les méthodes d'utilisation des rôles, consultez Utilisation de rôles IAM dans le Guide de l'utilisateur IAM.

Les rôles IAM avec des informations d'identification temporaires sont utiles dans les cas suivants :

- Accès utilisateur fédéré : pour attribuer des autorisations à une identité fédérée, vous créez un rôle et définissez des autorisations pour le rôle. Quand une identité externe s'authentifie, l'identité est associée au rôle et reçoit les autorisations qui sont définies par celui-ci. Pour obtenir des informations sur les rôles pour la fédération, consultez Création d'un rôle pour un fournisseur d'identité tiers (fédération) dans le Guide de l'utilisateur IAM. Si vous utilisez IAM Identity Center, vous configurez un jeu d'autorisations. IAM Identity Center met en corrélation le jeu d'autorisations avec un rôle dans IAM afin de contrôler à quoi vos identités peuvent accéder après leur authentification. Pour plus d'informations sur les ensembles d'autorisations, consultez la section Ensembles d'autorisations dans le guide de l'utilisateur d' AWS IAM Identity Center.
- Autorisations d'utilisateur IAM temporaires : un rôle ou un utilisateur IAM peut endosser un rôle IAM pour profiter temporairement d'autorisations différentes pour une tâche spécifique.
- Accès intercompte : vous pouvez utiliser un rôle IAM pour permettre à un utilisateur (principal de confiance) d'un compte différent d'accéder aux ressources de votre compte. Les rôles constituent le principal moyen d'accorder l'accès intercompte. Cependant, avec certains AWS services, vous pouvez associer une politique directement à une ressource (au lieu d'utiliser un rôle comme proxy). Pour en savoir plus sur la différence entre les rôles et les politiques basées sur les ressources pour l'accès intercompte, consultez Accès intercompte aux ressources dans IAM dans le Guide de l'utilisateur IAM.
- Accès interservices — Certains AWS services utilisent des fonctionnalités d'autres AWS services. Par exemple, lorsque vous effectuez un appel dans un service, il est courant que ce service

exécute des applications dans Amazon EC2 ou stocke des objets dans Amazon S3. Un service peut le faire en utilisant les autorisations d'appel du principal, une fonction du service ou un rôle lié au service.

- Sessions d'accès direct (FAS) : lorsque vous utilisez un utilisateur ou un rôle IAM pour effectuer des actions AWS, vous êtes considéré comme un mandant. Lorsque vous utilisez certains services, vous pouvez effectuer une action qui initie une autre action dans un autre service. Le FAS utilise les autorisations du principal appelant un AWS service, associées au AWS service demandeur pour adresser des demandes aux services en aval. Les demandes FAS ne sont effectuées que lorsqu'un service reçoit une demande qui nécessite des interactions avec d'autres AWS services ou ressources pour être traitée. Dans ce cas, vous devez disposer d'autorisations nécessaires pour effectuer les deux actions. Pour plus de détails sur une politique lors de la formulation de demandes FAS, consultez Transmission des sessions d'accès.
- Rôle de service : il s'agit d'un rôle IAM attribué à un service afin de réaliser des actions en votre nom. Un administrateur IAM peut créer, modifier et supprimer un rôle de service à partir d'IAM. Pour plus d'informations, consultez la section Création d'un rôle pour déléguer des autorisations à un AWS service dans le Guide de l'utilisateur IAM.
- Rôle lié à un service — Un rôle lié à un service est un type de rôle lié à un service. AWS Le service peut endosser le rôle afin d'effectuer une action en votre nom. Les rôles liés au service apparaissent dans votre AWS compte et appartiennent au service. Un administrateur IAM peut consulter, mais ne peut pas modifier, les autorisations concernant les rôles liés à un service.

- Applications exécutées sur Amazon EC2 : vous pouvez utiliser un rôle IAM pour gérer les informations d'identification temporaires pour les applications qui s'exécutent sur une EC2 instance et qui envoient des requêtes AWS CLI ou AWS API. Cela est préférable au stockage des clés d'accès dans l' EC2 instance. Pour attribuer un AWS rôle à une EC2 instance et le rendre disponible pour toutes ses applications, vous devez créer un profil d'instance attaché à l'instance. Un profil d'instance contient le rôle et permet aux programmes exécutés sur l' EC2 instance d'obtenir des informations d'identification temporaires. Pour plus d'informations, consultez la section Utilisation d'un rôle IAM pour accorder des autorisations aux applications exécutées sur des EC2 instances Amazon dans le guide de l'utilisateur IAM.

Pour savoir dans quel cas utiliser des rôles ou des utilisateurs IAM, consultez Quand créer un rôle IAM (au lieu d'un utilisateur) dans le Guide de l'utilisateur IAM.

Gestion des accès à l'aide de politiques

Vous contrôlez l'accès en AWS créant des politiques et en les associant à AWS des identités ou à des ressources. Une politique est un objet AWS qui, lorsqu'il est associé à une identité ou à une ressource, définit leurs autorisations. AWS évalue ces politiques lorsqu'un principal (utilisateur, utilisateur root ou session de rôle) fait une demande. Les autorisations dans les politiques déterminent si la demande est autorisée ou refusée. La plupart des politiques sont stockées AWS sous forme de documents JSON. Pour plus d'informations sur la structure et le contenu des documents de politique JSON, consultez Vue d'ensemble des politiques JSON dans le Guide de l'utilisateur IAM.

Les administrateurs peuvent utiliser les politiques AWS JSON pour spécifier qui a accès à quoi. C'est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

Par défaut, les utilisateurs et les rôles ne disposent d'aucune autorisation. Pour octroyer aux utilisateurs des autorisations d'effectuer des actions sur les ressources dont ils ont besoin, un administrateur IAM peut créer des politiques IAM. L'administrateur peut ensuite ajouter les politiques IAM aux rôles et les utilisateurs peuvent assumer les rôles.

Les politiques IAM définissent les autorisations d'une action, quelle que soit la méthode que vous utilisez pour exécuter l'opération. Par exemple, supposons que vous disposiez d'une politique qui autorise l'action `iam:GetRole`. Un utilisateur appliquant cette politique peut obtenir des informations sur le AWS Management Console rôle à partir de la AWS CLI ou de l' AWS API.

Politiques basées sur l'identité

Les politiques basées sur l'identité sont des documents de politique d'autorisations JSON que vous pouvez attacher à une identité telle qu'un utilisateur, un groupe d'utilisateurs ou un rôle IAM. Ces politiques contrôlent quel type d'actions des utilisateurs et des rôles peuvent exécuter, sur quelles ressources et dans quelles conditions. Pour découvrir comment créer une politique basée sur l'identité, consultez Création de politiques IAM dans le Guide de l'utilisateur IAM.

Les politiques basées sur l'identité peuvent être classées comme des politiques en ligne ou des politiques gérées. Les politiques en ligne sont intégrées directement à un utilisateur, groupe ou rôle. Les politiques gérées sont des politiques autonomes que vous pouvez associer à plusieurs utilisateurs, groupes et rôles dans votre AWS compte. Les politiques gérées incluent les politiques AWS gérées et les politiques gérées par le client. Pour découvrir comment choisir entre une politique

gérée et une politique en ligne, consultez Choix entre les politiques gérées et les politiques en ligne dans le Guide de l'utilisateur IAM.

Politiques basées sur les ressources

Les politiques basées sur les ressources sont des documents de politique JSON que vous attachez à une ressource. Par exemple, les politiques de confiance de rôle IAM et les politiques de compartiment Amazon S3 sont des politiques basées sur les ressources. Dans les services qui sont compatibles avec les politiques basées sur les ressources, les administrateurs de service peuvent les utiliser pour contrôler l'accès à une ressource spécifique. Pour la ressource dans laquelle se trouve la politique, cette dernière définit quel type d'actions un principal spécifié peut effectuer sur cette ressource et dans quelles conditions. Vous devez spécifier un principal dans une politique basée sur les ressources. Les principaux peuvent inclure des comptes, des utilisateurs, des rôles, des utilisateurs fédérés ou AWS des services.

Les politiques basées sur les ressources sont des politiques en ligne situées dans ce service. Vous ne pouvez pas utiliser les politiques AWS gérées par IAM dans une stratégie basée sur les ressources.

Listes de contrôle d'accès (ACLs)

Les listes de contrôle d'accès (ACLs) contrôlent les principaux (membres du compte, utilisateurs ou rôles) autorisés à accéder à une ressource. ACLs sont similaires aux politiques basées sur les ressources, bien qu'elles n'utilisent pas le format de document de politique JSON.

Amazon S3, AWS WAF et Amazon VPC sont des exemples de services compatibles. ACLs Pour en savoir plus ACLs, consultez la présentation de la liste de contrôle d'accès (ACL) dans le guide du développeur Amazon Simple Storage Service.

Autres types de politique

AWS prend en charge d'autres types de politiques moins courants. Ces types de politiques peuvent définir le nombre maximum d'autorisations qui vous sont accordées par des types de politiques plus courants.

- Limite d'autorisations : une limite d'autorisations est une fonctionnalité avancée dans laquelle vous définissez le nombre maximal d'autorisations qu'une politique basée sur l'identité peut accorder à une entité IAM (utilisateur ou rôle IAM). Vous pouvez définir une limite d'autorisations pour une entité. Les autorisations qui en résultent sont l'intersection des politiques basées sur

l'identité d'une entité et de ses limites d'autorisation. Les politiques basées sur les ressources qui spécifient l'utilisateur ou le rôle dans le champ `Principal` ne sont pas limitées par les limites d'autorisations. Un refus explicite dans l'une de ces politiques annule l'autorisation. Pour plus d'informations sur les limites d'autorisations, consultez Limites d'autorisations pour des entités IAM dans le Guide de l'utilisateur IAM.

- Politiques de contrôle des services (SCPs) : SCPs politiques JSON qui spécifient les autorisations maximales pour une organisation ou une unité organisationnelle (UO) dans AWS Organizations. AWS Organizations est un service permettant de regrouper et de gérer de manière centralisée plusieurs AWS comptes détenus par votre entreprise. Si vous activez toutes les fonctionnalités d'une organisation, vous pouvez appliquer des politiques de contrôle des services (SCPs) à l'un ou à l'ensemble de vos comptes. Le SCP limite les autorisations pour les entités des comptes membres, y compris pour chaque utilisateur root AWS du compte. Pour plus d'informations sur les Organizations et consultez SCPs les politiques de contrôle des services dans le Guide de l'utilisateur AWS des Organizations.
- Politiques de session : les politiques de session sont des politiques avancées que vous utilisez en tant que paramètre lorsque vous créez par programmation une session temporaire pour un rôle ou un utilisateur fédéré. Les autorisations de la session obtenue sont une combinaison des politiques basées sur l'identité de l'utilisateur ou du rôle et des politiques de session. Les autorisations peuvent également provenir d'une politique basée sur les ressources. Un refus explicite dans l'une de ces politiques annule l'autorisation. Pour plus d'informations, consultez Politiques de session dans le Guide de l'utilisateur IAM.

Plusieurs types de politique

Lorsque plusieurs types de politiques s'appliquent à la requête, les autorisations en résultant sont plus compliquées à comprendre. Pour savoir comment AWS déterminer s'il faut autoriser une demande lorsque plusieurs types de politiques sont impliqués, consultez la section Logique d'évaluation des politiques dans le guide de l'utilisateur IAM.

Fonctionnement d'Amazon EKS avec IAM

Avant d'utiliser IAM pour gérer l'accès à Amazon EKS, vous devez comprendre quelles sont les fonctionnalités IAM qui peuvent être utilisées dans cette situation. Pour obtenir une vue d'ensemble de la manière dont Amazon EKS et les autres AWS services fonctionnent avec IAM, consultez la section AWS Services compatibles avec IAM dans le guide de l'utilisateur d'IAM.

Rubriques

- Politiques basées sur l'identité Amazon EKS
- Politiques basées sur les ressources Amazon EKS
- Autorisation basée sur des identifications Amazon EKS
- Rôles IAM Amazon EKS

Politiques basées sur l'identité Amazon EKS

Avec les politiques IAM basées sur l'identité, vous pouvez spécifier des actions et ressources autorisées ou refusées, ainsi que les conditions dans lesquelles les actions sont autorisées ou refusées. Amazon EKS est compatible avec des actions, des ressources et des clés de condition spécifiques. Pour en savoir plus sur tous les éléments que vous utilisez dans une politique JSON, consultez Références des éléments de politique JSON IAM dans le Guide de l'utilisateur IAM.

Actions

Les administrateurs peuvent utiliser les politiques AWS JSON pour spécifier qui a accès à quoi. C'est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

L'élément `Action` d'une politique JSON décrit les actions que vous pouvez utiliser pour autoriser ou refuser l'accès à une politique. Les actions de stratégie portent généralement le même nom que l'opération AWS d'API associée. Il existe certaines exceptions, telles que les actions d'autorisation uniquement qui ne correspondent à aucune opération API. Certaines opérations nécessitent également plusieurs actions dans une politique. Ces actions supplémentaires sont nommées actions dépendantes.

Intégration d'actions dans une politique afin d'accorder l'autorisation d'exécuter les opérations associées.

Les actions de politique dans Amazon EKS utilisent le préfixe suivant avant l'action : `eks:`. Par exemple, pour accorder à une personne l'autorisation d'obtenir des informations descriptives sur un cluster Amazon EKS, incluez l'action `DescribeCluster` dans sa politique. Les déclarations de politique doivent inclure un élément `Action` ou `NotAction`.

Pour spécifier plusieurs actions dans une seule déclaration, séparez-les par des virgules comme suit :

```
"Action": ["eks:action1", "eks:action2"]
```

Vous pouvez aussi spécifier plusieurs actions à l'aide de caractères génériques (*). Par exemple, pour spécifier toutes les actions qui commencent par le mot `Describe`, incluez l'action suivante :

```
"Action": "eks:Describe*"
```

Pour afficher la liste des actions Amazon EKS, consultez la rubrique Actions définies par Amazon Elastic Kubernetes Service de la Référence de l'autorisation de service.

Ressources

Les administrateurs peuvent utiliser les politiques AWS JSON pour spécifier qui a accès à quoi. C'est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

L'élément de politique JSON `Resource` indique le ou les objets auxquels l'action s'applique. Les instructions doivent inclure un élément `Resource` ou `NotResource`. Il est recommandé de définir une ressource à l'aide de son Amazon Resource Name (ARN). Vous pouvez le faire pour des actions qui prennent en charge un type de ressource spécifique, connu sous la dénomination autorisations de niveau ressource.

Pour les actions qui ne prennent pas en charge les autorisations au niveau des ressources, telles que les opérations de listage, utilisez un caractère générique (*) pour indiquer que la déclaration s'applique à toutes les ressources.

```
"Resource": "*"
```

La ressource de cluster Amazon EKS intègre l'ARN suivant.

```
 arn:aws: eks:region-code:account-id:cluster/cluster-name
```

Pour plus d'informations sur le format de ARNs, consultez Amazon resource names (ARNs) et espaces de noms AWS de services Amazon.

Par exemple, pour spécifier le cluster dont le nom figure *`my-cluster`* dans votre instruction, utilisez l'ARN suivant :

```
"Resource": "arn:aws: eks:region-code:111122223333:cluster/my-cluster"
```

Pour spécifier tous les clusters appartenant à un compte et à une AWS région spécifiques, utilisez le caractère générique (*) :

```
"Resource": "arn:aws: eks:region-code:111122223333:cluster/*"
```

Certaines actions Amazon EKS, telles que celles permettant de créer des ressources, ne peuvent pas être effectuées sur une ressource spécifique. Dans ces cas-là, vous devez utiliser le caractère générique (*).

```
"Resource": "*"
```

Pour consulter la liste des types de ressources Amazon EKS et leurs caractéristiques ARNs, consultez la section Ressources définies par Amazon Elastic Kubernetes Service dans le Service Authorization Reference. Pour connaître les actions avec lesquelles vous pouvez spécifier l'ARN de chaque ressource, consultez la rubrique Actions définies par Amazon Elastic Kubernetes Service.

Clés de condition

Amazon EKS définit son propre ensemble de clés de condition et est également compatible avec l'utilisation de certaines clés de condition globales. Pour voir toutes les clés de condition AWS globales, consultez la section Clés contextuelles de condition AWS globale dans le guide de l'utilisateur IAM.

Vous pouvez définir des clés de condition lors de l'association d'un fournisseur OpenID Connect à votre cluster. Pour de plus amples informations, veuillez consulter the section called “Exemple de politique IAM”.

Toutes les actions Amazon EC2 prennent en charge les clés de condition `aws:RequestedRegion` et `ec2:Region`. Pour plus d'informations, voir Exemple : restriction de l'accès à une AWS région spécifique.

Pour obtenir la liste des clés de condition Amazon EKS, consultez la rubrique Conditions définies par Amazon Elastic Kubernetes Service de la Référence de l'autorisation de service. Pour savoir avec quelles actions et ressources vous pouvez utiliser une clé de condition, consultez la rubrique Actions définies par Amazon Elastic Kubernetes Service.

Exemples

Pour voir des exemples de politiques Amazon EKS basées sur l'identité, consultez the section called “Politiques basées sur l’identité”.

Lorsque vous créez un cluster Amazon EKS, le principal IAM qui crée le cluster se voit automatiquement accorder des autorisations `system:masters` dans la configuration du contrôle

d'accès basé sur les rôles (RBAC) du cluster dans le plan de contrôle Amazon EKS. Ce principal n'apparaît dans aucune configuration visible. Souvenez-vous donc du principal qui a créé le cluster à l'origine. Pour accorder à d'autres principaux IAM la possibilité d'interagir avec votre cluster, modifiez le `aws-auth ConfigMap` dans Kubernetes et créez un `rolebinding` ou `clusterrolebinding` Kubernetes associé au nom d'un `group` que vous spécifiez dans le `aws-auth ConfigMap`.

Pour plus d'informations sur l'utilisation du ConfigMap, consultezthe section called "Accès à l'API Kubernetes".

Politiques basées sur les ressources Amazon EKS

Amazon EKS ne prend pas en charge les politiques basées sur les ressources.

Autorisation basée sur des identifications Amazon EKS

Vous pouvez attacher des identifications aux ressources Amazon EKS ou transmettre des identifications dans une demande à Amazon EKS. Pour contrôler l'accès basé sur des identifications, vous devez fournir les informations d'identifications dans l'élément de condition d'une politique utilisant les clés de condition `aws:ResourceTag/key-name` , `aws:RequestTag/key-name` ou `aws:TagKeys`. Pour plus d'informations sur l'étiquetage des ressources Amazon EKS, consultez the section called "Balisage de vos ressources ". Pour plus d'informations sur les actions dans lesquelles vous pouvez utiliser des balises avec des clés de condition, consultez Actions définies par Amazon EKS dans Référence de l'autorisation de service.

Rôles IAM Amazon EKS

Un rôle IAM est une entité de votre AWS compte qui possède des autorisations spécifiques.

Utilisation d'informations d'identification temporaires avec Amazon EKS

Vous pouvez utiliser des informations d'identification temporaires pour vous connecter à l'aide de la fédération, endosser un rôle IAM ou encore pour endosser un rôle intercompte. Vous obtenez des informations d'identification de sécurité temporaires en appelant AWS des opérations d'API STS telles que AssumeRoleou GetFederationToken.

Amazon EKS est compatible avec l'utilisation des informations d'identification temporaires.

Rôles liés à un service

Les rôles liés aux AWS services permettent aux services d'accéder aux ressources d'autres services pour effectuer une action en votre nom. Les rôles liés à un service s'affichent dans votre compte IAM

et sont la propriété du service. Un administrateur peut afficher les autorisations des rôles liés à un service, mais ne peut pas les modifier.

Amazon EKS prend en charge les rôles liés à un service. Pour plus d'informations sur la création ou la gestion des rôles liés à un service Amazon EKS, consultez the section called “Rôles liés à un service”.

Rôles du service

Cette fonction permet à un service d'endosser une fonction du service en votre nom. Ce rôle autorise le service à accéder à des ressources d'autres services pour effectuer une action en votre nom. Les rôles de service s'affichent dans votre compte IAM et sont la propriété du compte. Cela signifie qu'un administrateur IAM peut modifier les autorisations associées à ce rôle. Toutefois, une telle action peut perturber le bon fonctionnement du service.

Amazon EKS prend en charge les rôles de service. Pour plus d'informations, consultez the section called “Rôle IAM du cluster” et the section called “Rôle IAM de nœud”.

Choix d'un rôle IAM dans Amazon EKS

Lorsque vous créez une ressource de cluster dans Amazon EKS, vous devez choisir un rôle pour permettre à Amazon EKS d'accéder à plusieurs autres AWS ressources en votre nom. Si vous avez déjà créé un rôle de service, Amazon EKS vous propose une liste de rôles parmi lesquels choisir. Il est important de choisir un rôle auquel les politiques gérées par Amazon EKS sont attachées. Pour plus d'informations, consultez the section called “Recherche d'un rôle de cluster existant” et the section called “Recherche d'un rôle de nœud existant”.

Exemples de politiques basées sur l'identité d'Amazon EKS

Par défaut, les utilisateurs et les rôles IAM n'ont pas l'autorisation de créer ou de modifier des ressources Amazon EKS. Ils ne peuvent pas non plus effectuer de tâches à l'aide de la AWS Management Console AWS CLI ou de AWS l'API. Un administrateur IAM doit créer des politiques IAM autorisant les utilisateurs et les rôles à exécuter des opérations d'API spécifiques sur les ressources spécifiées dont ils ont besoin. Il doit ensuite attacher ces politiques aux utilisateurs ou aux groupes IAM ayant besoin de ces autorisations.

Pour apprendre à créer une politique basée sur l'identité IAM à l'aide de ces exemples de documents de politique JSON, veuillez consulter Création de politiques dans l'onglet JSON dans le Guide de l'utilisateur IAM.

Lorsque vous créez un cluster Amazon EKS, le principal IAM qui crée le cluster se voit automatiquement accorder des autorisations `system:masters` dans la configuration du contrôle d'accès basé sur les rôles (RBAC) du cluster dans le plan de contrôle Amazon EKS. Ce principal n'apparaît dans aucune configuration visible. Souvenez-vous donc du principal qui a créé le cluster à l'origine. Pour accorder à d'autres principaux IAM la possibilité d'interagir avec votre cluster, modifiez le `aws-auth ConfigMap` dans Kubernetes et créez un `rolebinding` ou `clusterrolebinding` Kubernetes associé au nom d'un `group` que vous spécifiez dans le `aws-auth ConfigMap`.

Pour plus d'informations sur l'utilisation du ConfigMap, consultezthe section called "Accès à l'API Kubernetes".

Rubriques

- Bonnes pratiques en matière de politiques
- Utilisation de la console Amazon EKS
- Autorisation des utilisateurs IAM à afficher leurs propres autorisations
- Création d'un cluster Kubernetes sur le cloud AWS
- Créer un cluster Kubernetes local sur un Outpost
- Mise à jour d'un cluster Kubernetes
- Affichage de la liste ou description de tous les clusters

Bonnes pratiques en matière de politiques

Les politiques basées sur l'identité déterminent si une personne peut créer, consulter ou supprimer des ressources Amazon EKS dans votre compte. Ces actions peuvent entraîner des frais pour votre AWS compte. Lorsque vous créez ou modifiez des politiques basées sur l'identité, suivez ces instructions et recommandations :

- Commencez AWS par les politiques gérées et passez aux autorisations du moindre privilège : pour commencer à accorder des autorisations à vos utilisateurs et à vos charges de travail, utilisez les politiques AWS gérées qui accordent des autorisations pour de nombreux cas d'utilisation courants. Ils sont disponibles dans votre AWS compte. Nous vous recommandons de réduire davantage les autorisations en définissant des politiques gérées par les AWS clients spécifiques à vos cas d'utilisation. Pour plus d'informations, consultez politiques gérées par AWS ou politiques gérées par AWS pour les activités professionnelles dans le Guide de l'utilisateur IAM.
- Accordez les autorisations de moindre privilège : lorsque vous définissez des autorisations avec des politiques IAM, accordez uniquement les autorisations nécessaires à l'exécution d'une seule

tâche. Pour ce faire, vous définissez les actions qui peuvent être entreprises sur des ressources spécifiques dans des conditions spécifiques, également appelées autorisations de moindre privilège. Pour plus d'informations sur l'utilisation d'IAM pour appliquer des autorisations, consultez politiques et autorisations dans IAM dans le Guide de l'utilisateur IAM.

- Utilisez des conditions dans les politiques IAM pour restreindre davantage l'accès : vous pouvez ajouter une condition à vos politiques afin de limiter l'accès aux actions et aux ressources. Par exemple, vous pouvez écrire une condition de politique pour spécifier que toutes les demandes doivent être envoyées via SSL. Vous pouvez également utiliser des conditions pour accorder l'accès aux actions de service si elles sont utilisées par le biais d'un AWS service spécifique, tel que AWS CloudFormation. Pour plus d'informations, consultez Conditions pour éléments de politique JSON IAM dans le Guide de l'utilisateur IAM.
- Utilisez l'Analyseur d'accès IAM pour valider vos politiques IAM afin de garantir des autorisations sécurisées et fonctionnelles : l'Analyseur d'accès IAM valide les politiques nouvelles et existantes de manière à ce que les politiques IAM respectent le langage de politique IAM (JSON) et les bonnes pratiques IAM. IAM Access Analyzer fournit plus de 100 vérifications de politiques et des recommandations exploitables pour vous aider à créer des politiques sécurisées et fonctionnelles. Pour plus d'informations, consultez Validation de politique IAM Access Analyzer dans le Guide de l'utilisateur IAM.
- Exiger l'authentification multifactorielle (MFA) : si vous avez un scénario qui nécessite des utilisateurs IAM ou un utilisateur root dans AWS votre compte, activez l'authentification MFA pour une sécurité accrue. Pour exiger le MFA lorsque des opérations d'API sont appelées, ajoutez des conditions MFA à vos politiques. Pour plus d'informations, consultez Configuration de l'accès aux API protégé par MFA dans le Guide de l'utilisateur IAM.

Pour plus d'informations sur les bonnes pratiques dans IAM, consultez Bonnes pratiques de sécurité dans IAM dans le Guide de l'utilisateur IAM.

Utilisation de la console Amazon EKS

Pour accéder à la console Amazon EKS, un principal IAM doit disposer d'un ensemble minimum d'autorisations. Ces autorisations permettent au principal de répertorier et de consulter les informations relatives aux ressources Amazon EKS de votre AWS compte. Si vous créez une politique basée sur l'identité qui est plus restrictive que les autorisations minimales requises, la console ne fonctionnera pas comme prévu pour les principaux auxquels cette politique est associée.

Pour garantir que vos principaux IAM puissent toujours utiliser la console Amazon EKS, créez une politique avec votre propre nom unique, par exemple `AmazonEKSAdminPolicy`. Attachez

la politique aux principaux. Pour plus d'informations, consultez la rubrique Ajout et suppression d'autorisations basées sur l'identité IAM du Guide de l'utilisateur IAM.

> **Important**
>
> L'exemple de politique suivant permet à un principal d'afficher des informations dans l'onglet Configuration sur la console. Pour afficher les informations dans les onglets Vue d'ensemble et Ressources du AWS Management Console, le principal doit également disposer des autorisations Kubernetes. Pour de plus amples informations, veuillez consulter the section called "Autorisations requises".

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "eks:*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "eks.amazonaws.com"
                }
            }
        }
    ]
}
```

Il n'est pas nécessaire d'accorder des autorisations de console minimales aux principaux qui appellent uniquement la AWS CLI ou l' AWS API. Autorisez plutôt l'accès uniquement aux actions qui correspondent à l'opération API que vous essayez d'effectuer.

Autorisation des utilisateurs IAM à afficher leurs propres autorisations

Cet exemple montre comment créer une politique qui permet aux utilisateurs IAM d'afficher les politiques en ligne et gérées attachées à leur identité d'utilisateur. Cette politique inclut les autorisations permettant d'effectuer cette action sur la console ou par programmation à l'aide de la AWS CLI ou AWS de l'API.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Création d'un cluster Kubernetes sur le cloud AWS

Cet exemple de politique inclut les autorisations minimales requises pour créer un cluster Amazon EKS nommé *my-cluster* dans la *us-west-2* AWS région. Vous pouvez remplacer la AWS région par la AWS région dans laquelle vous souhaitez créer un cluster. Si vous voyez un avertissement indiquant que les actions de votre politique ne prennent pas en charge les autorisations au niveau des ressources et nécessitent que vous choisissiez **`All resources`** de le faire AWS Management Console, il peut être ignoré en toute sécurité. Si votre compte possède déjà le rôle *AWSServiceRoleForAmazonEKS*, vous pouvez supprimer l'action `iam:CreateServiceLinkedRole` de la politique. Si vous avez déjà créé un cluster Amazon EKS dans votre compte, ce rôle existe déjà, sauf si vous l'avez supprimé.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "eks:CreateCluster",
            "Resource": "arn:aws:eks:us-west-2:111122223333:cluster/my-cluster"
        },
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::111122223333:role/aws-service-role/
eks.amazonaws.com/AWSServiceRoleForAmazonEKS",
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "iam:AWSServiceName": "eks"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::111122223333:role/cluster-role-name"
        }
    ]
}
```

Créer un cluster Kubernetes local sur un Outpost

Cet exemple de politique inclut les autorisations minimales requises pour créer un cluster local Amazon EKS nommé *my-cluster* sur un avant-poste de la *us-west-2* AWS région. Vous pouvez remplacer la AWS région par la AWS région dans laquelle vous souhaitez créer un cluster. Si vous voyez un avertissement indiquant que les actions de votre politique ne prennent pas en charge les autorisations au niveau des ressources et nécessitent que vous choisissiez **`All resources`** de le faire AWS Management Console, il peut être ignoré en toute sécurité. Si votre compte possède déjà le rôle `AWSServiceRoleForAmazonEKSLocalOutpost`, vous pouvez supprimer l'action `iam:CreateServiceLinkedRole` de la politique. Si vous avez déjà créé un cluster Amazon EKS local sur un Outpost dans votre compte, ce rôle existe déjà, sauf si vous l'avez supprimé.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "eks:CreateCluster",
            "Resource": "arn:aws:eks:us-west-2:111122223333:cluster/my-cluster"
        },
        {
            "Action": [
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcs",
                "iam:GetRole"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::111122223333:role/aws-service-role/outposts.eks-
local.amazonaws.com/AWSServiceRoleForAmazonEKSLocalOutpost"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole",
                "iam:ListAttachedRolePolicies"
            ],
            "Resource": "arn:aws:iam::111122223333:role/cluster-role-name"
```

```
        },
        {
            "Action": [
                "iam:CreateInstanceProfile",
                "iam:TagInstanceProfile",
                "iam:AddRoleToInstanceProfile",
                "iam:GetInstanceProfile",
                "iam:DeleteInstanceProfile",
                "iam:RemoveRoleFromInstanceProfile"
            ],
            "Resource": "arn:aws:iam::*:instance-profile/eks-local-*",
            "Effect": "Allow"
        }
    ]
}
```

Mise à jour d'un cluster Kubernetes

Cet exemple de politique inclut l'autorisation minimale requise pour mettre à jour un cluster nommé *`my-cluster`* dans la région us-west-2. AWS

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "eks:UpdateClusterVersion",
            "Resource": "arn:aws:eks:us-west-2:111122223333:cluster/my-cluster"
        }
    ]
}
```

Affichage de la liste ou description de tous les clusters

Cet exemple de politique inclut les autorisations minimales requises pour répertorier et décrire tous les clusters de votre compte. Un principal IAM doit être capable de répertorier et de décrire les clusters pour utiliser la commande `update-kubeconfig` AWS CLI.

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

```
            "Effect": "Allow",
            "Action": [
                "eks:DescribeCluster",
                "eks:ListClusters"
            ],
            "Resource": "*"
        }
    ]
}
```

Utilisation des rôles liés à un service pour Amazon EKS

Amazon Elastic Kubernetes Service utilise les rôles liés au service Gestion des identités et des accès AWS (IAM). Un rôle lié à un service est un type unique de rôle IAM lié directement à Amazon EKS. Les rôles liés à un service sont prédéfinis par Amazon EKS et incluent toutes les autorisations requises par le service pour appeler d'autres services AWS en votre nom.

Rubriques

- Utilisation de rôles pour les clusters Amazon EKS
- Utilisation de rôles pour les groupes de nœuds Amazon EKS
- Utilisation de rôles pour les profils Amazon EKS Fargate
- Utilisation de rôles pour connecter un cluster Kubernetes à Amazon EKS
- Utilisation de rôles pour les clusters locaux Amazon EKS sur Outpost
- Utilisation des rôles pour le tableau de bord Amazon EKS

Utilisation de rôles pour les clusters Amazon EKS

Amazon Elastic Kubernetes Service utilise les rôles liés au service Gestion des identités et des accès AWS (IAM). Un rôle lié à un service est un type unique de rôle IAM lié directement à Amazon EKS. Les rôles liés à un service sont prédéfinis par Amazon EKS et incluent toutes les autorisations requises par le service pour appeler d'autres services AWS en votre nom.

Un rôle lié à un service facilite la configuration d'Amazon EKS, car vous n'avez pas à ajouter manuellement les autorisations nécessaires. Amazon EKS définit les autorisations de ses rôles liés à un service ; sauf définition contraire, seul Amazon EKS peut endosser ses rôles. Les autorisations définies comprennent la politique d'approbation et la politique d'autorisation. De plus, cette politique d'autorisation ne peut pas être attachée à une autre entité IAM.

Vous pouvez supprimer un rôle lié à un service uniquement après la suppression préalable de ses ressources connexes. Vos ressources Amazon EKS sont ainsi protégées, car vous ne pouvez pas involontairement supprimer l'autorisation d'accéder aux ressources.

Pour plus d'informations sur les autres services prenant en charge les rôles liés à un service, consultez les services AWS opérationnels avec IAM et recherchez les services présentant la mention Yes (Oui) dans la colonne Service-linked roles (Rôles liés à un service). Choisissez un Yes (oui) ayant un lien permettant de consulter les détails du rôle pour ce service.

Autorisations du rôle lié à un service pour Amazon EKS

Amazon EKS utilise le rôle lié à un service nommé `AWSServiceRoleForAmazonEKS`. Ce rôle permet à Amazon EKS de gérer les clusters de votre compte. Les politiques attachées permettent au rôle de gérer les ressources suivantes : interfaces réseau, groupes de sécurité, journaux et VPC.

> **Note**
>
> Le rôle lié à un service `AWSServiceRoleForAmazonEKS` est distinct du rôle requis pour la création de cluster. Pour de plus amples informations, consultez the section called "Rôle IAM du cluster".

Le rôle lié à un service `AWSServiceRoleForAmazonEKS` approuve les services suivants pour endosser le rôle :

- `eks.amazonaws.com`

La politique d'autorisations liée au rôle permet à Amazon EKS de réaliser les actions suivantes sur les ressources spécifiées :

- AmazonEKSServiceRolePolicy

Vous devez configurer les autorisations de manière à permettre à une entité IAM (comme un utilisateur, un groupe ou un rôle) de créer, modifier ou supprimer un rôle lié à un service. Pour plus d'informations, consultez Autorisations de rôles liés à un service dans le Guide de l'utilisateur IAM.

Création d'un rôle lié à un service pour Amazon EKS

Vous n'avez pas besoin de créer manuellement un rôle lié à un service. Lorsque vous créez un cluster dans la AWS Management Console, AWS CLI ou l'AWS API, Amazon EKS crée le rôle lié à un service pour vous.

Si vous supprimez ce rôle lié à un service et que vous avez ensuite besoin de le recréer, vous pouvez utiliser la même procédure pour recréer le rôle dans votre compte. Lorsque vous créez un cluster, Amazon EKS crée automatiquement le rôle lié au service à nouveau.

Modification d'un rôle lié à un service pour Amazon EKS

Amazon EKS ne vous autorise pas à modifier le rôle lié à un service `AWSServiceRoleForAmazonEKS`. Une fois que vous avez créé un rôle lié à un service, vous ne pouvez pas changer le nom du rôle, car plusieurs entités peuvent faire référence à ce rôle. Néanmoins, vous pouvez modifier la description du rôle à l'aide d'IAM. Pour plus d'informations, consultez Modification d'un rôle lié à un service dans le IAM Guide de l'utilisateur.

Suppression d'un rôle lié à un service pour Amazon EKS

Si vous n'avez plus besoin d'utiliser une fonctionnalité ou un service qui nécessite un rôle lié à un service, nous vous recommandons de supprimer ce rôle. De cette façon, vous n'avez aucune entité inutilisée qui n'est pas surveillée ou gérée activement. Cependant, vous devez nettoyer votre rôle lié à un service avant de pouvoir le supprimer manuellement.

Nettoyer un rôle lié à un service

Avant de pouvoir utiliser IAM pour supprimer un rôle lié à un service, vous devez supprimer toutes les ressources utilisées par le rôle.

> **Note**
>
> Si le service Amazon EKS utilise le rôle lorsque vous essayez de supprimer les ressources, la suppression peut échouer. Si cela se produit, patientez quelques minutes et réessayez.

1. Ouvrez la console Amazon EKS.
2. Dans le volet de navigation de gauche, choisissez Clusters.

3. Si votre cluster possède des groupes de nœuds ou des profils Fargate, vous devez les supprimer avant de pouvoir supprimer le cluster. Pour plus d'informations, consultez the section called "Suppression" et the section called "Supprimer les profils".
4. Dans la page Clusters, choisissez le cluster à supprimer et cliquez sur Delete (Supprimer).
5. Tapez le nom du cluster dans la fenêtre de confirmation de suppression, puis choisissez Delete (Supprimer).
6. Répétez cette procédure pour tous les autres clusters de votre compte. Attendez la fin de toutes les opérations de suppression.

Suppression manuelle du rôle lié au service

Utilisez la console IAM, l'interface de ligne de commande AWS ou l'API AWS pour supprimer le rôle lié à un service `AWSServiceRoleForAmazonEKS`. Pour plus d'informations, consultez Suppression d'un rôle lié à un service dans le Guide de l'utilisateur IAM.

Régions prises en charge pour les rôles liés à un service Amazon EKS

Amazon EKS prend en charge l'utilisation des rôles liés à un service dans toutes les régions où le service est disponible. Pour plus d'informations, consultez Amazon EKS endpoints and quotas (langue française non garantie).

Utilisation de rôles pour les groupes de nœuds Amazon EKS

Amazon EKS utilise les rôles liés au service de la gestion des identités et des accès AWS (IAM). Un rôle lié à un service est un type unique de rôle IAM lié directement à Amazon EKS. Les rôles liés à un service sont prédéfinis par Amazon EKS et incluent toutes les autorisations requises par le service pour appeler d'autres services AWS en votre nom.

Un rôle lié à un service facilite la configuration d'Amazon EKS, car vous n'avez pas à ajouter manuellement les autorisations nécessaires. Amazon EKS définit les autorisations de ses rôles liés à un service ; sauf définition contraire, seul Amazon EKS peut endosser ses rôles. Les autorisations définies comprennent la politique d'approbation et la politique d'autorisation. De plus, cette politique d'autorisation ne peut pas être attachée à une autre entité IAM.

Vous pouvez supprimer un rôle lié à un service uniquement après la suppression préalable de ses ressources connexes. Vos ressources Amazon EKS sont ainsi protégées, car vous ne pouvez pas involontairement supprimer l'autorisation d'accéder aux ressources.

Pour plus d'informations sur les autres services prenant en charge les rôles liés à un service, consultez les services AWS opérationnels avec IAM et recherchez les services présentant la mention Yes (Oui) dans la colonne Service-linked roles (Rôles liés à un service). Choisissez un Yes (oui) ayant un lien permettant de consulter les détails du rôle pour ce service.

Autorisations du rôle lié à un service pour Amazon EKS

Amazon EKS utilise le rôle lié à un service nommé `AWSServiceRoleForAmazonEKSNodegroup`. Ce rôle permet à Amazon EKS de gérer les groupes de nœuds de votre compte. La politique `AWSServiceRoleForAmazonEKSNodegroup` attachée permet au rôle de gérer les ressources suivantes : groupes Auto Scaling, groupes de sécurité, modèles de lancement et profils d'instance IAM. Pour de plus amples informations, consultez the section called "AWS politique gérée : AWSService RoleForAmazon EKSNodegroup".

Le rôle lié à un service `AWSServiceRoleForAmazonEKSNodegroup` approuve les services suivants pour endosser le rôle :

- `eks-nodegroup.amazonaws.com`

La politique d'autorisations liée au rôle permet à Amazon EKS de réaliser les actions suivantes sur les ressources spécifiées :

- AWSServiceRoleForAmazonEKSNodegroup

Vous devez configurer les autorisations de manière à permettre à une entité IAM (comme un utilisateur, un groupe ou un rôle) de créer, modifier ou supprimer un rôle lié à un service. Pour plus d'informations, consultez Autorisations de rôles liés à un service dans le Guide de l'utilisateur IAM.

Création d'un rôle lié à un service pour Amazon EKS

Vous n'avez pas besoin de créer manuellement un rôle lié à un service. Lorsque vous créez un groupe de nœuds dans la AWS Management Console, l'AWS CLI ou l'API AWS, Amazon EKS crée le rôle lié à un service pour vous.

> **Important**
>
> Ce rôle lié à un service peut apparaître dans votre compte si vous avez effectué une action dans un autre service qui utilise les fonctions prises en charge par ce rôle. Si vous utilisiez le service Amazon EKS avant le 1er janvier 2017, date à laquelle il a

commencé à prendre en charge les rôles liés à un service, Amazon EKS a créé le rôle AWSServiceRoleForAmazonEKSNodegroup dans votre compte. Pour en savoir plus, consultez Un nouveau rôle est apparu dans mon compte IAM.

Création d'un rôle lié à un service dans Amazon EKS (API AWS)

Vous n'avez pas besoin de créer manuellement un rôle lié à un service. Lorsque vous créez un groupe de nœuds géré dans la AWS Management Console, l'AWS CLI ou l'API AWS, Amazon EKS crée le rôle lié à un service pour vous.

Si vous supprimez ce rôle lié à un service et que vous avez ensuite besoin de le recréer, vous pouvez utiliser la même procédure pour recréer le rôle dans votre compte. Lorsque vous créez un autre groupe de nœuds gérés, Amazon EKS crée automatiquement le rôle lié au service.

Modification d'un rôle lié à un service pour Amazon EKS

Amazon EKS ne vous autorise pas à modifier le rôle lié à un service `AWSServiceRoleForAmazonEKSNodegroup`. Une fois que vous avez créé un rôle lié à un service, vous ne pouvez pas changer le nom du rôle, car plusieurs entités peuvent faire référence à ce rôle. Néanmoins, vous pouvez modifier la description du rôle à l'aide d'IAM. Pour plus d'informations, consultez Modification d'un rôle lié à un service dans le IAM Guide de l'utilisateur.

Suppression d'un rôle lié à un service pour Amazon EKS

Si vous n'avez plus besoin d'utiliser une fonctionnalité ou un service qui nécessite un rôle lié à un service, nous vous recommandons de supprimer ce rôle. De cette façon, vous n'avez aucune entité inutilisée qui n'est pas surveillée ou gérée activement. Cependant, vous devez nettoyer votre rôle lié à un service avant de pouvoir le supprimer manuellement.

Nettoyer un rôle lié à un service

Avant de pouvoir utiliser IAM pour supprimer un rôle lié à un service, vous devez supprimer toutes les ressources utilisées par le rôle.

> **Note**
>
> Si le service Amazon EKS utilise le rôle lorsque vous essayez de supprimer les ressources, la suppression peut échouer. Si cela se produit, patientez quelques minutes et réessayez.

1. Ouvrez la console Amazon EKS.
2. Dans le volet de navigation de gauche, choisissez Clusters.
3. Sélectionnez l'onglet Compute (Calcul).
4. Dans la section Node Groups (Groupes de nœuds), choisissez le groupe de nœuds à supprimer.
5. Tapez le nom du groupe de nœuds dans la fenêtre de confirmation de suppression, puis choisissez Delete (Supprimer).
6. Répétez cette procédure pour tous les autres groupes de nœuds du cluster. Attendez la fin de toutes les opérations de suppression.

Suppression manuelle du rôle lié au service

Utilisez la console IAM, l'interface de ligne de commande AWS ou l'API AWS pour supprimer le rôle lié à un service `AWSServiceRoleForAmazonEKSNodegroup`. Pour plus d'informations, consultez Suppression d'un rôle lié à un service dans le Guide de l'utilisateur IAM.

Régions prises en charge pour les rôles liés à un service Amazon EKS

Amazon EKS prend en charge l'utilisation des rôles liés à un service dans toutes les régions où le service est disponible. Pour plus d'informations, consultez Amazon EKS endpoints and quotas (langue française non garantie).

Utilisation de rôles pour les profils Amazon EKS Fargate

Amazon EKS utilise les rôles liés au service de la gestion des identités et des accès AWS (IAM). Un rôle lié à un service est un type unique de rôle IAM lié directement à Amazon EKS. Les rôles liés à un service sont prédéfinis par Amazon EKS et incluent toutes les autorisations requises par le service pour appeler d'autres services AWS en votre nom.

Un rôle lié à un service facilite la configuration d'Amazon EKS, car vous n'avez pas à ajouter manuellement les autorisations nécessaires. Amazon EKS définit les autorisations de ses rôles liés à un service ; sauf définition contraire, seul Amazon EKS peut endosser ses rôles. Les autorisations définies comprennent la politique d'approbation et la politique d'autorisation. De plus, cette politique d'autorisation ne peut pas être attachée à une autre entité IAM.

Vous pouvez supprimer un rôle lié à un service uniquement après la suppression préalable de ses ressources connexes. Vos ressources Amazon EKS sont ainsi protégées, car vous ne pouvez pas involontairement supprimer l'autorisation d'accéder aux ressources.

Pour plus d'informations sur les autres services prenant en charge les rôles liés à un service, consultez les services AWS opérationnels avec IAM et recherchez les services présentant la mention Yes (Oui) dans la colonne Service-linked roles (Rôles liés à un service). Choisissez un Yes (oui) ayant un lien permettant de consulter les détails du rôle pour ce service.

Autorisations du rôle lié à un service pour Amazon EKS

Amazon EKS utilise le rôle lié à un service nommé `AWSServiceRoleForAmazonEKSForFargate`. Ce rôle permet à Amazon EKS Fargate de configurer le réseau VPC requis pour les pods Fargate. Les politiques attachées permettent au rôle de créer et de supprimer des interfaces réseau Elastic et de décrire les interfaces et les ressources réseau Elastic.

Le rôle lié à un service `AWSServiceRoleForAmazonEKSForFargate` approuve les services suivants pour endosser le rôle :

- `eks-fargate.amazonaws.com`

La politique d'autorisations liée au rôle permet à Amazon EKS de réaliser les actions suivantes sur les ressources spécifiées :

- AmazonEKSForFargateServiceRolePolicy

Vous devez configurer les autorisations de manière à permettre à une entité IAM (comme un utilisateur, un groupe ou un rôle) de créer, modifier ou supprimer un rôle lié à un service. Pour plus d'informations, consultez Autorisations de rôles liés à un service dans le Guide de l'utilisateur IAM.

Création d'un rôle lié à un service pour Amazon EKS

Vous n'avez pas besoin de créer manuellement un rôle lié à un service. Lorsque vous créez un profil Fargate dans la AWS Management Console, l'AWS CLI ou l'API AWS, Amazon EKS crée le rôle lié à un service pour vous.

> **Important**
>
> Ce rôle lié à un service peut apparaître dans votre compte si vous avez effectué une action dans un autre service qui utilise les fonctions prises en charge par ce rôle. Si vous utilisiez le service Amazon EKS avant le 13 décembre 2019, date à laquelle il a commencé à prendre en charge les rôles liés à un service, Amazon EKS a créé le rôle

AWSServiceRoleForAmazonEKSForFargate dans votre compte. Pour plus d'informations, consultez Un nouveau rôle est apparu dans mon compte IAM.

Création d'un rôle lié à un service dans Amazon EKS (API AWS)

Vous n'avez pas besoin de créer manuellement un rôle lié à un service. Lorsque vous créez un profil Fargate dans la AWS Management Console, l'AWS CLI ou l'API AWS, Amazon EKS crée le rôle lié à un service pour vous.

Si vous supprimez ce rôle lié à un service et que vous avez ensuite besoin de le recréer, vous pouvez utiliser la même procédure pour recréer le rôle dans votre compte. Lorsque vous créez un autre groupe de nœuds gérés, Amazon EKS crée automatiquement le rôle lié au service.

Modification d'un rôle lié à un service pour Amazon EKS

Amazon EKS ne vous autorise pas à modifier le rôle lié à un service `AWSServiceRoleForAmazonEKSForFargate`. Une fois que vous avez créé un rôle lié à un service, vous ne pouvez pas changer le nom du rôle, car plusieurs entités peuvent faire référence à ce rôle. Néanmoins, vous pouvez modifier la description du rôle à l'aide d'IAM. Pour plus d'informations, consultez Modification d'un rôle lié à un service dans le IAM Guide de l'utilisateur.

Suppression d'un rôle lié à un service pour Amazon EKS

Si vous n'avez plus besoin d'utiliser une fonctionnalité ou un service qui nécessite un rôle lié à un service, nous vous recommandons de supprimer ce rôle. De cette façon, vous n'avez aucune entité inutilisée qui n'est pas surveillée ou gérée activement. Cependant, vous devez nettoyer votre rôle lié à un service avant de pouvoir le supprimer manuellement.

Nettoyer un rôle lié à un service

Avant de pouvoir utiliser IAM pour supprimer un rôle lié à un service, vous devez supprimer toutes les ressources utilisées par le rôle.

> **Note**
>
> Si le service Amazon EKS utilise le rôle lorsque vous essayez de supprimer les ressources, la suppression peut échouer. Si cela se produit, patientez quelques minutes et réessayez.

1. Ouvrez la console Amazon EKS.

2. Dans le volet de navigation de gauche, choisissez Clusters.
3. Sur la page Clusters, sélectionnez votre cluster.
4. Sélectionnez l'onglet Compute (Calcul).
5. S'il existe des profils Fargate dans la section Profils Fargate, sélectionnez chacun individuellement, puis choisissez Delete (Supprimer).
6. Tapez le nom du profile dans la fenêtre de confirmation de suppression, puis choisissez Delete (Supprimer).
7. Répétez cette procédure pour tous les autres profils Fargate dans le cluster et pour tous les autres clusters de votre compte.

Suppression manuelle du rôle lié au service

Utilisez la console IAM, l'AWS CLI ou l'API AWS pour supprimer le rôle lié à un service AWSServiceRoleForAmazonEKSForFargate. Pour plus d'informations, consultez Suppression d'un rôle lié à un service dans le Guide de l'utilisateur IAM.

Régions prises en charge pour les rôles liés à un service Amazon EKS

Amazon EKS prend en charge l'utilisation des rôles liés à un service dans toutes les régions où le service est disponible. Pour plus d'informations, consultez Amazon EKS endpoints and quotas (langue française non garantie).

Utilisation de rôles pour connecter un cluster Kubernetes à Amazon EKS

Amazon EKS utilise des AWS rôles liés au service Identity and Access Management (IAM). Un rôle lié à un service est un type unique de rôle IAM lié directement à Amazon EKS. Les rôles liés à un service sont prédéfinis par Amazon EKS et incluent toutes les autorisations requises par le service pour appeler d'autres AWS services en votre nom.

Un rôle lié à un service facilite la configuration d'Amazon EKS, car vous n'avez pas à ajouter manuellement les autorisations nécessaires. Amazon EKS définit les autorisations de ses rôles liés à un service ; sauf définition contraire, seul Amazon EKS peut endosser ses rôles. Les autorisations définies comprennent la politique d'approbation et la politique d'autorisation. De plus, cette politique d'autorisation ne peut pas être attachée à une autre entité IAM.

Vous pouvez supprimer un rôle lié à un service uniquement après la suppression préalable de ses ressources connexes. Vos ressources Amazon EKS sont ainsi protégées, car vous ne pouvez pas involontairement supprimer l'autorisation d'accéder aux ressources.

Pour plus d'informations sur les autres services prenant en charge les rôles liés à un service, consultez les services AWS opérationnels avec IAM et recherchez les services présentant la mention Yes (Oui) dans la colonne Service-linked roles (Rôles liés à un service). Choisissez un Oui ayant un lien permettant de consulter les détails du rôle pour ce service.

Autorisations du rôle lié à un service pour Amazon EKS

Amazon EKS utilise le rôle lié à un service nommé `AWSServiceRoleForAmazonEKSConnector`. Ce rôle permet à Amazon EKS de connecter des clusters Kubernetes. Les politiques associées permettent au rôle de gérer les ressources nécessaires pour se connecter à votre cluster Kubernetes enregistré.

Le rôle lié à un service `AWSServiceRoleForAmazonEKSConnector` approuve les services suivants pour endosser le rôle :

- `eks-connector.amazonaws.com`

La politique d'autorisations liée au rôle permet à Amazon EKS de réaliser les actions suivantes sur les ressources spécifiées :

- AmazonEKSConnectorServiceRolePolicy

Vous devez configurer les autorisations de manière à permettre à une entité IAM (comme un utilisateur, un groupe ou un rôle) de créer, modifier ou supprimer un rôle lié à un service. Pour plus d'informations, consultez Autorisations de rôles liés à un service dans le Guide de l'utilisateur IAM.

Ce rôle utilise les autorisations SSM (Systems Manager) pour établir des connexions sécurisées et gérer les clusters Kubernetes connectés.

Création d'un rôle lié à un service pour Amazon EKS

Il n'est pas nécessaire de créer manuellement un rôle lié à un service pour connecter un cluster. Lorsque vous connectez un cluster dans la AWS Management Console AWS CLI ou l' AWS API`eksctl`, Amazon EKS crée le rôle lié au service pour vous.

Si vous supprimez ce rôle lié à un service et que vous avez ensuite besoin de le recréer, vous pouvez utiliser la même procédure pour recréer le rôle dans votre compte. Lorsque vous connectez un cluster, Amazon EKS crée à nouveau le rôle lié à un service pour vous.

Modification d'un rôle lié à un service pour Amazon EKS

Amazon EKS ne vous autorise pas à modifier le rôle lié à un service `AWSServiceRoleForAmazonEKSConnector`. Après avoir créé un rôle lié à un service, vous ne pouvez pas changer le nom du rôle, car plusieurs entités peuvent faire référence à ce rôle. Néanmoins, vous pouvez modifier la description du rôle à l'aide d'IAM. Pour plus d'informations, consultez Modification d'un rôle lié à un service dans le Guide de l'utilisateur IAM.

Suppression d'un rôle lié à un service pour Amazon EKS

Si vous n'avez plus besoin d'utiliser une fonctionnalité ou un service qui nécessite un rôle lié à un service, nous vous recommandons de supprimer ce rôle. De cette façon, vous n'avez aucune entité inutilisée qui n'est pas surveillée ou gérée activement. Cependant, vous devez nettoyer votre rôle lié à un service avant de pouvoir le supprimer manuellement.

Nettoyage d'un rôle lié à un service

Avant de pouvoir utiliser IAM pour supprimer un rôle lié à un service, vous devez supprimer toutes les ressources utilisées par le rôle.

> **Note**
>
> Si le service Amazon EKS utilise le rôle lorsque vous essayez de supprimer les ressources, la suppression peut échouer. Si cela se produit, patientez quelques minutes et réessayez.

1. Ouvrez la console Amazon EKS.
2. Dans le volet de navigation de gauche, choisissez Clusters.
3. Sur la page Clusters, sélectionnez votre cluster.
4. Sélectionnez les onglets Annuler l'inscription puis Ok.

Suppression manuelle du rôle lié au service

Utilisez la console IAM, la AWS CLI ou l' AWS API pour supprimer le rôle lié au AWSService RoleForAmazon EKSConnector service. Pour plus d'informations, consultez la section Suppression d'un rôle lié à un service dans le Guide de l'utilisateur IAM.

Utilisation de rôles pour les clusters locaux Amazon EKS sur Outpost

Amazon Elastic Kubernetes Service utilise les rôles liés au service Gestion des identités et des accès AWS (IAM). Un rôle lié à un service est un type unique de rôle IAM lié directement à Amazon EKS. Les rôles liés à un service sont prédéfinis par Amazon EKS et incluent toutes les autorisations requises par le service pour appeler d'autres services AWS en votre nom.

Un rôle lié à un service facilite la configuration d'Amazon EKS, car vous n'avez pas à ajouter manuellement les autorisations nécessaires. Amazon EKS définit les autorisations de ses rôles liés à un service ; sauf définition contraire, seul Amazon EKS peut endosser ses rôles. Les autorisations définies comprennent la politique d'approbation et la politique d'autorisation. De plus, cette politique d'autorisation ne peut pas être attachée à une autre entité IAM.

Vous pouvez supprimer un rôle lié à un service uniquement après la suppression préalable de ses ressources connexes. Vos ressources Amazon EKS sont ainsi protégées, car vous ne pouvez pas involontairement supprimer l'autorisation d'accéder aux ressources.

Pour plus d'informations sur les autres services prenant en charge les rôles liés à un service, consultez les services AWS opérationnels avec IAM et recherchez les services présentant la mention Yes (Oui) dans la colonne Service-linked roles (Rôles liés à un service). Choisissez un Yes (oui) ayant un lien permettant de consulter les détails du rôle pour ce service.

Autorisations du rôle lié à un service pour Amazon EKS

Amazon EKS utilise le rôle lié à un service nommé `AWSServiceRoleForAmazonEKSLocalOutpost`. Ce rôle permet à Amazon EKS de gérer les clusters locaux de votre compte. Les politiques attachées permettent au rôle de gérer les ressources suivantes : interfaces réseau, groupes de sécurité, journaux et instances Amazon EC2.

> **Note**
>
> Le rôle lié à un service `AWSServiceRoleForAmazonEKSLocalOutpost` est distinct du rôle requis pour la création de cluster. Pour de plus amples informations, consultez the section called "Rôle IAM du cluster".

Le rôle lié à un service `AWSServiceRoleForAmazonEKSLocalOutpost` approuve les services suivants pour endosser le rôle :

- `outposts.eks-local.amazonaws.com`

La politique d'autorisations liée au rôle permet à Amazon EKS de réaliser les actions suivantes sur les ressources spécifiées :

- AmazonEKSServiceRolePolicy

Vous devez configurer les autorisations de manière à permettre à une entité IAM (comme un utilisateur, un groupe ou un rôle) de créer, modifier ou supprimer un rôle lié à un service. Pour plus d'informations, consultez Autorisations de rôles liés à un service dans le Guide de l'utilisateur IAM.

Création d'un rôle lié à un service pour Amazon EKS

Vous n'avez pas besoin de créer manuellement un rôle lié à un service. Lorsque vous créez un cluster dans la AWS Management Console, AWS CLI ou l'AWS API, Amazon EKS crée le rôle lié à un service pour vous.

Si vous supprimez ce rôle lié à un service et que vous avez ensuite besoin de le recréer, vous pouvez utiliser la même procédure pour recréer le rôle dans votre compte. Lorsque vous créez un cluster, Amazon EKS crée automatiquement le rôle lié au service à nouveau.

Modification d'un rôle lié à un service pour Amazon EKS

Amazon EKS ne vous autorise pas à modifier le rôle lié à un service `AWSServiceRoleForAmazonEKSLocalOutpost`. Une fois que vous avez créé un rôle lié à un service, vous ne pouvez plus modifier son nom, car diverses entités peuvent y faire référence. Néanmoins, vous pouvez modifier la description du rôle à l'aide d'IAM. Pour plus d'informations, consultez Modification d'un rôle lié à un service dans le IAM Guide de l'utilisateur.

Suppression d'un rôle lié à un service pour Amazon EKS

Si vous n'avez plus besoin d'utiliser une fonctionnalité ou un service qui nécessite un rôle lié à un service, nous vous recommandons de supprimer ce rôle. De cette façon, vous n'avez aucune entité inutilisée qui n'est pas surveillée ou gérée activement. Cependant, vous devez nettoyer votre rôle lié à un service avant de pouvoir le supprimer manuellement.

Nettoyer un rôle lié à un service

Avant de pouvoir utiliser IAM pour supprimer un rôle lié à un service, vous devez supprimer toutes les ressources utilisées par le rôle.

> **Note**
>
> Si le service Amazon EKS utilise le rôle lorsque vous essayez de supprimer les ressources, la suppression peut échouer. Si cela se produit, patientez quelques minutes et réessayez.

1. Ouvrez la console Amazon EKS.
2. Sélectionnez Amazon EKS Clusters (Clusters Amazon EKS) dans le panneau de navigation de gauche.
3. Si votre cluster possède des groupes de nœuds ou des profils Fargate, vous devez les supprimer avant de pouvoir supprimer le cluster. Pour plus d'informations, consultez the section called "Suppression" et the section called "Supprimer les profils".
4. Dans la page Clusters, choisissez le cluster à supprimer et cliquez sur Delete (Supprimer).
5. Tapez le nom du cluster dans la fenêtre de confirmation de suppression, puis choisissez Delete (Supprimer).
6. Répétez cette procédure pour tous les autres clusters de votre compte. Attendez la fin de toutes les opérations de suppression.

Suppression manuelle du rôle lié au service

Utilisez la console IAM, l'interface de ligne de commande AWS ou l'API AWS pour supprimer le rôle lié à un service `AWSServiceRoleForAmazonEKSLocalOutpost`. Pour plus d'informations, consultez Suppression d'un rôle lié à un service dans le Guide de l'utilisateur IAM.

Régions prises en charge pour les rôles liés à un service Amazon EKS

Amazon EKS prend en charge l'utilisation des rôles liés à un service dans toutes les régions où le service est disponible. Pour plus d'informations, consultez Amazon EKS endpoints and quotas (langue française non garantie).

Utilisation des rôles pour le tableau de bord Amazon EKS

Le tableau de bord Amazon EKS utilise ce rôle lié à un service pour agréger les informations sur les ressources du cluster provenant de plusieurs régions et comptes. Le tableau de bord s'intègre à AWS Organizations pour lire en toute sécurité les informations relatives à plusieurs comptes de votre organisation.

Pour en savoir plus sur le tableau de bord Amazon EKS, consultez the section called "Tableau de bord Amazon EKS".

Contexte

Amazon EKS utilise les rôles liés au service de la gestion des identités et des accès AWS (IAM). Un rôle lié à un service est un type unique de rôle IAM lié directement à Amazon EKS. Les rôles liés à un service sont prédéfinis par Amazon EKS et incluent toutes les autorisations requises par le service pour appeler d'autres services AWS en votre nom.

Un rôle lié à un service facilite la configuration d'Amazon EKS, car vous n'avez pas à ajouter manuellement les autorisations nécessaires. Amazon EKS définit les autorisations de ses rôles liés à un service ; sauf définition contraire, seul Amazon EKS peut endosser ses rôles. Les autorisations définies comprennent la politique d'approbation et la politique d'autorisation. De plus, cette politique d'autorisation ne peut pas être attachée à une autre entité IAM.

Vous pouvez supprimer un rôle lié à un service uniquement après la suppression préalable de ses ressources connexes. Vos ressources Amazon EKS sont ainsi protégées, car vous ne pouvez pas involontairement supprimer l'autorisation d'accéder aux ressources.

Pour plus d'informations sur les autres services prenant en charge les rôles liés à un service, consultez les services AWS opérationnels avec IAM et recherchez les services présentant la mention Yes (Oui) dans la colonne Service-linked roles (Rôles liés à un service). Choisissez un Yes (oui) ayant un lien permettant de consulter les détails du rôle pour ce service.

Autorisations du rôle lié à un service pour Amazon EKS

Amazon EKS utilise le rôle lié à un service nommé `AWSServiceRoleForAmazonEKSDashboard`. Ce rôle permet à Amazon EKS de générer et d'afficher le tableau de bord Amazon EKS, y compris les informations agrégées sur les ressources du cluster. La politique `AmazonEKSDashboardServiceRolePolicy` attachée permet au rôle de gérer les ressources suivantes : groupes Auto Scaling, groupes de sécurité, modèles de lancement et profils d'instance IAM. Pour de plus amples informations, consultez the section called "AWS politique gérée : Amazon EKSDashboard ServiceRolePolicy".

Le rôle lié à un service `AWSServiceRoleForAmazonEKSDashboard` approuve les services suivants pour endosser le rôle :

- `dashboard.eks.amazonaws.com`

Pour consulter la dernière version du document de politique JSON, consultez AmazonEKSDashboardServiceRolePolicy dans le guide de référence des politiques gérées par AWS.

Vous devez configurer les autorisations de manière à permettre à une entité IAM (comme un utilisateur, un groupe ou un rôle) de créer, modifier ou supprimer un rôle lié à un service. Pour plus d'informations, consultez Autorisations de rôles liés à un service dans le Guide de l'utilisateur IAM.

Création d'un rôle lié à un service pour Amazon EKS

Vous n'avez pas besoin de créer manuellement un rôle lié à un service. Lorsque vous activez le tableau de bord dans la console AWS, Amazon EKS crée le rôle lié à un service pour vous.

Pour plus d'informations sur l'activation du tableau de bord EKS, consultez the section called "Configuration des organisations".

> **Important**
>
> Ce rôle lié à un service peut apparaître dans votre compte si vous avez effectué une action dans un autre service qui utilise les fonctions prises en charge par ce rôle.

Modification d'un rôle lié à un service pour Amazon EKS

Amazon EKS ne vous autorise pas à modifier le rôle lié à un service `AWSServiceRoleForAmazonEKSDashboard`. Une fois que vous avez créé un rôle lié à un service, vous ne pouvez pas changer le nom du rôle, car plusieurs entités peuvent faire référence à ce rôle. Néanmoins, vous pouvez modifier la description du rôle à l'aide d'IAM. Pour plus d'informations, consultez Modification d'un rôle lié à un service dans le IAM Guide de l'utilisateur.

Suppression d'un rôle lié à un service pour Amazon EKS

Si vous n'avez plus besoin d'utiliser une fonctionnalité ou un service qui nécessite un rôle lié à un service, nous vous recommandons de supprimer ce rôle. De cette façon, vous n'avez aucune entité inutilisée qui n'est pas surveillée ou gérée activement. Cependant, vous devez nettoyer votre rôle lié à un service avant de pouvoir le supprimer manuellement.

Nettoyer un rôle lié à un service

Avant de pouvoir utiliser IAM pour supprimer un rôle lié à un service, vous devez supprimer toutes les ressources utilisées par le rôle.

> **Note**
>
> Si le service Amazon EKS utilise le rôle lorsque vous essayez de supprimer les ressources, la suppression peut échouer. Si cela se produit, patientez quelques minutes et réessayez.

Pour plus d'informations sur la désactivation du tableau de bord EKS, consultez the section called "Configuration des organisations".

Suppression manuelle du rôle lié au service

Utilisez la console IAM, l'interface de ligne de commande AWS ou l'API AWS pour supprimer le rôle lié à un service `AWSServiceRoleForAmazonEKSDashboard`. Pour plus d'informations, consultez Suppression d'un rôle lié à un service dans le Guide de l'utilisateur IAM.

Régions prises en charge pour les rôles liés à un service Amazon EKS

Amazon EKS prend en charge l'utilisation des rôles liés à un service dans toutes les régions où le service est disponible. Pour plus d'informations, consultez Amazon EKS endpoints and quotas (langue française non garantie).

Rôle IAM d'exécution de pod Amazon EKS

Le rôle d'exécution Amazon EKS Pod est requis pour exécuter des pods sur l'infrastructure AWS Fargate.

Lorsque votre cluster crée des pods sur l'infrastructure AWS Fargate, les composants exécutés sur l'infrastructure Fargate doivent effectuer des appels en votre nom. AWS APIs Cela leur permet d'effectuer des actions telles que l'extraction d'images de conteneurs depuis Amazon ECR ou le routage des journaux vers d'autres AWS services. Pour ce faire, le rôle d'exécution de pod Amazon EKS fournit les autorisations IAM.

Lorsque vous créez un profil Fargate, vous devez spécifier un rôle d'exécution de pod pour les composants Amazon EKS qui s'exécutent sur l'infrastructure Fargate à l'aide du profil. Ce rôle est ajouté au Contrôle d'accès basé sur les rôles (RBAC) Kubernetes du cluster à des fins d'autorisation. Cela permet au `kubelet` qui s'exécute sur l'infrastructure Fargate de s'enregistrer auprès de votre cluster Amazon EKS afin qu'il puisse apparaître dans votre cluster en tant que nœud.

> **Note**
>
> Le profil Fargate doit avoir un rôle IAM différent de celui EC2 des groupes de nœuds Amazon.

> **Important**
>
> Les conteneurs s'exécutant dans le pod Fargate ne peuvent pas assumer les autorisations IAM associées à un rôle d'exécution de pod. Pour autoriser les conteneurs de votre Fargate Pod à accéder à d' AWS autres services, vous devez utiliser des rôles IAM pour les comptes de service.

Avant de créer un profil Fargate, vous devez créer un rôle IAM auprès d'Amazon. EKSFargate PodExecutionRolePolicy

Vérifiez si un rôle d'exécution de pod existant est correctement configuré

Vous pouvez suivre la procédure suivante pour vérifier si votre compte dispose déjà d'un rôle d'exécution de pod Amazon EKS correctement configuré. Afin d'éviter tout problème de sécurité lié à un délégué confus, il est important que le rôle restreigne l'accès en fonction de `SourceArn`. Vous pouvez modifier le rôle d'exécution si nécessaire pour inclure la prise en charge des profils Fargate sur d'autres clusters.

1. Ouvrez la console IAM à https://console.aws.amazon.com/iam/ l'adresse.
2. Dans le volet de navigation de gauche, choisissez Rôles.
3. Sur la page Rôles, recherchez la liste des rôles pour Amazon EKSFargate PodExecutionRole. Si le rôle n'existe pas, consultez the section called "Création du rôle d'exécution de pod Amazon EKS" pour créer le rôle. Si le rôle existe, sélectionnez-le.
4. Sur la EKSFargate PodExecutionRole page Amazon, procédez comme suit :
   a. Choisissez Autorisations.
   b. Assurez-vous que la politique gérée par EKSFargatePodExecutionRolePolicyAmazon est attachée au rôle.
   c. Choisissez Trust Relationships (Relations d'approbation).
   d. Choisissez Edit trust policy (Modifier la politique).

5. Dans la page Edit trust policy (Modifier la politique), vérifiez que la relation d'approbation contient la politique suivante, ainsi qu'une ligne pour les profils Fargate sur votre cluster. Dans ce cas, sélectionnez Cancel (Annuler).

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Condition": {
         "ArnLike": {
            "aws:SourceArn": "arn:aws:eks:us-east-1:111122223333:fargateprofile/my-
cluster/*"
         }
      },
      "Principal": {
        "Service": "eks-fargate-pods.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Si la politique correspond mais ne comporte pas de ligne spécifiant les profils Fargate sur votre cluster, vous pouvez ajouter la ligne suivante en haut de l'objet `ArnLike`. Remplacez *`region-code`* par la AWS région dans laquelle se trouve votre cluster, *`111122223333`* par votre identifiant de compte et *`my-cluster`* par le nom de votre cluster.

```
"aws:SourceArn": "arn:aws: eks:region-code:111122223333:fargateprofile/my-cluster/*",
```

Si la politique ne correspond pas, copiez l'intégralité de la politique précédente dans le formulaire et sélectionnez Mettre à jour la politique. Remplacez *`region-code`* par la AWS région dans laquelle se trouve votre cluster. Si vous souhaitez utiliser le même rôle dans toutes les AWS régions de votre compte, remplacez *`region-code`* par*. Remplacez *`111122223333`* par l'ID de votre compte et *`my-cluster`* par le nom de votre cluster. Si vous souhaitez utiliser le même rôle pour tous les clusters de votre compte, remplacez *`my-cluster`* par *.

Création du rôle d'exécution de pod Amazon EKS

Si vous ne possédez pas encore le rôle d'exécution Amazon EKS Pod pour votre cluster, vous pouvez utiliser la AWS Management Console ou la AWS CLI pour le créer.

AWS Management Console

a. Ouvrez la console IAM à https://console.aws.amazon.com/iam/ l'adresse.
b. Dans le volet de navigation de gauche, choisissez Rôles.
c. Sur la page Rôles, choisissez Créer un rôle.
d. Sur la page Select trusted entity (Sélectionner une entité de confiance), procédez comme suit :
   i. Dans la section Type d'entité fiable, sélectionnez AWS service.
   ii. Dans la liste déroulante Cas d'utilisation d'autres AWS services, sélectionnez EKS.
   iii. Sélectionnez EKS – Fargate Pod.
   iv. Choisissez Suivant.
e. Sur la page Add permissions (Ajouter des autorisations), sélectionnez Next (Suivant).
f. Sur la page Name, review, and create (Nommer, vérifier et créer), procédez comme suit :
   i. Pour Role name (Nom de rôle), saisissez un nom unique pour votre rôle, par exemple, `AmazonEKSFargatePodExecutionRole`.
   ii. Sous Ajouter des balises (Facultatif), ajoutez des métadonnées au rôle en attachant les identifications sous forme de paires clé-valeur. Pour plus d'informations sur l'utilisation des balises dans IAM, consultez la rubrique Balisage des ressources IAM dans le Guide de l'utilisateur IAM.
   iii. Choisissez Créer un rôle.
g. Sur la page Rôles, recherchez la liste des rôles pour Amazon EKSFargate PodExecutionRole. Choisissez le rôle.
h. Sur la EKSFargate PodExecutionRole page Amazon, procédez comme suit :
   i. Choisissez Trust Relationships (Relations d'approbation).
   ii. Choisissez Edit trust policy (Modifier la politique).
i. Dans la page Edit trust policy (Modifier la politique), procédez comme suit :
   i. Copiez le contenu suivant et collez-le dans le formulaire Edit trust policy (Modifier la politique). Remplacez *`region-code`* par la AWS région dans laquelle se trouve votre cluster. Si vous souhaitez utiliser le même rôle dans toutes les AWS régions de votre compte, remplacez *`region-code`* par*. Remplacez *`111122223333`* par l'ID de votre

compte et *my-cluster* par le nom de votre cluster. Si vous souhaitez utiliser le même rôle pour tous les clusters de votre compte, remplacez *my-cluster* par *.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Condition": {
         "ArnLike": {
            "aws:SourceArn": "arn:aws:eks:us-
east-1:111122223333:fargateprofile/my-cluster/*"
         }
      },
      "Principal": {
        "Service": "eks-fargate-pods.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

ii. Choisissez Mettre à jour une politique.

AWS CLI

a. Copiez le contenu suivant et collez-le dans un fichier nommé `pod-execution-role-trust-policy.json`. Remplacez *region-code* par la AWS région dans laquelle se trouve votre cluster. Si vous souhaitez utiliser le même rôle dans toutes les AWS régions de votre compte, remplacez *region-code* par*. Remplacez *111122223333* par l'ID de votre compte et *my-cluster* par le nom de votre cluster. Si vous souhaitez utiliser le même rôle pour tous les clusters de votre compte, remplacez *my-cluster* par *.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Condition": {
         "ArnLike": {
            "aws:SourceArn": "arn:aws:eks:us-east-1:111122223333:fargateprofile/
my-cluster/*"
         }
```

```
      },
      "Principal": {
        "Service": "eks-fargate-pods.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

b. Créez un rôle IAM d'exécution de pod.

```
aws iam create-role \
  --role-name AmazonEKSFargatePodExecutionRole \
  --assume-role-policy-document file://"pod-execution-role-trust-policy.json"
```

c. Attachez la politique IAM gérée par Amazon EKS au rôle.

```
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::aws:policy/AmazonEKSFargatePodExecutionRolePolicy \
  --role-name AmazonEKSFargatePodExecutionRole
```

Rôle IAM du connecteur Amazon EKS

Vous pouvez connecter des clusters Kubernetes pour les afficher dans votre. AWS Management Console Pour vous connecter à un cluster Kubernetes, créez un rôle IAM.

Rechercher un rôle de connecteur EKS existant

Vous pouvez utiliser la procédure suivante pour vérifier si votre compte possède déjà le rôle de connecteur Amazon EKS.

1. Ouvrez la console IAM à https://console.aws.amazon.com/iam/ l'adresse.
2. Dans le volet de navigation de gauche, choisissez Rôles.
3. Recherchez dans la liste des rôles `AmazonEKSConnectorAgentRole`. Si un rôle qui inclut `AmazonEKSConnectorAgentRole` n'existe pas, consultez the section called "Création du rôle de l'agent Amazon EKS Connector" pour créer le rôle. Si un rôle qui inclut `AmazonEKSConnectorAgentRole` existe, sélectionnez le rôle pour afficher les politiques attachées.
4. Choisissez Permissions (Autorisations).

5. Assurez-vous que la politique EKSConnector AgentPolicy gérée par Amazon est attachée au rôle. Si la politique est attachée, votre rôle de connecteur Amazon EKS est configuré correctement.
6. Sélectionnez l'onglet Trust relationships (Relations d'approbation), puis Edit trust policy (Modifier la relation d'approbation).
7. Vérifiez que la relation d'approbation contient la politique suivante. Si la relation d'approbation correspond à la politique ci-dessous, sélectionnez Annuler. Si la relation d'approbation ne correspond pas, copiez la politique dans la fenêtre Modifier la politique d'approbation et sélectionnez Mettre à jour la politique.

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

Création du rôle de l'agent Amazon EKS Connector

Vous pouvez utiliser le AWS Management Console ou AWS CloudFormation pour créer le rôle d'agent du connecteur.

AWS CLI

a. Créez le fichier nommé `eks-connector-agent-trust-policy.json` qui contient le JSON suivant à utiliser pour le rôle IAM.

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

```
                "Service": [
                    "ssm.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

b. Créez le fichier nommé `eks-connector-agent-policy.json` qui contient le JSON suivant à utiliser pour le rôle IAM.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "SsmControlChannel",
            "Effect": "Allow",
            "Action": [
                "ssmmessages:CreateControlChannel"
            ],
            "Resource": "arn:aws:eks:*:*:cluster/*"
        },
        {
            "Sid": "ssmDataplaneOperations",
            "Effect": "Allow",
            "Action": [
                "ssmmessages:CreateDataChannel",
                "ssmmessages:OpenDataChannel",
                "ssmmessages:OpenControlChannel"
            ],
            "Resource": "*"
        }
    ]
}
```

c. Créez le rôle d'agent Amazon EKS Connector à l'aide de la politique d'approbation et de la politique que vous avez créées dans les listes précédentes.

```
aws iam create-role \
     --role-name AmazonEKSConnectorAgentRole \
     --assume-role-policy-document file://eks-connector-agent-trust-policy.json
```

d. Attachez la politique à votre rôle d'agent Amazon EKS Connector.

```
aws iam put-role-policy \
     --role-name AmazonEKSConnectorAgentRole \
     --policy-name AmazonEKSConnectorAgentPolicy \
     --policy-document file://eks-connector-agent-policy.json
```

AWS CloudFormation

a. Enregistrez le AWS CloudFormation modèle suivant dans un fichier texte sur votre système local.

> **Note**
>
> Ce modèle crée également le rôle lié à un service qui serait autrement créé lorsque l'API `registerCluster` est appelée. Consultez the section called “Rôle du connecteur de cluster” pour plus de détails.

```
---
AWSTemplateFormatVersion: '2010-09-09'
Description: 'Provisions necessary resources needed to register clusters in EKS'
Parameters: {}
Resources:
  EKSConnectorSLR:
    Type: AWS::IAM::ServiceLinkedRole
    Properties:
      AWSServiceName: eks-connector.amazonaws.com

  EKSConnectorAgentRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: Allow
            Action: [ 'sts:AssumeRole' ]
            Principal:
              Service: 'ssm.amazonaws.com'

  EKSConnectorAgentPolicy:
    Type: AWS::IAM::Policy
```

```
    Properties:
      PolicyName: EKSConnectorAgentPolicy
      Roles:
        - {Ref: 'EKSConnectorAgentRole'}
      PolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: 'Allow'
            Action: [ 'ssmmessages:CreateControlChannel' ]
            Resource:
            - Fn::Sub: 'arn:${AWS::Partition}:eks:*:*:cluster/*'
          - Effect: 'Allow'
            Action: [ 'ssmmessages:CreateDataChannel',
 'ssmmessages:OpenDataChannel', 'ssmmessages:OpenControlChannel' ]
            Resource: "*"
Outputs:
  EKSConnectorAgentRoleArn:
    Description: The agent role that EKS connector uses to communicate with AWS
 services.
    Value: !GetAtt EKSConnectorAgentRole.Arn
```

b. Ouvrez la AWS CloudFormation console.

c. Sélectionnez Créer une pile avec de nouvelles ressources (standard).

d. Pour Specify template (Spécifier un modèle), sélectionnez Upload a template file (Télécharger un fichier de modèle), puis choisissez Choose file (Choisir un fichier).

e. Choisissez le fichier que vous avez créé précédemment, puis choisissez Next (Suivant).

f. Pour Stack name (Nom de la pile), saisissez un nom pour votre rôle, par exemple `eksConnectorAgentRole`, puis choisissez Next (Suivant).

g. Sur la page Configurer les options de pile, choisissez Next (Suivant).

h. Sur la page Vérification, vérifiez vos informations, reconnaissez que la pile peut créer des ressources IAM, puis choisissez Create stack (Créer la pile).

AWS politiques gérées pour Amazon Elastic Kubernetes Service

Une politique AWS gérée est une politique autonome créée et administrée par AWS. AWS les politiques gérées sont conçues pour fournir des autorisations pour de nombreux cas d'utilisation courants afin que vous puissiez commencer à attribuer des autorisations aux utilisateurs, aux groupes et aux rôles.

N'oubliez pas que les politiques AWS gérées peuvent ne pas accorder d'autorisations de moindre privilège pour vos cas d'utilisation spécifiques, car elles sont accessibles à tous les AWS clients. Nous vous recommandons de réduire encore les autorisations en définissant des politiques gérées par le client qui sont propres à vos cas d'utilisation.

Vous ne pouvez pas modifier les autorisations définies dans les politiques AWS gérées. Si les autorisations définies dans une politique AWS gérée sont AWS mises à jour, la mise à jour affecte toutes les identités principales (utilisateurs, groupes et rôles) auxquelles la politique est attachée. AWS est le plus susceptible de mettre à jour une politique AWS gérée lorsqu'un nouveau AWS service est lancé ou lorsque de nouvelles opérations d'API sont disponibles pour les services existants.

Pour plus d'informations, consultez Politiques gérées par AWS dans le Guide de l'utilisateur IAM.

AWS politique gérée : Amazoneks_CNI_Policy

Vous pouvez attacher `AmazonEKS_CNI_Policy` à vos entités IAM. Avant de créer un groupe de nœuds Amazon EC2, cette politique doit être associée soit au rôle IAM du nœud, soit à un rôle IAM utilisé spécifiquement par le plug-in CNI Amazon VPC pour Kubernetes. Ceci lui permet d'effectuer des actions en votre nom. Nous vous recommandons d'associer la politique à un rôle utilisé uniquement par le plug-in. Pour plus d'informations, consultez the section called "CNI Amazon VPC" et the section called "Configuration pour IRSA".

Détails de l'autorisation

Cette politique inclut les autorisations suivantes qui permettent à Amazon EKS d'effectuer les tâches suivantes :

- **`ec2:*NetworkInterface`** et **`ec2:*PrivateIpAddresses`** : permet au plug-in CNI Amazon VPC d'effectuer des actions telles que le provisionnement d'interfaces réseau Elastic et d'adresses IP pour les pods afin de fournir une mise en réseau pour les applications qui s'exécutent dans Amazon EKS.
- Actions de lecture **`ec2`** : permet au plug-in CNI Amazon VPC d'effectuer des actions telles que la description d'instances et de sous-réseaux afin de connaître le nombre d'adresses IP libres dans vos sous-réseaux Amazon VPC. Le CNI VPC peut utiliser les adresses IP libres dans chaque sous-réseau pour sélectionner les sous-réseaux disposant du plus grand nombre d'adresses IP libres à utiliser lors de la création d'une interface réseau Elastic.

Pour consulter la dernière version du document de politique JSON, consultez Amazoneks_CNI_Policy dans le Managed Policy Reference Guide. AWS

AWS politique gérée : Amazon EKSCluster Policy

Vous pouvez attacher `AmazonEKSClusterPolicy` à vos entités IAM. Avant de créer un cluster, vous devez disposer d'un rôle IAM de cluster avec la politique ci-jointe. Les clusters Kubernetes gérés par Amazon EKS effectuent des appels vers d'autres AWS services en votre nom. Ils le font pour gérer les ressources que vous utilisez avec le service.

Cette politique inclut les autorisations suivantes qui permettent à Amazon EKS d'effectuer les tâches suivantes :

- **`autoscaling`** : lit et met à jour la configuration d'un groupe Auto Scaling. Ces autorisations ne sont pas utilisées par Amazon EKS, mais restent dans la politique pour des raisons de compatibilité ascendante.
- **`ec2`** : utilisez les volumes et les ressources réseau associés aux nœuds Amazon EC2. Ceci est nécessaire pour que le plan de contrôle Kubernetes puisse joindre des instances à un cluster et approvisionner et gérer dynamiquement les volumes Amazon EBS demandés par des volumes persistants Kubernetes.
- **`ec2`** : supprimer les interfaces réseau Elastic créées par le CNI VPC. Cette opération est nécessaire pour permettre à EKS de nettoyer les interfaces réseau Elastic qui restent si le VPC CNI se ferme de manière inattendue.
- **`elasticloadbalancing`** : utilisez les équilibreurs de charge Elastic et y ajouter des nœuds en tant que cibles. Ceci est nécessaire pour que le plan de contrôle Kubernetes puisse allouer de manière dynamique les Elastic Load Balancer demandés par les services Kubernetes.
- **`iam`** : créez un rôle lié à un service. Ceci est nécessaire pour que le plan de contrôle Kubernetes puisse allouer de manière dynamique les Elastic Load Balancer demandés par les services Kubernetes.
- **`kms`**— Lit une clé depuis AWS KMS. Cette condition est nécessaire pour que le plan de contrôle de Kubernetes prenne en charge le chiffrement des secrets Kubernetes stockés dans `etcd`.

Pour consulter la dernière version du document de politique JSON, consultez Amazon EKSCluster Policy dans le AWS Managed Policy Reference Guide.

AWS politique gérée : Amazon EKSDashboard ConsoleReadOnly

Vous pouvez attacher `AmazonEKSDashboardConsoleReadOnly` à vos entités IAM.

Cette politique inclut les autorisations suivantes qui permettent à Amazon EKS d'effectuer les tâches suivantes :

- **`eks`** : accès en lecture seule aux données du tableau de bord EKS, aux ressources et aux informations sur les versions du cluster. Cela permet de consulter les métriques liées à EKS et les détails de configuration du cluster.
- **`organizations`**- Accès en lecture seule aux informations des AWS Organizations, notamment :
  - Affichage des détails de l'organisation et de l'accès aux services
  - Affichage de la liste des racines organisationnelles, des comptes et des unités organisationnelles
  - Affichage de la structure organisationnelle

Pour consulter la dernière version du document de politique JSON, consultez Amazon EKSDashboard ConsoleReadOnly dans le AWS Managed Policy Reference Guide.

AWS politique gérée : Amazon EKSFargate PodExecutionRolePolicy

Vous pouvez attacher `AmazonEKSFargatePodExecutionRolePolicy` à vos entités IAM. Avant de pouvoir créer un profil Fargate, vous devez créer un rôle d'exécution de pod Fargate et y associer cette politique. Pour plus d'informations, consultez the section called "Étape 2 : créer un rôle d'exécution Fargate Pod" et the section called "Définir des profils".

Cette politique accorde au rôle les autorisations lui permettant d'accéder aux autres ressources de AWS service requises pour exécuter les pods Amazon EKS sur Fargate.

Détails de l'autorisation

Cette politique inclut les autorisations suivantes qui permettent à Amazon EKS d'effectuer les tâches suivantes :

- **`ecr`** : permet aux pods qui s'exécutent sur Fargate d'extraire les images de conteneur stockées dans Amazon ECR.

Pour consulter la dernière version du document de politique JSON, consultez Amazon EKSFargate PodExecutionRolePolicy dans le AWS Managed Policy Reference Guide.

AWS politique gérée : Amazon EKSConnector ServiceRolePolicy

Vous ne pouvez pas associer `AmazonEKSConnectorServiceRolePolicy` à vos entités IAM. Cette politique est attachée à un rôle lié à un service qui permet à Amazon EKS d'effectuer des actions en votre nom. Pour de plus amples informations, veuillez consulter the section called "Rôle du connecteur de cluster".

Ce rôle permet à Amazon EKS de connecter des clusters Kubernetes. Les politiques associées permettent au rôle de gérer les ressources nécessaires pour se connecter à votre cluster Kubernetes enregistré.

Détails de l'autorisation

Cette politique inclut les autorisations suivantes qui permettent à Amazon EKS d'effectuer les tâches suivantes.

- **`SSM Management`**— Créez, décrivez et supprimez les activations SSM, et annulez l'enregistrement des instances gérées. Cela permet les opérations de base de Systems Manager.
- **`Session Management`**— Démarrez des sessions SSM spécifiquement pour les clusters EKS et exécutez des commandes non interactives à l'aide du document AmazoneKS.
- **`IAM Role Passing`**— Transmettez les rôles IAM spécifiquement au service SSM, sous le contrôle d'une condition qui restreint les rôles transmis à. `ssm.amazonaws.com`
- **`EventBridge Rules`**— Créez des EventBridge règles et des cibles, mais uniquement lorsque vous êtes géré par`eks-connector.amazonaws.com`. Les règles sont spécifiquement limitées au AWS SSM en tant que source d'événement.

Pour consulter la dernière version du document de politique JSON, consultez Amazon EKSConnector ServiceRolePolicy dans le AWS Managed Policy Reference Guide.

AWS politique gérée : Amazon EKSFor FargateServiceRolePolicy

Vous ne pouvez pas associer `AmazonEKSForFargateServiceRolePolicy` à vos entités IAM. Cette politique est attachée à un rôle lié à un service qui permet à Amazon EKS d'effectuer des actions en votre nom. Pour de plus amples informations, veuillez consulter `AWSServiceRoleforAmazonEKSForFargate`.

Cette politique accorde les autorisations nécessaires à Amazon EKS pour exécuter des tâches Fargate. La politique n'est utilisée que si vous avez des nœuds Fargate.

Détails de l'autorisation

Cette politique inclut les autorisations suivantes qui permettent à Amazon EKS d'effectuer les tâches suivantes.

- **`ec2`** : créer et supprimer des interfaces réseau Elastic et décrire les interfaces réseau Elastic et les ressources. Ceci est nécessaire pour que le service Amazon EKS Fargate puisse configurer le réseau VPC requis pour les pods Fargate.

Pour consulter la dernière version du document de politique JSON, consultez Amazon EKSFor FargateServiceRolePolicy dans le AWS Managed Policy Reference Guide.

AWS politique gérée : Amazon EKSCompute Policy

Vous pouvez attacher `AmazonEKSComputePolicy` à vos entités IAM. Vous pouvez associer cette politique à votre rôle IAM du cluster afin d'étendre les ressources qu'EKS peut gérer dans votre compte.

Cette politique accorde les autorisations requises pour qu'Amazon EKS puisse créer et gérer des instances EC2 pour le cluster EKS, ainsi que les autorisations IAM nécessaires pour configurer EC2. De plus, cette politique accorde à Amazon EKS les autorisations nécessaires pour créer le rôle lié à un service EC2 Spot en votre nom.

Détails de l'autorisation

Cette politique inclut les autorisations suivantes qui permettent à Amazon EKS d'effectuer les tâches suivantes :

- Autorisations **`ec2`** :
  - `ec2:CreateFleet` et `ec2:RunInstances` : permet de créer des instances EC2 et d'utiliser des ressources EC2 spécifiques (images, groupes de sécurité, sous-réseaux) pour les nœuds du cluster EKS.
  - `ec2:CreateLaunchTemplate` : permet de créer des modèles de lancement EC2 pour les nœuds du cluster EKS.
  - La politique comprend également des conditions visant à restreindre l'utilisation de ces autorisations EC2 aux ressources balisées avec le nom du cluster EKS et d'autres balises pertinentes.
  - `ec2:CreateTags` : permet de baliser les ressources EC2 créées par les actions `CreateFleet`, `RunInstances` et `CreateLaunchTemplate`.

- Autorisations **`iam`** :
  - `iam:AddRoleToInstanceProfile` : permet d'ajouter un rôle IAM au profil d'instance de calcul EKS.
  - `iam:PassRole` : permet de transmettre les rôles IAM nécessaires au service EC2.

Pour consulter la dernière version du document de politique JSON, consultez Amazon EKSCompute Policy dans le AWS Managed Policy Reference Guide.

AWS politique gérée : Amazon EKSNetworking Policy

Vous pouvez attacher `AmazonEKSNetworkingPolicy` à vos entités IAM. Vous pouvez associer cette politique à votre rôle IAM du cluster afin d'étendre les ressources qu'EKS peut gérer dans votre compte.

Cette politique est conçue pour accorder les autorisations nécessaires à Amazon EKS afin de créer et de gérer les interfaces réseau pour le cluster EKS, permettant ainsi au plan de contrôle et aux composants master de communiquer et de fonctionner correctement.

Détails de l'autorisation

Cette politique accorde les autorisations suivantes pour permettre à Amazon EKS de gérer les interfaces réseau pour le cluster :

- Autorisations d'interface réseau **`ec2`** :
  - `ec2:CreateNetworkInterface` : permet de créer des interfaces réseau EC2.
  - La politique inclut des conditions visant à restreindre l'utilisation de cette autorisation aux interfaces réseau balisées avec le nom du cluster EKS et le nom du nœud CNI Kubernetes.
  - `ec2:CreateTags` : permet d'ajouter des balises aux interfaces réseau créées par l'action `CreateNetworkInterface`.
- Autorisations de gestion de l'interface réseau **`ec2`** :
  - `ec2:AttachNetworkInterface`, `ec2:DetachNetworkInterface` : permet de connecter et de déconnecter des interfaces réseau à des instances EC2.
  - `ec2:UnassignPrivateIpAddresses`, `ec2:UnassignIpv6Addresses`, `ec2:AssignPrivateIpAddresses`, `ec2:AssignIpv6Addresses` : permet de gérer les attributions d'adresses IP des interfaces réseau.
  - Ces autorisations sont limitées aux interfaces réseau balisées avec le nom du cluster Amazon EKS.

Pour consulter la dernière version du document de politique JSON, consultez Amazon EKSNetworking Policy dans le AWS Managed Policy Reference Guide.

AWS politique gérée : Amazon EKSBlock StoragePolicy

Vous pouvez attacher `AmazonEKSBlockStoragePolicy` à vos entités IAM. Vous pouvez associer cette politique à votre rôle IAM du cluster afin d'étendre les ressources qu'EKS peut gérer dans votre compte.

Cette politique accorde les autorisations nécessaires à Amazon EKS pour créer, gérer et maintenir des volumes EC2 et des instantanés pour le cluster Amazon EKS, permettant ainsi au plan de contrôle et aux composants master de provisionner et d'utiliser le stockage persistant selon les besoins des charges de travail Kubernetes.

Détails de l'autorisation

Cette politique IAM accorde les autorisations suivantes pour permettre à Amazon EKS de gérer les volumes et les instantanés EC2 :

- Autorisations de gestion des volumes **`ec2`** :
  - `ec2:AttachVolume`, `ec2:DetachVolume`, `ec2:ModifyVolume`, `ec2:EnableFastSnapshotRestores` : permet de connecter, déconnecter, modifier et activer la restauration rapide des instantanés pour les volumes EC2.
  - Ces autorisations sont limitées aux volumes balisés avec le nom du cluster EKS.
  - `ec2:CreateTags` : permet d'ajouter des balises aux volumes EC2 et aux instantanés créés par les actions `CreateVolume` et `CreateSnapshot`.
- Autorisations de création de volume **`ec2`** :
  - `ec2:CreateVolume` : permet de créer de nouveaux volumes EC2.
  - La politique comprend des conditions qui limitent l'utilisation de cette autorisation aux volumes balisés avec le nom du cluster EKS et d'autres balises pertinentes.
  - `ec2:CreateSnapshot` : permet de créer de nouveaux instantanés de volume EC2.
  - La politique comprend des conditions qui limitent l'utilisation de cette autorisation aux instantanés balisés avec le nom du cluster EKS et d'autres balises pertinentes.

Pour consulter la dernière version du document de politique JSON, consultez Amazon EKSBlock StoragePolicy dans le AWS Managed Policy Reference Guide.

AWS politique gérée : Amazon EKSLoad BalancingPolicy

Vous pouvez attacher `AmazonEKSLoadBalancingPolicy` à vos entités IAM. Vous pouvez associer cette politique à votre rôle IAM du cluster afin d'étendre les ressources qu'EKS peut gérer dans votre compte.

Cette politique IAM accorde les autorisations nécessaires à Amazon EKS pour travailler avec différents AWS services afin de gérer les Elastic Load Balancers (ELBs) et les ressources associées.

Détails de l'autorisation

Les principales autorisations accordées par cette politique sont les suivantes :

- **`elasticloadbalancing`** : permet de créer, modifier et gérer des Elastic Load Balancers et des groupes cibles. Cela inclut les autorisations de créer, mettre à jour et supprimer des équilibreurs de charge, des groupes cibles, des écouteurs et des règles.
- **`ec2`** : permet de créer et de gérer des groupes de sécurité, qui sont nécessaires au plan de contrôle Kubernetes pour joindre des instances à un cluster et gérer les volumes Amazon EBS. Permet également de décrire et de répertorier les ressources EC2 telles que les instances VPCs, les sous-réseaux, les groupes de sécurité et d'autres ressources réseau.
- **`iam`**: Permet de créer un rôle lié à un service pour Elastic Load Balancing, qui est nécessaire au provisionnement dynamique du plan de contrôle Kubernetes. ELBs
- **`kms`**: Permet de lire une clé depuis AWS KMS, qui est requise pour que le plan de contrôle Kubernetes prenne en charge le chiffrement des secrets Kubernetes stockés dans etcd.
- **`wafv2`**et **`shield`**: Permet d'associer et de dissocier le Web ACLs et de créer/supprimer des protections AWS Shield pour les Elastic Load Balancers.
- **`cognito-idp`** , **`acm`** et **`elasticloadbalancing`** : accorde les autorisations nécessaires pour décrire les clients du groupe d'utilisateurs, répertorier et décrire les certificats, et décrire les groupes cibles, qui sont nécessaires au plan de contrôle Kubernetes pour gérer les Elastic Load Balancers.

La politique comprend également plusieurs vérifications de conditions afin de garantir que les autorisations ont une portée limitée au cluster EKS spécifique géré, à l'aide de la balise `eks:eks-cluster-name`.

Pour consulter la dernière version du document de politique JSON, consultez Amazon EKSLoad BalancingPolicy dans le AWS Managed Policy Reference Guide.

AWS politique gérée : Amazon EKSMCPRead OnlyAccess

Vous pouvez attacher `AmazonEKSMCPReadOnlyAccess` à vos entités IAM. Cette politique fournit un accès en lecture seule aux ressources Amazon EKS et aux AWS services associés, permettant au serveur Amazon EKS Model Context Protocol (MCP) d'effectuer des opérations d'observabilité et de dépannage sans apporter de modifications à votre infrastructure.

Détails de l'autorisation

Cette politique inclut les autorisations suivantes qui permettent aux directeurs d'effectuer les tâches suivantes :

- **`eks`**Permet aux principaux de décrire et de répertorier les clusters EKS, les groupes de nœuds, les modules complémentaires, les entrées d'accès, les informations et d'accéder à l'API Kubernetes pour les opérations en lecture seule.
- **`iam`**Permet aux principaux de récupérer des informations sur les rôles IAM, les politiques et leurs pièces jointes afin de comprendre les autorisations associées aux ressources EKS.
- **`ec2`**Permet aux principaux de décrire VPCs les sous-réseaux et les tables de routage afin de comprendre la configuration réseau des clusters EKS.
- **`sts`**Permet aux principaux de récupérer les informations d'identité de l'appelant à des fins d'authentification et d'autorisation.
- **`logs`**Permet aux principaux de lancer des requêtes et de récupérer les résultats des requêtes à partir des CloudWatch journaux à des fins de dépannage et de surveillance.
- **`cloudwatch`**Permet aux principaux de récupérer des données métriques pour surveiller les performances du cluster et de la charge de travail.
- **`eks-mcp`**Permet aux principaux d'invoquer des opérations MCP et d'appeler des outils en lecture seule au sein du serveur MCP Amazon EKS.

Pour consulter la dernière version du document de politique JSON, consultez Amazon EKSMCPRead OnlyAccess dans le AWS Managed Policy Reference Guide.

AWS politique gérée : Amazon EKSService Policy

Vous pouvez attacher `AmazonEKSServicePolicy` à vos entités IAM. Les clusters créés avant le 16 avril 2020 nécessitaient la création d'un rôle IAM et l'ajout de cette politique. Les clusters créés à partir du 16 avril 2020 ne nécessitent pas la création d'un rôle ni l'attribution de cette politique. Lorsque vous créez un cluster à l'aide d'un principal IAM `iam:CreateServiceLinkedRole`

autorisé, le rôle lié au service AWSServiceRoleforAmazonEKS est automatiquement créé pour vous. Le rôle lié à un service est EKSService RolePolicy associé à la politique gérée suivante : Amazon.

Cette politique permet à Amazon EKS de créer et de gérer les ressources nécessaires à l'exploitation des clusters Amazon EKS.

Détails de l'autorisation

Cette politique inclut les autorisations suivantes qui permettent à Amazon EKS d'effectuer les tâches suivantes.

- **`eks`** : mettez à jour la version Kubernetes de votre cluster après avoir lancé une mise à jour. Cette autorisation n'est pas utilisée par Amazon EKS, mais reste dans la politique pour des raisons de compatibilité ascendante.
- **`ec2`** : utilisez les interfaces réseau Elastic et d'autres ressources et balises réseau. Cela est requis par Amazon EKS pour configurer la mise en réseau qui facilite la communication entre les nœuds et le plan de contrôle Kubernetes. Lisez les informations sur les groupes de sécurité. Mettez à jour les balises sur les groupes de sécurité.
- **`route53`** : associez un VPC à une zone hébergée. Cela est requis par Amazon EKS pour activer la mise en réseau de points de terminaison privés pour votre serveur d'API de cluster Kubernetes.
- **`logs`**— Enregistrez les événements. Cela est nécessaire pour qu'Amazon EKS puisse envoyer les journaux du plan de contrôle Kubernetes à. CloudWatch
- **`iam`** : créez un rôle lié à un service. Ceci est nécessaire pour qu'Amazon EKS puisse créer le rôle lié à un service the section called "Autorisations du rôle lié à un service pour Amazon EKS" en votre nom.

Pour consulter la dernière version du document de politique JSON, consultez Amazon EKSService Policy dans le AWS Managed Policy Reference Guide.

AWS politique gérée : Amazon EKSService RolePolicy

Vous ne pouvez pas associer `AmazonEKSServiceRolePolicy` à vos entités IAM. Cette politique est attachée à un rôle lié à un service qui permet à Amazon EKS d'effectuer des actions en votre nom. Pour de plus amples informations, veuillez consulter the section called "Autorisations du rôle lié à un service pour Amazon EKS". Lorsque vous créez un cluster à l'aide d'un principal IAM `iam:CreateServiceLinkedRole` autorisé, le rôle lié au service AWSServiceRoleforAmazonEKS est automatiquement créé pour vous et cette politique y est associée.

Cette politique permet au rôle lié au service d'appeler les AWS services en votre nom.

Détails de l'autorisation

Cette politique inclut les autorisations suivantes qui permettent à Amazon EKS d'effectuer les tâches suivantes.

- **`ec2`** : créez et décrivez les interfaces réseau Elastic et les instances Amazon EC2, le groupe de sécurité du cluster et le VPC nécessaires à la création d'un cluster. Pour de plus amples informations, veuillez consulter the section called "Exigences de groupe de sécurité". Lisez les informations sur les groupes de sécurité. Mettez à jour les balises sur les groupes de sécurité. Consultez les informations relatives aux réserves de capacité à la demande. Lisez la configuration du VPC, y compris les tables de routage et le réseau, ACLs pour détecter les problèmes de configuration dans le cadre des informations sur le cluster.
- Mode automatique **`ec2`** : résiliez les instances EC2 créées par le mode automatique EKS. Pour de plus amples informations, veuillez consulter Mode automatique EKS.
- **`iam`** : répertoriez toutes les politiques gérées associées à un rôle IAM. Cette condition est nécessaire pour permettre à Amazon EKS de répertorier et valider toutes les politiques et autorisations gérées requises pour créer un cluster.
- Associer un VPC à une zone hébergée : cela est requis par Amazon EKS pour activer la mise en réseau de points de terminaison privés pour votre serveur d'API de cluster Kubernetes.
- Journaliser les événements : cela est nécessaire pour qu'Amazon EKS puisse envoyer les journaux du plan de contrôle Kubernetes à. CloudWatch
- Put metric : cela est nécessaire pour qu'Amazon EKS puisse envoyer les journaux du plan de contrôle Kubernetes à. CloudWatch
- **`eks`** : gérez les entrées et les politiques d'accès au cluster, ce qui permet un contrôle précis des personnes autorisées à accéder aux ressources EKS et des actions qu'elles peuvent effectuer. Cela inclut l'association de politiques d'accès standard pour les opérations de calcul, de mise en réseau, d'équilibrage de charge et de stockage.
- **`elasticloadbalancing`** : créez, gérez et supprimez les équilibreurs de charge et leurs composants (écouteurs, groupes cibles, certificats) associés aux clusters EKS. Affichez les attributs et l'état des équilibreurs de charge.
- **`events`**- Créez et gérez des EventBridge règles pour surveiller les événements EC2 et AWS Health liés aux clusters EKS, afin de permettre des réponses automatisées aux modifications de l'infrastructure et aux alertes de santé.

- **`iam`** : gérez les profils d'instance EC2 avec le préfixe « eks », y compris la création, la suppression et l'association de rôles, ce qui est nécessaire pour la gestion des nœuds EKS. Permet de décrire n'importe quel profil d'instance afin de permettre aux utilisateurs de définir des profils d'instance personnalisés à utiliser par leurs nœuds de travail.
- **`pricingshield`**- Accédez aux informations AWS tarifaires et à l'état de protection du Shield, permettant de gérer les coûts et de bénéficier de fonctionnalités de sécurité avancées pour les ressources EKS.
- Nettoyage des ressources : supprimez en toute sécurité les ressources balisées EKS, y compris les volumes, les instantanés, les modèles de lancement et les interfaces réseau, lors des opérations de nettoyage des clusters.

Pour consulter la dernière version du document de politique JSON, consultez Amazon EKSService RolePolicy dans le AWS Managed Policy Reference Guide.

AWS politique gérée : Amazon EKSVPCResource Controller

Vous pouvez associer la politique `AmazonEKSVPCResourceController` à vos identités IAM. Si vous utilisez des groupes de sécurité pour les pods, vous devez associer cette politique à votre rôle IAM de cluster Amazon EKS pour effectuer des actions en votre nom.

Cette politique accorde les autorisations de rôle de cluster pour gérer les interfaces réseau Elastic et les adresses IP pour les nœuds.

Détails de l'autorisation

Cette politique inclut les autorisations suivantes qui permettent à Amazon EKS d'effectuer les tâches suivantes :

- **`ec2`** : gérez les interfaces réseau Elastic et les adresses IP pour prendre en charge les groupes de sécurité des pods et les nœuds Windows.

Pour consulter la dernière version du document de politique JSON, consultez Amazon EKSVPCResource Controller dans le AWS Managed Policy Reference Guide.

AWS politique gérée : Amazon EKSWorker NodePolicy

Vous pouvez attacher `AmazonEKSWorkerNodePolicy` à vos entités IAM. Vous devez attacher cette politique à un rôle IAM de nœud spécifié lorsque vous créez des nœuds Amazon EC2 qui permettent

à Amazon EKS d'effectuer des actions en votre nom. Si vous créez un groupe de nœuds à l'aide de `eksctl`, il crée le rôle IAM de nœud et attache automatiquement cette politique au rôle.

Cette politique accorde aux nœuds Amazon EKS Amazon EC2 les autorisations de connexion aux clusters Amazon EKS.

Détails de l'autorisation

Cette politique inclut les autorisations suivantes qui permettent à Amazon EKS d'effectuer les tâches suivantes :

- **`ec2`** : lisez les informations relatives au volume et au réseau de l'instance. Ceci est nécessaire pour que les nœuds Kubernetes puissent décrire des informations sur les ressources Amazon EC2 requises pour que le nœud puisse rejoindre le cluster Amazon EKS.
- **`eks`** : décrivez éventuellement le cluster dans le cadre du démarrage du nœud.
- **`eks-auth:AssumeRoleForPodIdentity`** : autorisez la récupération des informations d'identification pour les charges de travail EKS sur le nœud. Cela est nécessaire pour que l'identité du pod EKS fonctionne correctement.

Pour consulter la dernière version du document de politique JSON, consultez Amazon EKSWorker NodePolicy dans le AWS Managed Policy Reference Guide.

AWS politique gérée : Amazon EKSWorker NodeMinimalPolicy

Vous pouvez associer les WorkerNodeMinimalPolicy Amazoneks à vos entités IAM. Vous pouvez associer cette politique à un rôle IAM de nœud que vous spécifiez lors de la création de nœuds Amazon EC2 qui permettent à Amazon EKS d'effectuer des actions en votre nom.

Cette politique accorde aux nœuds Amazon EKS Amazon EC2 les autorisations de connexion aux clusters Amazon EKS. Cette politique comporte moins d'autorisations que celle d'Amazon EKSWorkerNodePolicy.

Détails de l'autorisation

Cette politique inclut les autorisations suivantes qui permettent à Amazon EKS d'effectuer les tâches suivantes :

- `eks-auth:AssumeRoleForPodIdentity` : permet la récupération des informations d'identification pour les charges de travail EKS sur le nœud. Cela est nécessaire pour que l'identité du pod EKS fonctionne correctement.

Pour consulter la dernière version du document de politique JSON, consultez Amazon EKSWorker NodePolicy dans le AWS Managed Policy Reference Guide.

AWS politique gérée : AWSService RoleForAmazon EKSNodegroup

Vous ne pouvez pas associer `AWSServiceRoleForAmazonEKSNodegroup` à vos entités IAM. Cette politique est attachée à un rôle lié à un service qui permet à Amazon EKS d'effectuer des actions en votre nom. Pour de plus amples informations, veuillez consulter the section called "Autorisations du rôle lié à un service pour Amazon EKS".

Cette politique accorde au rôle les autorisations `AWSServiceRoleForAmazonEKSNodegroup` qui lui permettent de créer et de gérer les groupes de nœuds Amazon EC2 dans votre compte.

Détails de l'autorisation

Cette politique inclut les autorisations suivantes qui permettent à Amazon EKS d'effectuer les tâches suivantes :

- **`ec2`** : permet de travailler avec des groupes de sécurité, des balises, des réserves de capacité et des modèles de lancement. Ceci est nécessaire pour que les groupes de nœuds gérés Amazon EKS puissent activer la configuration de l'accès à distance et décrire les réserves de capacité pouvant être utilisées dans les groupes de nœuds gérés. En outre, les groupes de nœuds gérés par Amazon EKS créent un modèle de lancement en votre nom. Cela permet de configurer le groupe Amazon EC2 Auto Scaling qui prend en charge chaque groupe de nœuds gérés.
- **`iam`** : permet de créer un rôle lié à un service et de transmettre un rôle. Ceci est requis par les groupes de nœuds gérés Amazon EKS pour gérer les profils d'instance pour le rôle transmis lors de la création d'un groupe de nœuds gérés. Ce profil d'instance est utilisé par les instances Amazon EC2 lancées dans le cadre d'un groupe de nœuds gérés. Amazon EKS doit créer des rôles liés au service pour d'autres services tels que les groupes Amazon EC2 Auto Scaling. Ces autorisations sont utilisées dans la création d'un groupe de nœuds gérés.
- **`autoscaling`** : permet de travailler avec des groupes Auto Scaling. Ceci est requis par les groupes de nœuds gérés par Amazon EKS pour gérer le groupe Amazon EC2 Auto Scaling qui sauvegarde chaque groupe de nœuds gérés. Il est également utilisé pour prendre en charge des fonctionnalités telles que l'expulsion des pods lorsque les nœuds sont résiliés ou recyclés lors des mises à jour des groupes de nœuds et la gestion des pools de chaleur configurés sur des groupes de nœuds gérés.

Pour consulter la dernière version du document de politique JSON, consultez AWSServiceRoleForAmazonEKSNodegrouple Guide de référence des politiques AWS gérées.

AWS politique gérée : Amazon EKSDashboard ServiceRolePolicy

Vous ne pouvez pas associer `AmazonEKSDashboardServiceRolePolicy` à vos entités IAM. Cette politique est attachée à un rôle lié à un service qui permet à Amazon EKS d'effectuer des actions en votre nom. Pour de plus amples informations, veuillez consulter the section called "Autorisations du rôle lié à un service pour Amazon EKS".

Cette politique accorde au rôle les autorisations `AWSServiceRoleForAmazonEKSDashboard` qui lui permettent de créer et de gérer les groupes de nœuds Amazon EC2 dans votre compte.

Détails de l'autorisation

Cette politique comprend les autorisations suivantes qui permettent d'accéder à ces tâches :

- **`organizations`**— Consultez les informations relatives à la structure et aux comptes de vos AWS Organisations. Cela inclut les autorisations permettant de répertorier les comptes de votre organisation, d'afficher les unités organisationnelles et les racines, de répertorier les administrateurs délégués, d'afficher les services qui ont accès à votre organisation et de récupérer des informations détaillées sur votre organisation et vos comptes.

Pour consulter la dernière version du document de politique JSON, consultez Amazon EKSDashboard ServiceRolePolicy dans le AWS Managed Policy Reference Guide.

AWS politique gérée : Amazon EBSCSIDriver Policy

Cette `AmazonEBSCSIDriverPolicy` politique permet au pilote Amazon EBS Container Storage Interface (CSI) de créer, modifier, copier, joindre, détacher et supprimer des volumes en votre nom. Cela inclut la modification des balises sur les volumes existants et l'activation de la restauration rapide des instantanés (FSR) sur les volumes EBS. Il accorde également au pilote EBS CSI les autorisations nécessaires pour créer, verrouiller, restaurer et supprimer des instantanés, ainsi que pour répertorier vos instances, volumes et instantanés.

Pour consulter la dernière version du document de politique JSON, consultez Amazon EBSCSIDriver ServiceRolePolicy dans le AWS Managed Policy Reference Guide.

AWS politique gérée : Amazon EFSCSIDriver Policy

La politique `AmazonEFSCSIDriverPolicy` permet à l'interface CSI (Amazon EFS Container Storage Interface) de créer et de supprimer des points d'accès en votre nom. Il autorise également le pilote CSI Amazon EFS à répertorier vos points d'accès, vos systèmes de fichiers, vos cibles de montage et les zones de disponibilité Amazon EC2.

Pour consulter la dernière version du document de politique JSON, consultez Amazon EFSCSIDriver ServiceRolePolicy dans le AWS Managed Policy Reference Guide.

AWS politique gérée : Amazon EKSLocal OutpostClusterPolicy

Vous ne pouvez pas attacher cette politique à des entités IAM. Avant de créer un cluster local, vous devez associer cette politique à votre rôle de cluster. Les clusters Kubernetes gérés par Amazon EKS effectuent des appels vers d'autres AWS services en votre nom. Ils le font pour gérer les ressources que vous utilisez avec le service.

La politique `AmazonEKSLocalOutpostClusterPolicy` inclut les autorisations suivantes :

- **`ec2`** lecture d'actions : permet aux instances du plan de contrôle de décrire les propriétés de la zone de disponibilité, de la table de routage, de l'instance et de l'interface réseau. Autorisations requises pour que les instances Amazon EC2 puissent rejoindre le cluster en tant qu'instances du plan de contrôle.
- **`ssm`** : permet à Amazon EC2 Systems Manager de se connecter à l'instance du plan de contrôle, qui est utilisée par Amazon EKS pour communiquer et gérer le cluster local dans votre compte.
- **`logs`**— Permet aux instances de transmettre des journaux à Amazon CloudWatch.
- **`secretsmanager`**— Permet aux instances d'obtenir et de supprimer des données de démarrage pour les instances du plan de contrôle en toute sécurité à partir de AWS Secrets Manager.
- **`ecr`** : permet aux pods et aux conteneurs qui s'exécutent sur les instances du plan de contrôle d'extraire les images de conteneurs stockées dans Amazon Elastic Container Registry.

Pour consulter la dernière version du document de politique JSON, consultez Amazon EKSLocal OutpostClusterPolicy dans le AWS Managed Policy Reference Guide.

AWS politique gérée : Amazon EKSLocal OutpostServiceRolePolicy

Vous ne pouvez pas associer cette politique à vos entités IAM. Lorsque vous créez un cluster à l'aide d'un principal IAM `iam:CreateServiceLinkedRole` autorisé, Amazon EKS crée automatiquement

le rôle lié au service AWSServiceRoleforAmazonEKSLocalOutpost pour vous et y associe cette politique. Cette politique permet au rôle lié au service d'appeler des AWS services en votre nom pour les clusters locaux.

La politique `AmazonEKSLocalOutpostServiceRolePolicy` inclut les autorisations suivantes :

- **`ec2`** : permet à Amazon EKS de fonctionner avec des ressources de sécurité, de réseau et autres afin de lancer et de gérer avec succès les instances du plan de contrôle dans votre compte.
- **`ssm`**, **`ssmmessages`** : permet à Amazon EC2 Systems Manager de se connecter aux instances du plan de contrôle, qui sont utilisées par Amazon EKS pour communiquer et gérer le cluster local de votre compte.
- **`iam`** : permet à Amazon EKS de gérer le profil d'instance associé aux instances du plan de contrôle.
- **`secretsmanager`**- Permet à Amazon EKS de placer les données de démarrage des instances du plan de contrôle dans AWS Secrets Manager afin qu'elles puissent être référencées de manière sécurisée lors du démarrage des instances.
- **`outposts`** : permet à Amazon EKS d'obtenir des informations Outpost à partir de votre compte afin de lancer avec succès un cluster local dans un Outpost.

Pour consulter la dernière version du document de politique JSON, consultez Amazon EKSLocal OutpostServiceRolePolicy dans le AWS Managed Policy Reference Guide.

Amazon EKS met à jour les politiques AWS gérées

Consultez les informations relatives aux mises à jour des politiques AWS gérées pour Amazon EKS depuis que ce service a commencé à suivre ces modifications.

Pour recevoir des notifications concernant toutes les modifications apportées aux fichiers sources de cette page de documentation spécifique, vous pouvez vous abonner à l'URL suivante à l'aide d'un lecteur RSS :

```
https://github.com/awsdocs/amazon-eks-user-guide/commits/mainline/latest/ug/security/
iam-reference/security-iam-awsmanpol.adoc.atom
```

Modifier	Description	Date
Ajout d'autorisations à the section called "AWS politique gérée : AWSService RoleForAmazon EKSNodegr oup".	Ajouté `autoscali ng:PutWar mPool autoscali ng:DeleteWarmPool` , et `autoscaling:Descri beWarmPool` autorisations pour`AWSServiceRoleForA mazonEKSNodegroup` . Cela permet aux groupes de nœuds gérés par Amazon EKS de gérer les ressources du warm pool ASG sous-jace ntes tout au long du cycle de vie des groupes de nœuds.	17 février 2026
Autorisation ajoutée à the section called "AWS politique gérée : Amazon EKSService RolePolicy".	Suppression de l'exigenc e de préfixe « eks » dans le nom du profil d'instanc e cible pour l'`iam:GetIn stanceProfile` autorisat ion dans`AmazonEKS ServiceRolePolicy` . Cela permet au mode automatique d'Amazon EKS de valider et d'utiliser des profils d'instance personnal isés NodeClasses sans avoir besoin du préfixe de dénomination « eks ».	2 février 2026
Autorisations ajoutées à Amazon EBSCSIDriver Policy.	Ajout `ec2:LockSnapshot` d'une autorisation permettant au pilote EBS CSI de verrouill	15 janvier 2026

Modifier	Description	Date
	er directement les instantanés EBS.	
Introduction de the section called "AWS politique gérée : Amazon EKSMCPRead OnlyAccess".	Amazon EKS a introduit une nouvelle politique gérée `AmazonEKSMCPReadOnlyAccess` pour activer les outils en lecture seule sur le serveur Amazon EKS MCP à des fins d'observabilité et de résolution des problèmes.	21 novembre 2025
Autorisations ajoutées à Amazon EBSCSIDriver Policy.	`ec2:CopyVolumes` Autorisation ajoutée permettant au pilote EBS CSI de copier directement les volumes EBS.	17 novembre 2025
Autorisation ajoutée à the section called "AWS politique gérée : Amazon EKSService RolePolicy".	Ajouté `ec2:DescribeRouteTables` et `ec2:DescribeNetworkAcls` autorisations pour`AmazonEKSServiceRolePolicy` . Cela permet à Amazon EKS de détecter les problèmes de configuration liés aux tables de routage VPC et au réseau ACLs pour les nœuds hybrides dans le cadre de l'analyse du cluster.	22 octobre 2025
Autorisation ajoutée à AWSServiceRoleForAmazonEKSConnector	`ssmmessages:OpenDataChannel` Autorisation ajoutée pour `AmazonEKSConnectorServiceRolePolicy`	15 octobre 2025

Modifier	Description	Date
Autorisation ajoutée à the section called "AWS politique gérée : Amazon EKSService RolePolicy"	Ce rôle peut associer une nouvelle stratégie d'accès `AmazonEKSEventPolicy` . Autorisations restreintes pour `ec2:DeleteLaunchTemplate` et `ec2:TerminateInstances` .	26 août 2025
Autorisation ajoutée à the section called "AWS politique gérée : Amazon EKSLocal OutpostServiceRolePolicy"	Autorisation `ssmmessages:OpenDataChannel` ajoutée à `AmazonEKSLocalOutpostServiceRolePolicy` .	26 juin 2025
Autorisation ajoutée à the section called "AWS politique gérée : Amazon EKSCompute Policy".	Mise à jour des autorisations de ressources pour les actions `ec2:RunInstances` et `ec2:CreateFleet` afin d'inclure les réserves de capacité `arn:aws:ec2:*:*:capacity-reservation/*` . Cela permet au mode automatique Amazon EKS de lancer des instances à l'aide des réserves de capacité à la demande EC2 de votre compte. Ajout de `iam:CreateServiceLinkedRole` pour permettre au mode automatique Amazon EKS de créer le rôle lié à un service EC2 Spot `AWSServiceRoleForEC2Spot` en votre nom.	20 juin 2025

Modifier	Description	Date
Autorisation ajoutée à Amazon EKSService RolePolicy.	Ajout de l'autorisation `ec2:DescribeCapacityReservations` pour permettre au mode automatique Amazon EKS de lancer des instances à l'aide des réserves de capacité à la demande EC2 de votre compte.	20 juin 2025
Introduction de the section called "AWS politique gérée : Amazon EKSDashboard ConsoleReadOnly".	Introduction de la nouvelle politique `AmazonEKSDashboardConsoleReadOnly` .	19 juin 2025
Introduction de the section called "AWS politique gérée : Amazon EKSDashboard ServiceRolePolicy".	Introduction de la nouvelle politique `AmazonEKSDashboardServiceRolePolicy` .	21 mai 2025
Autorisations ajoutées à Amazon EKSCluster Policy.	Ajout de l'autorisation `ec2:DeleteNetworkInterfaces` pour permettre à Amazon EKS de supprimer les interfaces réseau Elastic qui sont laissées de côté si le VPC CNI se ferme de manière inattendue.	16 avril 2025

Modifier	Description	Date
Autorisation ajoutée à Amazon EKSService RolePolicy.	Ajout `ec2:RevokeSecurityGroupEgress` d'`ec2:AuthorizeSecurityGroupEgress` autorisations permettant aux AI/ML clients d'EKS d'ajouter des règles de sortie du groupe de sécurité au cluster EKS SG par défaut compatibles avec EFA, dans le cadre de la version 1.33 d'EKS.	14 avril 2025
Des autorisations ont été ajoutées à Amazon EKSService RolePolicy.	Ajout d'une autorisation pour résilier les instances EC2 créées par le mode automatique EKS.	28 février 2025

Modifier	Description	Date
Autorisations ajoutées à Amazon EBSCSIDriver Policy.	Ajout d'une nouvelle déclaration autorisant le pilote CSI EBS à restaurer tous les instantanés. Cela était déjà autorisé par la politique existante, mais une nouvelle déclaration explicite est nécessaire en raison d'un changement dans la gestion de l'IAM pour `CreateVolume` . Ajout de la possibilité pour le pilote CSI EBS de modifier les balises sur les volumes existants. Le pilote EBS CSI peut modifier les balises des volumes existants via un paramètre dans Kubernetes. VolumeAttributesClasses Ajout de la possibilité pour le pilote CSI EBS d'activer la restauration rapide des instantanés (FSR) sur les volumes EBS. Le pilote CSI EBS peut activer la FSR sur les nouveaux volumes via des paramètres dans les classes de stockage Kubernetes.	13 janvier 2025

Modifier	Description	Date
Ajout d'autorisations à the section called "AWS politique gérée : Amazon EKSLoad BalancingPolicy".	Mise à jour d'`AmazonEKS LoadBalancingPolic y` pour permettre la liste et la description des ressources réseau et des adresses IP.	26 décembre 2024
Ajout d'autorisations à the section called "AWS politique gérée : AWSService RoleForAmazon EKSNodegr oup".	`AWSServiceRoleForA mazonEKSNodegroup` mis à jour pour être compatible avec les régions chinoises.	22 novembre 2024
Ajout d'autorisations à the section called "AWS politique gérée : Amazon EKSLocal OutpostClusterPolicy"	`ec2:DescribeAvaila bilityZones` Autorisat ion ajoutée `AmazonEKS LocalOutpostCluste rPolicy` afin que le AWS Cloud Controller Manager du plan de contrôle du cluster puisse identifier la zone de disponibilité dans laquelle se trouve chaque nœud.	21 novembre 2024
Ajout d'autorisations à the section called "AWS politique gérée : AWSService RoleForAmazon EKSNodegr oup".	Politique `AWSServic eRoleForAmazonEKSN odegroup` mise à jour pour autoriser `ec2:Reboo tInstances` sur les instances créées par des groupes de nœuds gérés par Amazon EKS. Restriction des autorisations `ec2:Creat eTags` pour les ressources Amazon EC2.	20 novembre 2024

Modifier	Description	Date
Ajout d'autorisations à the section called "AWS politique gérée : Amazon EKSService RolePolicy".	Politique AWS gérée mise à jour par EKS`AmazonEKS ServiceRolePolicy` . Ajout d'autorisations pour les stratégies d'accès EKS, la gestion des équilibreurs de charge et le nettoyage automatisé des ressources du cluster.	16 novembre 2024
Introduction de the section called "AWS politique gérée : Amazon EKSCompute Policy".	Politique AWS gérée mise à jour par EKS`AmazonEKS ComputePolicy` . Autorisations de ressources mises à jour pour l'action `iam:AddRoleToInstanceProfile` .	7 novembre 2024
Introduction de the section called "AWS politique gérée : Amazon EKSCompute Policy".	AWS a présenté le`AmazonEKSComputePolicy` .	1er novembre 2024
Ajout d'autorisations à `AmazonEKSClusterPolicy`	Ajout d'une autorisation `ec2:DescribeInstanceTopology` permettant à Amazon EKS d'associer des informations de topologie au nœud sous forme d'étiquettes.	1er novembre 2024
Introduction de the section called "AWS politique gérée : Amazon EKSBlock StoragePolicy".	AWS a présenté le`AmazonEKSBlockStoragePolicy` .	30 octobre 2024

Modifier	Description	Date
Introduction de the section called "AWS politique gérée : Amazon EKSLoad Balancing Policy".	AWS a présenté leAmazonEKSLoadBalancingPolicy .	30 octobre 2024
Des autorisations ont été ajoutées à Amazon EKSService RolePolicy.	Des cloudwatch:PutMetricData autorisations ont été ajoutées pour permettre à Amazon EKS de publier des métriques sur Amazon CloudWatch.	29 octobre 2024
Introduction de the section called "AWS politique gérée : Amazon EKSNetworking Policy".	AWS a présenté leAmazonEKSNetworkingPolicy .	28 octobre 2024
Ajout d'autorisations à AmazonEKSServicePolicy et AmazonEKS ServiceRolePolicy	Ajout de ec2:GetSecurityGroupsForVpc et association d'autorisations de balises pour permettre à EKS de lire les informations relatives aux groupes de sécurité et de mettre à jour les balises associées.	10 octobre 2024
Présentation d'Amazon EKSWorker NodeMinimalPolicy.	AWS a présenté leAmazonEKSWorkerNodeMinimalPolicy .	3 octobre 2024

Modifier	Description	Date
Ajout d'autorisations à AWSServiceRoleForAmazonEKSNodegroup.	Ajout des autorisations `autoscaling:ResumeProcesses` et `autoscaling:SuspendProcesses` pour permettre à Amazon EKS de suspendre et de reprendre `AZRebalance` dans les groupes Auto Scaling gérés par Amazon EKS.	21 août 2024
Ajout d'autorisations à AWSServiceRoleForAmazonEKSNodegroup.	Ajout de l'autorisation `ec2:DescribeCapacityReservations` pour permettre à Amazon EKS de décrire la réserve de capacité dans le compte de l'utilisateur. Ajout de l'autorisation `autoscaling:PutScheduledUpdateGroupAction` pour permettre la configuration d'une mise à l'échelle planifiée sur les groupes de nœuds `CAPACITY_BLOCK` .	27 juin 2024

Modifier	Description	Date
AmazonEKS_CNI_Policy : mise à jour d'une politique existante	Amazon EKS a ajouté de nouvelles autorisations `ec2:DescribeSubnets` pour permettre au plug-in CNI Amazon VPC pour Kubernetes de voir le nombre d'adresses IP libres dans vos sous-réseaux Amazon VPC. Le CNI VPC peut utiliser les adresses IP libres dans chaque sous-réseau pour sélectionner les sous-réseaux disposant du plus grand nombre d'adresses IP libres à utiliser lors de la création d'une interface réseau Elastic.	4 mars 2024
Amazon EKSWorker NodePolicy — Mise à jour d'une politique existante	Amazon EKS a ajouté de nouvelles autorisations pour permettre les identités du pod EKS. L'agent d'identité du pod Amazon EKS utilise le rôle de nœud.	26 novembre 2023
Présentation de la EFSCSIDriverpolitique d'Amazon.	AWS a présenté le`AmazonEFSCSIDriver Policy` .	26 juillet 2023

Modifier	Description	Date
Autorisations ajoutées à Amazon EKSCluster Policy.	Ajout d'une autorisation `ec2:DescribeAvailabilityZones` permettant à Amazon EKS d'obtenir les détails des zones de disponibilité lors de la découverte automatique des sous-réseaux lors de la création d'équilibreurs de charge.	7 février 2023
Conditions de politique mises à jour dans Amazon EBSCSIDriver Policy.	Conditions de stratégie non valides supprimées comportant des caractères génériques dans le champ clé `StringLike` . Nouvelle condition `ec2:ResourceTag/kubernetes.io/created-for/pvc/name: "*"` ajoutée à `ec2:DeleteVolume` , permettant au pilote EBS CSI de supprimer les volumes créés par le plug-in intégré à l'arborescence.	17 novembre 2022

Modifier	Description	Date
Des autorisations ont été ajoutées à Amazon EKSLocal OutpostServiceRolePolicy.	Ajout de `ec2:DescribeVPCAttribute` , `ec2:GetConsoleOutput` et `ec2:DescribeSecret` pour permettre une meilleure validation des prérequis et un meilleur contrôle du cycle de vie géré. Également, ajout de `ec2:DescribePlacementGroups` et `"arn:aws:ec2:*:*:placement-group/*"` à `ec2:RunInstances` pour permettre le contrôle du placement des instances Amazon EC2 du plan de contrôle sur les Outposts.	24 octobre 2022
Mettez à jour les autorisations d'Amazon Elastic Container Registry sur Amazon EKSLocal OutpostClusterPolicy.	Déplacement de l'action `ecr:GetDownloadUrlForLayer` de toutes les sections de ressources vers une section délimitée. Ajout de la ressource `arn:aws:ecr:*:*:repository/eks/` . Suppression de la ressource `arn:aws: ecr:`. Cette ressource est couverte par la ressource `arn:aws:ecr:*:*:repository/eks/*` ajoutée.	20 octobre 2022

Modifier	Description	Date
Des autorisations ont été ajoutées à Amazon EKSLocal OutpostClusterPolicy.	Ajout du référentiel Amazon Elastic Container Registry `arn:aws: ecr:*:*:repository/kubelet-config-updater` pour que les instances du plan de contrôle du cluster puissent mettre à jour des arguments `kubelet`.	31 août 2022
Présentation d'Amazon EKSLocal OutpostClusterPolicy.	AWS a présenté le`AmazonEKSLocalOutpostClusterPolicy` .	24 août 2022
Présentation d'Amazon EKSLocal OutpostServiceRolePolicy.	AWS a présenté le`AmazonEKSLocalOutpostServiceRolePolicy` .	23 août 2022
Présentation de la EBSCSIDriverpolitique d'Amazon.	AWS a présenté le`AmazonEBSCSIDriverPolicy` .	4 avril 2022
Des autorisations ont été ajoutées à Amazon EKSWorker NodePolicy.	Ajouté `ec2:DescribeInstanceTypes` pour permettre à Amazon EKS d'être optimisé pour détecter automatiquement AMIs les propriétés au niveau de l'instance.	21 mars 2022

Modifier	Description	Date
Ajout d'autorisations à AWSServiceRoleForAmazonEKSNodegroup.	Ajout d'autorisations `autoscaling:EnableMetricsCollection` permettant à Amazon EKS d'activer la collecte de métriques.	13 décembre 2021
Autorisations ajoutées à Amazon EKSCluster Policy.	Ajout de `ec2:DescribeAccountAttributes` , `ec2:DescribeAddresses` et `ec2:DescribeInternetGateways` pour autoriser Amazon EKS à créer un rôle lié à un service pour un Network Load Balancer.	17 juin 2021
Amazon EKS a commencé à assurer le suivi des modifications.	Amazon EKS a commencé à suivre les modifications apportées AWS à ses politiques gérées.	17 juin 2021

Dépannage IAM

Cette rubrique traite de certaines erreurs courantes que vous pouvez rencontrer lorsque vous utilisez Amazon EKS avec IAM, ainsi que des solutions.

AccessDeniedException

Si vous recevez un message `AccessDeniedException` lors de l'appel d'une opération d' AWS API, cela signifie que les informations d'identification principales IAM que vous utilisez ne disposent pas des autorisations requises pour effectuer cet appel.

```
An error occurred (AccessDeniedException) when calling the DescribeCluster operation:
User: arn:aws: iam::111122223333:user/user_name is not authorized to perform:
```

```
eks:DescribeCluster on resource: arn:aws: eks:region:111122223333:cluster/my-cluster
```

Dans l'exemple de message ci-précédent, l'utilisateur n'est pas autorisé à appeler l'opération d'API `DescribeCluster` Amazon EKS. Pour accorder des autorisations d'administrateur Amazon EKS à un principal IAM, consultez la rubrique the section called “Politiques basées sur l'identité”.

Pour plus d'informations générales sur IAM, consultez Contrôle de l'accès à l'aide de politiques dans le Guide de l'utilisateur IAM.

Impossible de voir les nœuds dans l'onglet Calcul ou dans l'onglet Ressources et vous recevez un message d'erreur dans le AWS Management Console

Vous pouvez voir un message d'erreur de la console indiquant `Your current user or role does not have access to Kubernetes objects on this EKS cluster`. Assurez-vous que l'utilisateur principal IAM AWS Management Console avec lequel vous utilisez dispose des autorisations nécessaires. Pour de plus amples informations, veuillez consulter the section called “Autorisations requises”.

aws-auth **`ConfigMap`** n'accorde pas l'accès au cluster

L'AWS IAM Authenticator n'autorise pas de chemin dans l'ARN de rôle utilisé dans le `ConfigMap`. Par conséquent, avant de spécifier `rolearn`, supprimez le chemin d'accès. Remplacez, par exemple, `arn:aws: iam::111122223333:role/team/developers/eks-admin` par `arn:aws: iam::111122223333:role/eks-admin` .

Je ne suis pas autorisé à effectuer iam : PassRole

Si vous recevez une erreur indiquant que vous n'êtes pas autorisé à effectuer l'action `iam:PassRole`, vos politiques doivent être mises à jour pour vous permettre de transmettre un rôle à Amazon EKS.

Certains AWS services vous permettent de transmettre un rôle existant à ce service au lieu de créer un nouveau rôle de service ou un rôle lié à un service. Pour ce faire, vous devez disposer des autorisations nécessaires pour transmettre le rôle au service.

L'exemple d'erreur suivant se produit lorsqu'un utilisateur IAM nommé `marymajor` essaie d'utiliser la console pour effectuer une action dans Amazon EKS. Toutefois, l'action nécessite que le service ait des autorisations accordées par une fonction du service. Mary n'est pas autorisée à transmettre le rôle au service.

```
User: {arn-aws}iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

Dans ce cas, les politiques de Mary doivent être mises à jour pour lui permettre d'effectuer l'action `iam:PassRole`.

Si vous avez besoin d'aide, contactez votre AWS administrateur. Votre administrateur vous a fourni vos informations d'identification de connexion.

Je souhaite autoriser des personnes extérieures à mon AWS compte à accéder à mes ressources Amazon EKS

Vous pouvez créer un rôle que les utilisateurs provenant d'autres comptes ou les personnes extérieures à votre organisation pourront utiliser pour accéder à vos ressources. Vous pouvez spécifier qui est autorisé à assumer le rôle. Pour les services qui prennent en charge les politiques basées sur les ressources ou les listes de contrôle d'accès (ACLs), vous pouvez utiliser ces politiques pour autoriser les utilisateurs à accéder à vos ressources.

Pour plus d'informations, consultez les éléments suivants :

- Pour savoir si Amazon EKS est compatible avec ces fonctionnalités, consultez the section called "Amazon EKS et IAM".
- Pour savoir comment fournir un accès à vos ressources sur les AWS comptes que vous possédez, consultez la section Fournir un accès à un utilisateur IAM sur un autre AWS compte que vous possédez dans le Guide de l'utilisateur IAM.
- Pour savoir comment fournir l'accès à vos ressources à des AWS comptes tiers, consultez la section Fournir un accès aux AWS comptes détenus par des tiers dans le guide de l'utilisateur IAM.
- Pour savoir comment fournir un accès par le biais de la fédération d'identité, consultez Fournir un accès à des utilisateurs authentifiés en externe (fédération d'identité) dans le Guide de l'utilisateur IAM.
- Pour en savoir plus sur la différence entre l'utilisation des rôles et des politiques basées sur les ressources pour l'accès intercompte, consultez Accès intercompte aux ressources dans IAM dans le Guide de l'utilisateur IAM.

Les conteneurs Pod reçoivent l'erreur suivante : **An error occurred (SignatureDoesNotMatch) when calling the GetCallerIdentity operation: Credential should be scoped to a valid region**

Vos conteneurs reçoivent cette erreur si votre application envoie explicitement des demandes au point de terminaison global AWS STS (`https://sts.amazonaws`) et si votre compte de service Kubernetes est configuré pour utiliser un point de terminaison régional. Vous pouvez résoudre le problème avec l'une des options suivantes :

- Mettez à jour le code de votre application pour supprimer les appels explicites au point de terminaison global AWS STS.
- Mettez à jour le code de votre application pour effectuer des appels explicites vers des points de terminaison régionaux tels que `https://sts.us-west-2.amazonaws.com`. Votre application doit comporter une redondance intégrée pour choisir une autre AWS région en cas de panne du service dans la AWS région. Pour plus d'informations, consultez la section Gestion du AWS STS dans une AWS région dans le guide de l'utilisateur IAM.
- Configurez vos comptes de service pour utiliser le point de terminaison mondial. Les clusters utilisent le point de terminaison régional par défaut. Pour de plus amples informations, veuillez consulter the section called “Critères d’évaluation STS”.

Rôle IAM de cluster Amazon EKS

Un rôle IAM du cluster Amazon EKS est requis pour chaque cluster. Les clusters Kubernetes gérés par Amazon EKS utilisent ce rôle pour gérer les nœuds, tandis que l’ancien fournisseur de cloud utilise ce rôle pour créer des équilibreurs de charge avec Elastic Load Balancing pour les services.

Avant de pouvoir créer des clusters Amazon EKS, vous devez créer un rôle IAM avec l'une des politiques IAM suivantes :

- EKSClusterPolitique d'Amazon
- Une politique IAM personnalisée. Les autorisations minimales ci-dessous permettent au cluster Kubernetes de gérer les nœuds, mais n’autorisent pas l’ancien fournisseur de cloud à créer des équilibreurs de charge avec Elastic Load Balancing. Votre politique IAM personnalisée doit disposer au moins des autorisations suivantes :

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

```
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:CreateTags"
      ],
      "Resource": "arn:aws:ec2:*:*:instance/*",
      "Condition": {
        "ForAnyValue:StringLike": {
          "aws:TagKeys": "kubernetes.io/cluster/*"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeInstances",
        "ec2:DescribeNetworkInterfaces",
        "ec2:DescribeVpcs",
        "ec2:DescribeDhcpOptions",
        "ec2:DescribeAavailabilityZones",
        "ec2:DescribeInstanceTopology",
        "kms:DescribeKey"
      ],
      "Resource": "*"
    }
  ]
}
```

> **Note**
>
> Avant le 3 octobre 2023, Amazon EKSCluster Policy était obligatoire concernant le rôle IAM de chaque cluster.
> Avant le 16 avril 2020, Amazon EKSService Policy et Amazon EKSCluster Policy étaient obligatoires et le nom suggéré pour le rôle était`eksServiceRole`. Avec le rôle `AWSServiceRoleForAmazonEKS` lié au service, la EKSServicepolitique Amazon Policy n'est plus requise pour les clusters créés le 16 avril 2020 ou après cette date.

Recherche d'un rôle de cluster existant

Vous pouvez utiliser la procédure suivante pour vérifier si votre compte possède déjà le rôle de cluster Amazon EKS.

1. Ouvrez la console IAM à https://console.aws.amazon.com/iam/ l'adresse.
2. Dans le volet de navigation de gauche, choisissez Rôles.
3. Recherchez dans la liste des rôles `eksClusterRole`. Si un rôle qui inclut `eksClusterRole` n'existe pas, consultez the section called "Création du rôle de cluster Amazon EKS" pour créer le rôle. Si un rôle qui inclut `eksClusterRole` existe, sélectionnez le rôle pour afficher les politiques attachées.
4. Choisissez Permissions (Autorisations).
5. Assurez-vous que la EKSClusterpolitique gérée par Amazon Policy est attachée au rôle. Si la politique est attachée, votre rôle de cluster Amazon EKS est configuré correctement.
6. Sélectionnez l'onglet Trust relationships (Relations d'approbation), puis Edit trust policy (Modifier la relation d'approbation).
7. Vérifiez que la relation d'approbation contient la politique suivante. Si la relation d'approbation correspond à la politique ci-dessous, sélectionnez Annuler. Si la relation d'approbation ne correspond pas, copiez la politique dans la fenêtre Modifier la politique d'approbation et sélectionnez Mettre à jour la politique.

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

Création du rôle de cluster Amazon EKS

Vous pouvez utiliser la AWS Management Console ou la AWS CLI pour créer le rôle de cluster.

AWS Management Console

a. Ouvrez la console IAM à https://console.aws.amazon.com/iam/ l'adresse.
b. Choisissez Roles (Rôles) puis Create role (Créer un rôle).
c. Sous Type d'entité de confiance, sélectionnez AWS service.
d. Dans la liste déroulante Cas d'utilisation d'autres AWS services, sélectionnez EKS.
e. Sélectionnez EKS - Cluster pour votre cas d'utilisation, puis Next (Suivant).
f. Sous l'onglet Add permissions (Ajouter des autorisations), sélectionnez Next (Suivant).
g. Pour Role name (Nom de rôle), saisissez un nom unique pour votre rôle, par exemple, `eksClusterRole`.
h. Pour Description, saisissez un texte descriptif tel que `Amazon EKS - Cluster role`.
i. Choisissez Créer un rôle.

AWS CLI

a. Copiez le contenu suivant dans un fichier nommé *`cluster-trust-policy.json`*.

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

b. Créez le rôle. Vous pouvez remplacer *`eksClusterRole`* par n'importe quel nom que vous choisissez.

```
aws iam create-role \
  --role-name eksClusterRole \
  --assume-role-policy-document file://"cluster-trust-policy.json"
```

c. Attachez la politique IAM requise au rôle.

```
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::aws:policy/AmazonEKSClusterPolicy \
```

```
  --role-name eksClusterRole
```

Rôle IAM de nœud Amazon EKS

Le `kubelet` démon du nœud Amazon EKS effectue des appels AWS APIs en votre nom. Les nœuds reçoivent l'autorisation pour ces appels d'API via un profil d'instance IAM et les politiques associées. Avant de pouvoir lancer les nœuds et les enregistrer dans un cluster, vous devez créer un rôle IAM qui sera utilisé par ces nœuds lors de leur lancement. Cette exigence s'applique aux nœuds lancés avec l'AMI optimisée Amazon EKS fournie par Amazon, ou avec tout autre nœud AMIs que vous avez l'intention d'utiliser. En outre, cette exigence s'applique à la fois aux groupes de nœuds gérés et aux nœuds autogérés.

> **Note**
>
> Vous ne pouvez pas utiliser le même rôle que celui utilisé pour créer des clusters.

Avant de créer des nœuds, vous devez créer un rôle IAM avec les autorisations suivantes :

- Autorisations permettant `kubelet` de décrire les EC2 ressources Amazon dans le VPC, telles que prévues par la politique d'Amazon EKSWorker NodePolicy. Cette politique fournit également les autorisations pour l'agent d'identité du pod Amazon EKS.
- Autorisations relatives `kubelet` à l'utilisation d'images de conteneurs issues d'Amazon Elastic Container Registry (Amazon ECR), telles que prévues par la politique d'Amazon. EC2 ContainerRegistryPullOnly Les autorisations d'utiliser des images de conteneurs provenant d'Amazon Elastic Container Registry (Amazon ECR) sont nécessaires parce que les modules complémentaires intégrés pour la mise en réseau exécutent des pods qui utilisent des images de conteneurs provenant d'Amazon ECR.
- (Facultatif) Autorisations permettant à l'agent d'identité du pod Amazon EKS d'utiliser l'action `eks-auth:AssumeRoleForPodIdentity` pour récupérer les informations d'identification pour les pods. Si vous n'utilisez pas Amazon EKSWorker NodePolicy, vous devez fournir cette autorisation en plus des EC2 autorisations d'utilisation d'EKS Pod Identity.
- (Facultatif) Si vous n'utilisez pas IRSA ou l'identité du pod EKS pour accorder des autorisations aux pods VPC CNI, vous devez alors attribuer les autorisations du VPC CNI au rôle de l'instance. Vous pouvez utiliser la politique `AmazonEKS_CNI_Policy`gérée (si vous avez créé votre cluster avec la `IPv4` famille) ou une IPv6 politique que vous créez (si vous avez créé votre cluster avec

la IPv6 famille). Cependant, plutôt que d'attacher la politique à ce rôle, nous vous recommandons de l'attacher à un rôle distinct utilisé spécifiquement pour le module complémentaire Amazon VPC CNI. Pour plus d'informations sur la création d'un rôle distinct pour le module complémentaire Amazon VPC CNI, veuillez consulter the section called “Configuration pour IRSA”.

> **Note**
>
> Les groupes de EC2 nœuds Amazon doivent avoir un rôle IAM différent de celui du profil Fargate. Pour de plus amples informations, veuillez consulter the section called “Rôle IAM d'exécution de pod”.

Recherche d'un rôle de nœud existant

Vous pouvez utiliser la procédure suivante pour vérifier si votre compte possède déjà le rôle de nœud Amazon EKS.

1. Ouvrez la console IAM à https://console.aws.amazon.com/iam/ l'adresse.
2. Dans le volet de navigation de gauche, choisissez Rôles.
3. Recherchez `eksNodeRole`, `AmazonEKSNodeRole` ou `NodeInstanceRole` dans la liste des rôles. Si aucun rôle portant l'un de ces noms n'existe, consultez la section the section called “Création du rôle IAM de nœud Amazon EKS” pour créer le rôle. Si un rôle qui contient `eksNodeRole`, `AmazonEKSNodeRole` ou `NodeInstanceRole` existe, sélectionnez-le pour afficher les politiques attachées.
4. Choisissez Autorisations.
5. Assurez-vous que les politiques EC2 ContainerRegistryPullOnly gérées par Amazon EKSWorker NodePolicy et Amazon sont associées au rôle ou qu'une politique personnalisée est attachée avec les autorisations minimales.

> **Note**
>
> Si la politique AmazonEKS_CNI_Policy est attachée au rôle, nous vous recommandons de la supprimer et de l'attacher à un rôle IAM qui est plutôt mappé au compte de service `aws-node` Kubernetes. Pour de plus amples informations, veuillez consulter the section called “Configuration pour IRSA”.

6. Sélectionnez l'onglet Trust relationships (Relations d'approbation), puis Edit trust policy (Modifier la relation d'approbation).
7. Vérifiez que la relation d'approbation contient la politique suivante. Si la relation d'approbation correspond à la politique ci-dessous, sélectionnez Annuler. Si la relation d'approbation ne correspond pas, copiez la politique dans la fenêtre Modifier la politique d'approbation et sélectionnez Mettre à jour la politique.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sts:AssumeRole"
            ],
            "Principal": {
                "Service": [
                    "ec2.amazonaws.com"
                ]
            }
        }
    ]
}
```

Création du rôle IAM de nœud Amazon EKS

Vous pouvez créer le rôle IAM du nœud à l'aide de la AWS Management Console ou de la AWS CLI.

AWS Management Console

a. Ouvrez la console IAM à https://console.aws.amazon.com/iam/ l'adresse.
b. Dans le volet de navigation de gauche, choisissez Rôles.
c. Sur la page Rôles, choisissez Créer un rôle.
d. Sur la page Select trusted entity (Sélectionner une entité de confiance), procédez comme suit :
    i. Dans la section Type d'entité fiable, sélectionnez AWS service.
    ii. Sous Use case (Cas d'utilisation), choisissez EC2.
    iii. Choisissez Suivant.

e. Sur la page Ajouter des autorisations, associez une stratégie personnalisée ou procédez comme suit :

   i. Dans la zone Filter policies (Politiques de filtre), saisissez `AmazonEKSWorkerNodePolicy`.

   ii. Cochez la case située à gauche d'Amazon EKSWorker NodePolicy dans les résultats de recherche.

   iii. Sélectionnez Clear filters (Effacer les filtres).

   iv. Dans la zone Filter policies (Politiques de filtre), saisissez `AmazonEC2ContainerRegistryPullOnly`.

   v. Cochez la case située à gauche d'Amazon EC2 ContainerRegistryPullOnly dans les résultats de recherche.

   La politique gérée Amazoneks_CNI_Policy ou une politique que vous créez doit également être attachée à ce rôle ou à IPv6 un autre rôle mappé au compte de service Kubernetes. `aws-node` Nous vous recommandons d'attribuer la politique au rôle associé au compte de service Kubernetes au lieu de l'affecter à ce rôle. Pour de plus amples informations, veuillez consulter the section called “Configuration pour IRSA”.

   vi. Choisissez Suivant.

f. Sur la page Name, review, and create (Nommer, vérifier et créer), procédez comme suit :

   i. Pour Role name (Nom de rôle), saisissez un nom unique pour votre rôle, par exemple, `AmazonEKSNodeRole`.

   ii. Pour Description, remplacez le texte actuel par un texte descriptif tel que `Amazon EKS - Node role`.

   iii. Sous Ajouter des balises (Facultatif), ajoutez des métadonnées au rôle en attachant les identifications sous forme de paires clé-valeur. Pour plus d'informations sur l'utilisation des balises dans IAM, consultez la rubrique Balisage des ressources IAM dans le Guide de l'utilisateur IAM.

   iv. Choisissez Créer un rôle.

AWS CLI

a. Exécutez la commande ci-dessous pour créer un fichier`node-role-trust-relationship.json`.

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

```
            "Effect": "Allow",
            "Action": [
                "sts:AssumeRole"
            ],
            "Principal": {
                "Service": [
                    "ec2.amazonaws.com"
                ]
            }
        }
    ]
}
```

b. Créez le rôle IAM.

```
aws iam create-role \
  --role-name AmazonEKSNodeRole \
  --assume-role-policy-document file://"node-role-trust-relationship.json"
```

c. Attachez deux politiques gérées IAM requises au rôle IAM.

```
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::aws:policy/AmazonEKSWorkerNodePolicy \
  --role-name AmazonEKSNodeRole
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::aws:policy/AmazonEC2ContainerRegistryPullOnly \
  --role-name AmazonEKSNodeRole
```

d. Attachez l'une des politiques IAM suivantes au rôle IAM en fonction de la famille IP avec laquelle vous avez créé votre cluster. La politique doit être attachée à ce rôle ou à un rôle associé au compte de service `aws-node` Kubernetes utilisé pour le plug-in Amazon VPC CNI pour Kubernetes. Nous vous recommandons d'attribuer la politique au rôle associé au compte de service Kubernetes. Pour attribuer la politique au rôle associé au compte de service Kubernetes, veuillez consulter the section called “Configuration pour IRSA”.

- IPv4

```
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::aws:policy/AmazonEKS_CNI_Policy \
  --role-name AmazonEKSNodeRole
```

- IPv6

A. Copiez la politique suivante et enregistrez-la dans un fichier appelé `vpc-cni-ipv6-policy.json`.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AssignIpv6Addresses",
                "ec2:DescribeInstances",
                "ec2:DescribeTags",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeInstanceTypes"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:network-interface/*"
            ]
        }
    ]
}
```

B. Créez la politique IAM.

```
aws iam create-policy --policy-name AmazonEKS_CNI_IPv6_Policy --policy-
document file://vpc-cni-ipv6-policy.json
```

C. Attachez la politique IAM au rôle IAM. Remplacez *111122223333* par votre ID de compte.

```
aws iam attach-role-policy \
  --policy-arn arn:aws: iam::111122223333:policy/AmazonEKS_CNI_IPv6_Policy \
  --role-name AmazonEKSNodeRole
```

Rôle IAM du cluster du mode automatique Amazon EKS

Un rôle IAM du cluster Amazon EKS est requis pour chaque cluster. Les clusters Kubernetes gérés par Amazon EKS utilisent ce rôle pour automatiser les tâches courantes liées au stockage, à la mise en réseau et à l'autoscaling de la puissance de calcul.

Avant de pouvoir créer des clusters Amazon EKS, vous devez créer un rôle IAM avec les politiques nécessaires pour le mode automatique EKS. Vous pouvez soit joindre les stratégies gérées par AWS IAM suggérées, soit créer des politiques personnalisées avec des autorisations équivalentes.

- EKSComputePolitique d'Amazon
- AmazonEKSBlockStoragePolicy
- AmazonEKSLoadBalancingPolicy
- EKSNetworkingPolitique d'Amazon
- EKSClusterPolitique d'Amazon

Recherche d'un rôle de cluster existant

Vous pouvez utiliser la procédure suivante pour vérifier si votre compte possède déjà le rôle de cluster Amazon EKS.

1. Ouvrez la console IAM à https://console.aws.amazon.com/iam/ l'adresse.
2. Dans le volet de navigation de gauche, choisissez Rôles.
3. Recherchez dans la liste des rôles `AmazonEKSAutoClusterRole`. Si aucun rôle contenant `AmazonEKSAutoClusterRole` n'existe, consultez les instructions dans la section suivante pour créer le rôle. Si un rôle qui inclut `AmazonEKSAutoClusterRole` existe, sélectionnez le rôle pour afficher les politiques attachées.
4. Choisissez Permissions (Autorisations).
5. Assurez-vous que la EKSClusterpolitique gérée par Amazon Policy est attachée au rôle. Si la politique est attachée, votre rôle de cluster Amazon EKS est configuré correctement.
6. Sélectionnez l'onglet Trust relationships (Relations d'approbation), puis Edit trust policy (Modifier la relation d'approbation).
7. Vérifiez que la relation d'approbation contient la politique suivante. Si la relation d'approbation correspond à la politique ci-dessous, sélectionnez Annuler. Si la relation d'approbation ne correspond pas, copiez la politique dans la fenêtre Modifier la politique d'approbation et sélectionnez Mettre à jour la politique.

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

> **Note**
>
> AWS ne nécessite pas le nom `AmazonEKSAutoClusterRole` de ce rôle.

Création du rôle de cluster Amazon EKS

Vous pouvez utiliser la AWS Management Console ou la AWS CLI pour créer le rôle de cluster.

AWS Management Console

1. Ouvrez la console IAM à https://console.aws.amazon.com/iam/ l'adresse.
2. Choisissez Roles (Rôles) puis Create role (Créer un rôle).
3. Sous Type d'entité de confiance, sélectionnez AWS service.
4. Dans la liste déroulante Cas d'utilisation d'autres AWS services, sélectionnez EKS.
5. Sélectionnez EKS - Cluster pour votre cas d'utilisation, puis Next (Suivant).
6. Dans l'onglet Ajouter des autorisations, sélectionnez les politiques, puis cliquez sur Suivant.
   - EKSComputePolitique d'Amazon
   - AmazonEKSBlockStoragePolicy
   - AmazonEKSLoadBalancingPolicy

- EKSNetworkingPolitique d'Amazon
- EKSClusterPolitique d'Amazon

7. Pour Role name (Nom de rôle), saisissez un nom unique pour votre rôle, par exemple, `AmazonEKSAutoClusterRole`.
8. Pour Description, saisissez un texte descriptif tel que `Amazon EKS - Cluster role`.
9. Choisissez Créer un rôle.

AWS CLI

1. Copiez le contenu suivant dans un fichier nommé *`cluster-trust-policy.json`*.

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

2. Créez le rôle. Vous pouvez remplacer *`AmazonEKSAutoClusterRole`* par n'importe quel nom que vous choisissez.

```
aws iam create-role \
  --role-name AmazonEKSAutoClusterRole \
  --assume-role-policy-document file://"cluster-trust-policy.json"
```

3. Attachez les politiques IAM requises au rôle :

EKSClusterPolitique d'Amazon :

```
aws iam attach-role-policy \
```

```
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSClusterPolicy
```

EKSComputePolitique d'Amazon :

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSComputePolicy
```

Amazon EKSBlock StoragePolicy :

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSBlockStoragePolicy
```

Amazon EKSLoad BalancingPolicy :

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSLoadBalancingPolicy
```

EKSNetworkingPolitique d'Amazon :

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoClusterRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSNetworkingPolicy
```

Rôle IAM de nœud du mode automatique Amazon EKS

> **Note**
>
> Vous ne pouvez pas utiliser le même rôle que celui utilisé pour créer des clusters.

Avant de créer des nœuds, vous devez créer un rôle IAM incluant les politiques suivantes ou des autorisations équivalentes :

- AmazonEKSWorkerNodeMinimalPolicy
- AmazonEC2ContainerRegistryPullOnly

Recherche d'un rôle de nœud existant

Vous pouvez utiliser la procédure suivante pour vérifier si votre compte possède déjà le rôle de nœud Amazon EKS.

1. Ouvrez la console IAM à https://console.aws.amazon.com/iam/ l'adresse.
2. Dans le volet de navigation de gauche, choisissez Rôles.
3. Recherchez dans la liste des rôles `AmazonEKSAutoNodeRole`. Si aucun rôle portant l'un de ces noms n'existe, consultez les instructions de la section suivante pour le créer. Si un rôle qui contient `AmazonEKSAutoNodeRole` existe, sélectionnez le rôle pour afficher les stratégies attachées.
4. Choisissez Autorisations.
5. Assurez-vous que les politiques requises ci-dessus sont bien attachées, ou utilisez des politiques personnalisées équivalentes.
6. Sélectionnez l'onglet Trust relationships (Relations d'approbation), puis Edit trust policy (Modifier la relation d'approbation).
7. Vérifiez que la relation d'approbation contient la politique suivante. Si la relation d'approbation correspond à la politique ci-dessous, sélectionnez Annuler. Si la relation d'approbation ne correspond pas, copiez la politique dans la fenêtre Modifier la politique d'approbation et sélectionnez Mettre à jour la politique.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Création du rôle IAM de nœud Amazon EKS

Vous pouvez créer le rôle IAM du nœud à l'aide de la AWS Management Console ou de la AWS CLI.

AWS Management Console

1. Ouvrez la console IAM à https://console.aws.amazon.com/iam/ l'adresse.
2. Dans le volet de navigation de gauche, choisissez Rôles.
3. Sur la page Rôles, choisissez Créer un rôle.
4. Sur la page Select trusted entity (Sélectionner une entité de confiance), procédez comme suit :
   a. Dans la section Type d'entité fiable, sélectionnez AWS service.
   b. Sous Use case (Cas d'utilisation), choisissez EC2.
   c. Choisissez Suivant.
5. Sur la page Ajouter des autorisations, attachez les politiques suivantes :
   - AmazonEKSWorkerNodeMinimalPolicy
   - AmazonEC2ContainerRegistryPullOnly
6. Sur la page Name, review, and create (Nommer, vérifier et créer), procédez comme suit :
   a. Pour Role name (Nom de rôle), saisissez un nom unique pour votre rôle, par exemple, `AmazonEKSAutoNodeRole`.
   b. Pour Description, remplacez le texte actuel par un texte descriptif tel que `Amazon EKS - Node role`.
   c. Sous Ajouter des balises (Facultatif), ajoutez des métadonnées au rôle en attachant les identifications sous forme de paires clé-valeur. Pour plus d'informations sur l'utilisation des balises dans IAM, consultez la rubrique Balisage des ressources IAM dans le Guide de l'utilisateur IAM.
   d. Choisissez Créer un rôle.

AWS CLI

Création du rôle IAM du nœud

Utilisez le fichier node-trust-policy.json de l'étape précédente pour définir les entités qui peuvent assumer le rôle. Exécutez la commande suivante pour créer le rôle IAM du nœud :

```
aws iam create-role \
    --role-name AmazonEKSAutoNodeRole \
    --assume-role-policy-document file://node-trust-policy.json
```

Enregistrement de l'ARN du rôle

Après la création du rôle, extrayez et enregistrez l'ARN du rôle IAM du nœud. Cet ARN sera nécessaire dans les étapes suivantes. Utilisez la commande suivante pour extraire l'ARN :

```
aws iam get-role --role-name AmazonEKSAutoNodeRole --query "Role.Arn" --output text
```

Association des politiques requises

Associez les politiques AWS gérées suivantes au rôle Node IAM pour fournir les autorisations nécessaires :

Pour joindre Amazon, procédez comme suit EKSWorker NodeMinimalPolicy :

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoNodeRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEKSWorkerNodeMinimalPolicy
```

Pour joindre Amazon, procédez comme suit EC2 ContainerRegistryPullOnly :

```
aws iam attach-role-policy \
    --role-name AmazonEKSAutoNodeRole \
    --policy-arn arn:aws: iam::aws:policy/AmazonEC2ContainerRegistryPullOnly
```

Fonctionnalité Amazon EKS : rôle IAM

Les fonctionnalités EKS nécessitent la configuration d'un rôle IAM de fonctionnalité (ou rôle de capacité). Les fonctionnalités utilisent ce rôle pour effectuer des actions sur les AWS services et pour accéder aux ressources Kubernetes de votre cluster via des entrées d'accès créées automatiquement.

Avant de pouvoir spécifier un rôle de capacité lors de la création d'une fonctionnalité, vous devez créer le rôle IAM avec la politique de confiance et les autorisations appropriées pour le type de capacité. Une fois ce rôle IAM créé, il peut être réutilisé pour n'importe quel nombre de ressources de capacité.

Exigences relatives aux rôles de capacité

Le rôle de capacité doit répondre aux exigences suivantes :

- Le rôle doit figurer dans le même AWS compte que le cluster et la ressource de capacité

- Le rôle doit avoir une politique de confiance qui permet au service des capacités EKS d'assumer le rôle
- Le rôle doit disposer d'autorisations adaptées au type de fonctionnalité et aux exigences du cas d'utilisation (voirthe section called “Autorisations par type de capacité”)

Politique de confiance pour les rôles liés aux capacités

Tous les rôles de capacité doivent inclure la politique de confiance suivante :

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

Cette politique de confiance permet à EKS de :

- Assumez le rôle d'effectuer des opérations AWS d'API
- Étiquetez les sessions à des fins d'audit et de suivi

Autorisations par type de capacité

Les autorisations IAM requises dépendent de la fonctionnalité que vous utilisez et de votre modèle de déploiement.

> **Note**
>
> Pour les déploiements de production utilisant des sélecteurs de rôles IAM avec ACK, ou lorsque vous utilisez Kro ou Argo CD sans intégration de AWS services, le rôle de capacité peut ne nécessiter aucune autorisation IAM au-delà de la politique de confiance.

kro (Kube Resource Orchestrator)

Aucune autorisation IAM n'est requise. Vous pouvez créer un rôle de capacité sans politique attachée. Kro n'a besoin que des autorisations RBAC de Kubernetes pour créer et gérer les ressources Kubernetes.

AWS Contrôleurs pour Kubernetes (ACK)

ACK prend en charge deux modèles d'autorisation :

- Configuration simple (développement/test) : ajoutez des autorisations AWS de service directement au rôle de capacité. Cela fonctionne bien pour la mise en route, les déploiements à compte unique ou lorsque tous les utilisateurs ont besoin des mêmes autorisations.
- Bonnes pratiques de production : utilisez les sélecteurs de rôles IAM pour implémenter l'accès avec le moindre privilège. Avec cette approche, le rôle de capacité n'a besoin que d'une `sts:AssumeRole` autorisation pour assumer des rôles spécifiques au service. Vous n'ajoutez pas d'autorisations de AWS service (comme S3 ou RDS) au rôle de capacité lui-même : ces autorisations sont accordées à des rôles IAM individuels mappés à des espaces de noms spécifiques.

  Les sélecteurs de rôles IAM permettent de :

  - Isolation des autorisations au niveau de l'espace de noms
  - Gestion des ressources entre comptes
  - Rôles IAM spécifiques à l'équipe
  - Modèle de sécurité fondé sur le moindre privilège

  Exemple de politique de rôle de capacité pour l'approche du sélecteur de rôle IAM :

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

```
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Resource": [
        "arn:aws:iam::111122223333:role/ACK-S3-Role",
        "arn:aws:iam::111122223333:role/ACK-RDS-Role",
        "arn:aws:iam::444455556666:role/ACKCrossAccountRole"
      ]
    }
  ]
}
```

Pour une configuration détaillée des autorisations ACK, y compris les sélecteurs de rôle IAM, voir. the section called “Configurer les autorisations ”

Argo CD

Aucune autorisation IAM n'est requise par défaut. Des autorisations facultatives peuvent être nécessaires pour :

- AWS Secrets Manager : si vous utilisez Secrets Manager pour stocker les informations d'identification du référentiel Git
- AWS CodeConnections: en cas d'utilisation CodeConnections pour l'authentification du dépôt Git

Exemple de politique pour Secrets Manager et CodeConnections :

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "secretsmanager:GetSecretValue",
        "secretsmanager:DescribeSecret"
      ],
      "Resource": "arn:aws:secretsmanager:region:account-id:secret:argocd/*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "codeconnections:UseConnection",
        "codeconnections:GetConnection"
      ],
```

```
      "Resource": "arn:aws:codeconnections:region:account-id:connection/*"
    }
  ]
}
```

Pour connaître les exigences détaillées en matière d'autorisation sur Argo CD, voirthe section called "Considérations relatives à Argo CD".

Vérifier la présence d'un rôle de capacité existant

Vous pouvez utiliser la procédure suivante pour vérifier si votre compte possède déjà un rôle IAM de fonctionnalité adapté à votre cas d'utilisation.

1. Ouvrez la console IAM à https://console.aws.amazon.com/iam/ l'adresse.
2. Dans le volet de navigation de gauche, choisissez Rôles.
3. Recherchez dans la liste des rôles le nom de votre rôle de capacité (par exemple, `ACKCapabilityRole` ou`ArgoCDCapabilityRole`).
4. Si un rôle existe, sélectionnez-le pour afficher les politiques et la relation de confiance associées.
5. Sélectionnez l'onglet Trust relationships (Relations d'approbation), puis Edit trust policy (Modifier la relation d'approbation).
6. Vérifiez que la relation de confiance correspond à la politique de confiance des capacités. Si elle ne correspond pas, mettez à jour la politique de confiance.
7. Choisissez Autorisations et vérifiez que le rôle dispose des autorisations appropriées pour votre type de fonctionnalité et votre cas d'utilisation.

Création d'un rôle IAM de capacité

Vous pouvez utiliser la AWS Management Console ou la AWS CLI pour créer un rôle de capacité.

AWS Management Console

a. Ouvrez la console IAM à https://console.aws.amazon.com/iam/ l'adresse.
b. Choisissez Roles (Rôles) puis Create role (Créer un rôle).
c. Sous Type d'entité fiable, sélectionnez Politique de confiance personnalisée.
d. Copiez et collez la politique de confiance des capacités dans l'éditeur de politique de confiance.
e. Choisissez Suivant.

f. Dans l'onglet Ajouter des autorisations, sélectionnez ou créez des politiques adaptées à votre type de fonctionnalité (voirthe section called "Autorisations par type de capacité"). Pour Kro, vous pouvez ignorer cette étape.

g. Choisissez Suivant.

h. Dans Nom du rôle, entrez un nom unique pour votre rôle, tel que `ACKCapabilityRole`ArgoCD`CapabilityRole`, ou`kroCapabilityRole`.

i. Pour Description, saisissez un texte descriptif tel que `Amazon EKS - ACK capability role`.

j. Choisissez Créer un rôle.

AWS CLI

a. Copiez la politique de confiance en matière de capacités dans un fichier nommé`capability-trust-policy.json`.

b. Créez le rôle. `ACKCapabilityRole`Remplacez-le par le nom de rôle de votre choix.

```
aws iam create-role \
  --role-name ACKCapabilityRole \
  --assume-role-policy-document file://capability-trust-policy.json
```

c. Associez les politiques IAM requises au rôle. Pour ACK, associez des politiques aux AWS services que vous souhaitez gérer. Pour Argo CD, joignez des politiques pour Secrets Manager ou CodeConnections si nécessaire. Pour Kro, vous pouvez ignorer cette étape.

Exemple pour ACK avec autorisations S3 :

```
aws iam put-role-policy \
  --role-name ACKCapabilityRole \
  --policy-name S3Management \
  --policy-document file://s3-policy.json
```

Résolution des problèmes liés aux rôles liés aux capacités

La création de capacités échoue avec un « rôle IAM non valide »

Vérifiez que :

- Le rôle existe dans le même compte que le cluster
- La politique de confiance correspond à la politique de confiance des capacités

- Vous êtes `iam:PassRole` autorisé à jouer le rôle

La fonctionnalité affiche les erreurs d'autorisation

Vérifiez que :

- Le rôle dispose des autorisations IAM nécessaires pour le type de fonctionnalité
- L'entrée d'accès existe sur le cluster pour le rôle
- Des autorisations Kubernetes supplémentaires sont configurées si nécessaire (voir) the section called “Autorisations Kubernetes supplémentaires”

Les ressources ACK échouent avec des erreurs « autorisation refusée »

Vérifiez que :

- Le rôle dispose des autorisations IAM nécessaires pour votre cas d'utilisation
- Pour les contrôleurs ACK qui font référence à des secrets, assurez-vous d'avoir associé la politique d'entrée d'`AmazonEKSSecretReaderPolicy`accès aux espaces de noms appropriés.

Pour plus de conseils de résolution des problèmes, consultezthe section called “Considérations relatives aux fonctionnalités d'EKS”.

Surveillez les performances de votre cluster et consultez les journaux

Vous pouvez observer vos données dans Amazon EKS à l'aide de nombreux outils de surveillance ou de journalisation disponibles. Les données de vos journaux Amazon EKS peuvent être transmises à AWS des services ou à des outils partenaires pour l'analyse des données. De nombreux services sont disponibles dans le document AWS Management Console qui fournissent des données pour résoudre vos problèmes avec Amazon EKS. Vous pouvez également utiliser une solution open source AWS prise en charge pour surveiller l'infrastructure Amazon EKS.

Après avoir sélectionné Clusters dans le volet de navigation gauche de la console Amazon EKS, vous pouvez afficher l'état et les détails du cluster en choisissant le nom de votre cluster, puis l'onglet Observabilité. Pour afficher des détails sur les ressources Kubernetes existantes déployées sur votre cluster, consultez the section called "Accéder aux ressources du cluster".

La surveillance joue un rôle important dans le maintien de la fiabilité, de la disponibilité et des performances d'Amazon EKS et de vos AWS solutions. Nous vous recommandons de collecter des données de surveillance à partir de toutes les parties de votre AWS solution. De cette façon, vous pouvez déboguer plus facilement une éventuelle défaillance à plusieurs points. Avant de commencer à surveiller Amazon EKS, assurez-vous que votre plan de surveillance réponde aux questions suivantes.

- Quels sont vos objectifs ? Avez-vous besoin de notifications en temps réel si vos clusters évoluent considérablement ?
- Quelles ressources faut-il observer ?
- À quelle fréquence devez-vous observer ces ressources ? Votre entreprise veut-elle réagir rapidement aux risques ?
- Quels outils avez-vous l'intention d'utiliser ? Si vous exécutez déjà AWS Fargate lors de votre lancement, vous pouvez utiliser le routeur de log intégré.
- À qui souhaitez-vous confier les tâches de surveillance ?
- À qui voulez-vous que les notifications soient envoyées en cas de problème ?

Surveillance et journalisation sur Amazon EKS

Amazon EKS fournit des outils intégrés pour la surveillance et la journalisation. Pour les versions prises en charge, le tableau de bord d'observabilité offre une visibilité sur les performances de votre cluster. Il vous aide à détecter, dépanner et résoudre rapidement les problèmes. En plus des fonctionnalités de surveillance, il comprend des listes basées sur les journaux d'audit du plan de contrôle. Le plan de contrôle Kubernetes expose un certain nombre de métriques qui peuvent également être récupérées en dehors de la console.

La journalisation du plan de contrôle enregistre tous les appels d'API vers vos clusters, les informations d'audit capturant quels utilisateurs ont effectué quelles actions sur vos clusters, et les informations basées sur les rôles. Pour plus d'informations, consultez la section Journalisation et surveillance sur Amazon EKS dans le guide AWS prescriptif.

La journalisation du plan de contrôle Amazon EKS fournit des journaux d'audit et de diagnostic directement depuis le plan de contrôle Amazon EKS vers CloudWatch les journaux de votre compte. Ces journaux vous permettent de sécuriser et d'exécuter facilement vos clusters. Vous pouvez sélectionner les types de journaux exacts dont vous avez besoin, et les journaux sont envoyés sous forme de flux de journaux à un groupe pour chaque cluster Amazon EKS inclus CloudWatch. Pour de plus amples informations, veuillez consulter the section called "journaux de plan de contrôle ".

> **Note**
>
> Lorsque vous consultez les journaux de l'authentificateur Amazon EKS sur Amazon CloudWatch, les entrées affichées contiennent du texte similaire à l'exemple de texte suivant.
>
> ```
> level=info msg="mapping IAM role" groups="[]" role="arn:aws:
>  iam::111122223333:role/XXXXXXXXXXXXXXXXX-NodeManagerRole-XXXXXXXX"
>  username="eks:node-manager"
> ```
>
> Des entrées contenant ce texte sont attendues. `username` est un rôle de service interne Amazon EKS qui effectue des opérations spécifiques pour les groupes de nœuds gérés et Fargate.
> Pour une journalisation personnalisable de bas niveau, la journalisation Kubernetes est disponible.

Amazon EKS est intégré à AWS CloudTrail un service qui fournit un enregistrement des actions entreprises par un utilisateur, un rôle ou un AWS service dans Amazon EKS. CloudTrail capture tous les appels d'API pour Amazon EKS sous forme d'événements. Ces captures incluent les appels de la console Amazon EKS et les appels de code vers les opérations d'API Amazon EKS. Pour de plus amples informations, veuillez consulter the section called " AWS CloudTrail".

Le serveur d'API Kubernetes expose un certain nombre de métriques qui sont utiles pour la surveillance et l'analyse. Pour de plus amples informations, veuillez consulter the section called "Métriques Prometheus".

Pour configurer Fluent Bit pour les CloudWatch journaux Amazon personnalisés, consultez la section Configuration de Fluent Bit dans le guide de CloudWatch l'utilisateur Amazon.

Outils de surveillance et de journalisation Amazon EKS

Amazon Web Services fournit différents outils que vous pouvez utiliser pour surveiller Amazon EKS. Vous pouvez configurer certains outils pour configurer la surveillance automatique, mais certains nécessitent des appels manuels. Nous vous recommandons d'automatiser les tâches de surveillance autant que votre environnement et le jeu d'outils existant le permettent.

Le tableau suivant décrit les différentes options disponibles en matière d'outils de surveillance.

Zones	Outil	Description	Configuration
Plan de contrôle	Tableau de bord d'observabilité	Pour les versions prises en charge, le tableau de bord d'observabilité offre une visibilité sur les performances de votre cluster. Il vous aide à détecter, dépanner et résoudre rapidement les problèmes.	Procédures de configuration
Applications/plan de contrôle	Prometheus	Prometheus peut être utilisé pour surveille	Procédures de configuration

Zones	Outil	Description	Configuration
		r les métriques et les alertes relatives aux applications et au plan de contrôle.	
Applications	CloudWatch Container Insights	CloudWatch Container Insights collecte, agrège et résume les métriques et les journaux de vos applications conteneurisées et de vos microservices.	Procédures de configuration
Applications	AWS Distribution pour OpenTelemetry (ADOT)	ADOT peut collecter et envoyer des métriques corrélées, des données de suivi et des métadonné es aux services de AWS surveillance ou aux partenaires. Il peut être configuré via CloudWatch Container Insights.	Procédures de configuration
Applications	Amazon DevOps Guru	Amazon DevOps Guru détecte les performances opérationnelles et la disponibilité au niveau des nœuds.	Procédures de configuration

Zones	Outil	Description	Configuration
Applications	AWS X-Ray	AWS X-Ray reçoit des données de suivi concernant votre application. Ces données de suivi incluent les demandes entrantes et sortantes ainsi que les métadonné es relatives aux demandes. Pour Amazon EKS, la mise en œuvre nécessite le OpenTelemetry module complémen taire.	Procédures de configuration
Applications	Amazon CloudWatch	CloudWatch fournit gratuitement certaines métriques de base d'Amazon EKS sur les versions prises en charge. Vous pouvez étendre cette fonctionnalité avec l'opérateur CloudWatc h d'observabilité pour gérer la collecte des métriques, des journaux et des données de suivi.	Procédures de configuration

Le tableau suivant décrit les différentes options des outils de journalisation.

Zones	Outil	Description	Configuration
Plan de contrôle	Tableau de bord d'observabilité	Pour les versions prises en charge, le tableau de bord d'observabilité affiche des listes basées sur les journaux d'audit du plan de contrôle. Il inclut également des liens vers les journaux du plan de contrôle sur Amazon CloudWatch.	Procédures de configuration
Applications	Informations sur les CloudWatch conteneurs Amazon	Amazon CloudWatch Container Insights collecte, agrège et résume les métriques et les journaux provenant de vos applications conteneurisées et de vos microservices.	Procédures de configuration
Plan de contrôle	Amazon CloudWatch Logs	Vous pouvez envoyer des journaux d'audit et de diagnostic directement depuis le plan de contrôle Amazon EKS vers CloudWatch Logs in your account.	Procédures de configuration
Plan de contrôle	AWS CloudTrail	Il consigne les appels d'API par un utilisate	Procédures de configuration

Zones	Outil	Description	Configuration
		ur, un rôle ou un service.	
Plusieurs zones pour les instances AWS Fargate	AWS Routeur Fargate Log	Pour les instances AWS Fargate, le routeur de journaux transmet les journaux aux services ou aux AWS outils partenair es. Il utilise AWS pour Fluent Bit. Les journaux peuvent être diffusés vers d'autres AWS services ou outils partenaires.	Procédures de configuration

Surveillez votre cluster à l'aide du tableau de bord d'observabilité

La console Amazon EKS comprend un tableau de bord d'observabilité qui offre une visibilité sur les performances de votre cluster. Les informations qu'il fournit vous aident à détecter, dépanner et résoudre rapidement les problèmes. Vous pouvez ouvrir la section applicable du tableau de bord d'observabilité en sélectionnant un élément dans le Résumé de l'état et des performances. Ce résumé est inclus à plusieurs endroits, notamment dans l'onglet Observabilité.

Le tableau de bord d'observabilité est divisé en plusieurs onglets.

Résumé

Le Résumé de l'état de santé et des performances répertorie la quantité d'articles dans différentes catégories. Chaque numéro agit comme un lien hypertexte vers un emplacement du tableau de bord d'observabilité contenant une liste pour cette catégorie.

État du cluster

L'état du cluster fournit des notifications importantes à prendre en compte, dont certaines peuvent nécessiter une action immédiate de votre part. Cette liste vous permet de consulter les descriptions

et les ressources concernées. L'état du cluster comprend deux tableaux : Problèmes d'état et Informations sur la configuration. Pour actualiser le statut des Problèmes d'état, cliquez sur le bouton d'actualisation (↻). Les Informations sur la configuration sont mises à jour automatiquement toutes les 24 h et ne peuvent pas être actualisées manuellement.

Pour plus d'informations sur les Problèmes d'état, consultez the section called "Foire aux questions sur l'intégrité du cluster et codes d'erreur avec chemins de résolution". Pour plus d'informations sur les Informations sur la configuration, consultez the section called "Informations sur le cluster".

Surveillance du plan de contrôle

L'onglet Surveillance du plan de contrôle est divisé en trois sections, chacune vous aidant à surveiller et à dépanner le plan de contrôle de votre cluster.

Métriques

Pour les clusters Kubernetes version `1.28` et supérieure, la section Métriques affiche des graphiques de plusieurs métriques recueillies pour divers composants du plan de contrôle.

Vous pouvez définir la période utilisée par l'axe X de chaque graphique en effectuant des sélections en haut de la section. Vous pouvez actualiser les données à l'aide du bouton d'actualisation (↻). Pour chaque graphique distinct, le bouton représentant des ellipses verticales (1) ouvre un menu contenant les options de CloudWatch.

Ces métriques et bien d'autres sont automatiquement disponibles CloudWatch sous forme de métriques de surveillance de base dans l'espace de `AWS/EKS` noms. Pour plus d'informations, consultez les sections Surveillance de base et surveillance détaillée dans le guide de CloudWatch l'utilisateur Amazon. Pour obtenir des statistiques, des visualisations et des informations plus détaillées, consultez Container Insights dans le guide de CloudWatch l'utilisateur Amazon. Si vous préférez la surveillance basée sur Prometheus, consultez the section called "Métriques Prometheus".

Le tableau suivant décrit les métriques disponibles.

Métrique	Description
APIServer Demandes	Nombre de demandes par minute effectuées au serveur API.

Métrique	Description
APIServer Nombre total de demandes 4XX	Nombre de demandes au serveur API par minute ayant reçu des codes de réponse HTTP 4XX (erreurs côté client).
APIServer Nombre total de demandes : 5XX	Nombre de demandes au serveur API par minute ayant reçu des codes de réponse HTTP 5XX (erreurs côté serveur).
APIServer Nombre total de demandes 429	Nombre de demandes au serveur API par minute ayant reçu des codes de réponse HTTP 429 (trop de demandes).
Taille de stockage	Taille de la base de données de stockage (`etcd`).
Tentatives du planificateur	Nombre de tentatives de planification de pods par résultats « non planifiable », « erreur » et « planifié ».
Pods en attente	Nombre de pods en attente par type de file d'attente « actif », « reprise », « non planifiable » et « protégé ».
Latence des demandes du serveur API	Latence des demandes du serveur API.
Demandes en cours du serveur API	Demandes en cours du serveur API.
Demandes Webhook	Nombre de demandes Webhook par minute.
Rejets de demandes Webhook	Nombre de demandes Webhook rejetées.
Latence des demandes Webhook P99	Latence au 99e centile des demandes Webhook externes provenant de tiers.

CloudWatch Informations sur les journaux

La section CloudWatch Log Insights affiche différentes listes basées sur les journaux d'audit du plan de contrôle. Les journaux du plan de contrôle Amazon EKS doivent être activés pour utiliser cette fonctionnalité, ce que vous pouvez faire dans la section Afficher les journaux du plan de contrôle dans la CloudWatch section Afficher les journaux du plan de contrôle.

Une fois que suffisamment de temps s'est écoulé pour collecter les données, vous pouvez Exécuter toutes les requêtes ou choisir Exécuter la requête pour une seule liste à la fois. Un coût supplémentaire sera facturé CloudWatch chaque fois que vous lancerez des requêtes. Sélectionnez la période pour laquelle vous voulez afficher les résultats en haut de la section. Si vous souhaitez un contrôle plus avancé pour n'importe quelle requête, vous pouvez choisir Afficher dans CloudWatch. Cela vous permettra de mettre à jour une requête CloudWatch en fonction de vos besoins.

Pour plus d'informations, consultez Analyser les données des CloudWatch journaux avec Logs Insights dans le guide de l'utilisateur Amazon CloudWatch Logs.

Afficher les connexions au plan de contrôle CloudWatch

Sélectionnez Gérer la journalisation pour mettre à jour les types de journaux disponibles. Plusieurs minutes sont nécessaires pour que les journaux apparaissent dans CloudWatch les journaux une fois que vous avez activé la journalisation. Une fois ce délai écoulé, sélectionnez l'un des liens Afficher de cette section pour accéder au journal correspondant.

Pour de plus amples informations, veuillez consulter the section called "journaux de plan de contrôle ".

Informations sur le cluster

Le tableau Informations sur la mise à niveau met en évidence les problèmes et recommande des mesures correctives, ce qui accélère le processus de validation pour mettre à niveau vers les nouvelles versions de Kubernetes. Amazon EKS analyse automatiquement les clusters par rapport à une liste de problèmes potentiels liés à la mise à niveau de la version de Kubernetes. Le tableau Informations sur la mise à niveau répertorie les vérifications effectuées par Amazon EKS sur ce cluster, ainsi que leur statut associé.

Amazon EKS tient à jour et actualise périodiquement la liste des vérifications d'informations à effectuer en fonction des évaluations des modifications apportées au projet Kubernetes ainsi que des modifications du service Amazon EKS liées aux nouvelles versions. La console Amazon EKS

actualise automatiquement le statut de chaque information, qui peut être consulté dans la colonne « Dernière actualisation ».

Pour de plus amples informations, veuillez consulter the section called "Informations sur le cluster".

Problèmes d'état du nœud

L'agent de surveillance des nœuds Amazon EKS lit automatiquement les journaux des nœuds pour détecter les problèmes d'état. Quel que soit le paramètre de réparation automatique, tous les problèmes d'état des nœuds sont signalés afin que vous puissiez les examiner si nécessaire. Si un type de problème est répertorié mais pas décrit, vous pouvez lire la description dans son élément de fenêtre contextuelle.

Lorsque vous actualisez la page, tous les problèmes résolus disparaissent de la liste. Si la réparation automatique est activée, vous pouvez rencontrer temporairement des problèmes d'état qui seront résolus sans action de votre part. Les problèmes qui ne sont pas pris en charge par la réparation automatique peuvent nécessiter une action manuelle de votre part, selon leur nature.

Pour que les problèmes d'état des nœuds soient signalés, votre cluster doit utiliser le mode automatique Amazon EKS ou disposer du module complémentaire d'agent de surveillance des nœuds. Pour de plus amples informations, veuillez consulter the section called "État du nœud".

Fonctionnalités d'EKS

La section Capabilities indique l'état et l'état de santé de vos ressources EKS Capability dans le cluster. Les notifications relatives à l'état et à l'état des deux fonctionnalités et de leurs ressources Kubernetes gérées dans votre cluster peuvent être surveillées ici. Lorsque vous actualisez la page, tous les problèmes résolus disparaissent de la liste.

Pour de plus amples informations, veuillez consulter the section called "Travailler avec les capacités".

Surveillez le trafic de charge de travail de Kubernetes grâce à Container Network Observability

Amazon EKS fournit des fonctionnalités d'observabilité réseau améliorées qui fournissent des informations plus approfondies sur votre environnement réseau de conteneurs. Ces fonctionnalités vous aident à mieux comprendre, surveiller et dépanner votre environnement réseau Kubernetes

dans. AWS Grâce à l'observabilité améliorée du réseau de conteneurs, vous pouvez tirer parti de mesures granulaires liées au réseau pour une meilleure détection proactive des anomalies dans le trafic des clusters, les flux inter-AZ et les services. AWS À l'aide de ces indicateurs, vous pouvez mesurer les performances du système et visualiser les indicateurs sous-jacents à l'aide de votre stack d'observabilité préféré.

En outre, Amazon EKS fournit désormais des visualisations de surveillance du réseau dans la AWS console qui accélèrent et améliorent le dépannage précis pour une analyse plus rapide des causes premières. Vous pouvez également tirer parti de ces fonctionnalités visuelles pour identifier les principaux intervenants et les flux réseau à l'origine des retransmissions et des délais de retransmission, éliminant ainsi les angles morts lors d'incidents.

Ces fonctionnalités sont activées par Amazon CloudWatch Network Flow Monitor.

Cas d'utilisation

Mesurez les performances du réseau pour détecter les anomalies

Plusieurs équipes s'appuient sur une pile d'observabilité qui leur permet de mesurer les performances de leur système, de visualiser les indicateurs du système et de s'alarmer en cas de dépassement d'un seuil spécifique. L'observabilité du réseau de conteneurs dans EKS va dans ce sens en exposant les indicateurs clés du système que vous pouvez extraire pour élargir l'observabilité des performances du réseau de votre système au niveau du pod et du nœud de travail.

Tirez parti des visualisations de console pour un dépannage plus précis

En cas d'alarme provenant de votre système de surveillance, vous souhaiterez peut-être vous concentrer sur le cluster et la charge de travail à l'origine du problème. À cette fin, vous pouvez tirer parti des visualisations de la console EKS qui réduisent le champ d'investigation au niveau du cluster et accélèrent la divulgation des flux réseau responsables du plus grand nombre de retransmissions, des délais de retransmission et du volume de données transférées.

Suivez les personnes qui parlent le plus dans votre environnement Amazon EKS

De nombreuses équipes utilisent EKS comme base de leurs plateformes, ce qui en fait le point central de l'activité réseau d'un environnement applicatif. À l'aide des fonctionnalités de surveillance du réseau de cette fonctionnalité, vous pouvez déterminer quelles charges de travail sont responsables de la majeure partie du trafic (mesuré par le volume de données) au sein du cluster, à

travers celui-ci AZs, ainsi que du trafic vers des destinations externes au sein AWS (DynamoDB et S3) et au-delà AWS du cloud (Internet ou sur site). En outre, vous pouvez surveiller les performances de chacun de ces flux en fonction des retransmissions, des délais de retransmission et des données transférées.

Caractéristiques

1. Indicateurs de performance - Cette fonctionnalité vous permet de récupérer les indicateurs système liés au réseau pour les pods et les nœuds de travail directement à partir de l'agent Network Flow Monitor (NFM) exécuté dans votre cluster EKS.
2. Carte des services : cette fonctionnalité permet de visualiser de manière dynamique l'intercommunication entre les charges de travail du cluster, ce qui vous permet de divulguer rapidement les indicateurs clés (retransmissions - RT, délais de retransmission - RTO et données transférées - DT) associés aux flux réseau entre les modules communicants.
3. Tableau de flux : avec ce tableau, vous pouvez surveiller les principaux intervenants des charges de travail Kubernetes de votre cluster sous trois angles différents : vue du AWS service, vue du cluster et vue externe. Pour chaque vue, vous pouvez voir les retransmissions, les délais de retransmission et les données transférées entre le module source et sa destination.
   - AWS vue des services : affiche les principaux interlocuteurs des AWS services (DynamoDB et S3)
   - Vue du cluster : affiche les principaux orateurs du cluster (est ← vers → ouest)
   - Vue externe : affiche les meilleurs orateurs vers des destinations externes au cluster situées à l'extérieur AWS

Mise en route

Pour commencer, activez Container Network Observability dans la console EKS pour un cluster nouveau ou existant. Cela automatisera la création des dépendances du Network Flow Monitor (NFM) (ressources Scope et Monitor). En outre, vous devrez installer le module complémentaire Network Flow Monitor Agent. Vous pouvez également installer ces dépendances à l'aide d'EKS APIs (pour le module complémentaire), de NFM APIs ou d'Infrastructure as Code (comme Terraform). `AWS CLI` Une fois ces dépendances établies, vous pouvez configurer votre outil de surveillance préféré pour extraire de l'agent NFM les indicateurs de performance réseau pour les pods et les nœuds de travail. Pour visualiser l'activité réseau et les performances de vos charges de travail, vous pouvez accéder à la console EKS sous l'onglet « Réseau » du tableau de bord d'observabilité du cluster.

Lorsque vous utilisez Network Flow Monitor dans EKS, vous pouvez conserver votre flux de travail d'observabilité et votre infrastructure technologique existants tout en tirant parti d'un ensemble de fonctionnalités supplémentaires qui vous permettent de mieux comprendre et d'optimiser la couche réseau de votre environnement EKS. Pour en savoir plus sur la tarification du Network Flow Monitor, cliquez ici.

Prérequis et remarques importantes

1. Comme indiqué ci-dessus, si vous activez Container Network Observability depuis la console EKS, les dépendances des ressources NFM sous-jacentes (Scope et Monitor) seront automatiquement créées en votre nom, et vous serez guidé tout au long du processus d'installation du module complémentaire EKS pour NFM.
2. Si vous souhaitez activer cette fonctionnalité à l'aide de l'infrastructure en tant que code (iAc) comme Terraform, vous devrez définir les dépendances suivantes dans votre iAC : NFM Scope, NFM Monitor, module complémentaire EKS pour NFM. En outre, vous devrez accorder les autorisations nécessaires au module complémentaire EKS à l'aide de Pod Identity ou de rôles IAM pour les comptes de service (IRSA).
3. Vous devez exécuter une version minimale de 1.1.0 pour le module complémentaire EKS de l'agent NFM.
4. Vous devez utiliser la version 6.21.0 ou supérieure du AWS fournisseur Terraform pour prendre en charge les ressources du Network Flow Monitor.

Autorisations IAM requises

Module complémentaire EKS pour agent NFM

Vous pouvez utiliser la politique `CloudWatchNetworkFlowMonitorAgentPublishPolicy`AWS gérée avec Pod Identity. Cette politique contient des autorisations permettant à l'agent NFM d'envoyer des rapports de télémétrie (métriques) à un point de terminaison Network Flow Monitor.

```
{
  "Version" : "2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "networkflowmonitor:Publish"
      ],
```

```
      "Resource" : "*"
    }
  ]
}
```

Observabilité du réseau de conteneurs dans la console EKS

Les autorisations suivantes sont requises pour activer la fonctionnalité et visualiser la carte des services et le tableau des flux dans la console.

```
{
  "Version" : "2012-10-17",
  "Statement" : [
    {
      "Effect": "Allow",
      "Action": [
        "networkflowmonitor:ListScopes",
        "networkflowmonitor:ListMonitors",
        "networkflowmonitor:GetScope",
        "networkflowmonitor:GetMonitor",
        "networkflowmonitor:CreateScope",
        "networkflowmonitor:CreateMonitor",
        "networkflowmonitor:TagResource",
        "networkflowmonitor:StartQueryMonitorTopContributors",
        "networkflowmonitor:StopQueryMonitorTopContributors",
        "networkflowmonitor:GetQueryStatusMonitorTopContributors",
        "networkflowmonitor:GetQueryResultsMonitorTopContributors"
      ],
      "Resource": "*"
    }
  ]
}
```

Utilisation de la AWS CLI, de l'API EKS et de l'API NFM

```
#!/bin/bash

# Script to create required Network Flow Monitor resources
set -e

CLUSTER_NAME="my-eks-cluster"
CLUSTER_ARN="arn:aws:eks:{Region}:{Account}:cluster/{ClusterName}"
```

```
REGION="us-west-2"
AGENT_NAMESPACE="amazon-network-flow-monitor"

echo "Creating Network Flow Monitor resources..."

# Check if Network Flow Monitor agent is running in the cluster
echo "Checking for Network Flow Monitor agent in cluster..."
if kubectl get pods -n "$AGENT_NAMESPACE" --no-headers 2>/dev/null | grep -q "Running";
 then
    echo "Network Flow Monitor agent exists and is running in the cluster"
else
    echo "Network Flow Monitor agent not found. Installing as EKS addon..."
    aws eks create-addon \
        --cluster-name "$CLUSTER_NAME" \
        --addon-name "$AGENT_NAMESPACE" \
        --region "$REGION"
    echo "Network Flow Monitor addon installation initiated"
fi

# Get Account ID
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)

echo "Cluster ARN: $CLUSTER_ARN"
echo "Account ID: $ACCOUNT_ID"

# Check for existing scope
echo "Checking for existing Network Flow Monitor Scope..."
EXISTING_SCOPE=$(aws networkflowmonitor list-scopes --region $REGION --query
 'scopes[0].scopeArn' --output text 2>/dev/null || echo "None")

if [ "$EXISTING_SCOPE" != "None" ] && [ "$EXISTING_SCOPE" != "null" ]; then
    echo "Using existing scope: $EXISTING_SCOPE"
    SCOPE_ARN=$EXISTING_SCOPE
else
    echo "Creating new Network Flow Monitor Scope..."
    SCOPE_RESPONSE=$(aws networkflowmonitor create-scope \
        --targets "[{\"targetIdentifier\":{\"targetId\":{\"accountId\":
\"${ACCOUNT_ID}\"},\"targetType\":\"ACCOUNT\"},\"region\":\"${REGION}\"}]" \
        --region $REGION \
        --output json)

    SCOPE_ARN=$(echo $SCOPE_RESPONSE | jq -r '.scopeArn')
    echo "Scope created: $SCOPE_ARN"
fi
```

```
# Create Network Flow Monitor with EKS Cluster as local resource
echo "Creating Network Flow Monitor..."
MONITOR_RESPONSE=$(aws networkflowmonitor create-monitor \
    --monitor-name "${CLUSTER_NAME}-monitor" \
    --local-resources "type=AWS::EKS::Cluster,identifier=${CLUSTER_ARN}" \
    --scope-arn "$SCOPE_ARN" \
    --region $REGION \
    --output json)

MONITOR_ARN=$(echo $MONITOR_RESPONSE | jq -r '.monitorArn')

echo "Monitor created: $MONITOR_ARN"

echo "Network Flow Monitor setup complete!"
echo "Monitor ARN: $MONITOR_ARN"
echo "Scope ARN: $SCOPE_ARN"
echo "Local Resource: AWS::EKS::Cluster (${CLUSTER_ARN})"
```

Utilisation de l'infrastructure en tant que code (IaC)

Terraform

Si vous utilisez Terraform pour gérer votre infrastructure AWS cloud, vous pouvez inclure les configurations de ressources suivantes pour activer l'observabilité du réseau de conteneurs pour votre cluster.

Champ d'application NFM

```
data "aws_caller_identity" "current" {}

resource "aws_networkflowmonitor_scope" "example" {
  target {
    region = "us-east-1"
    target_identifier {
      target_type = "ACCOUNT"
      target_id {
        account_id = data.aws_caller_identity.current.account_id
      }
    }
  }

  tags = {
```

```
    Name = "example"
  }
}
```

Moniteur NFM

```
resource "aws_networkflowmonitor_monitor" "example" {
  monitor_name = "eks-cluster-name-monitor"
  scope_arn    = aws_networkflowmonitor_scope.example.scope_arn

  local_resource {
    type       = "AWS::EKS::Cluster"
    identifier = aws_eks_cluster.example.arn
  }

  remote_resource {
    type       = "AWS::Region"
    identifier = "us-east-1" # this must be the same region that the cluster is in
  }

  tags = {
    Name = "example"
  }
}
```

Module complémentaire EKS pour NFM

```
resource "aws_eks_addon" "example" {
  cluster_name                = aws_eks_cluster.example.name
  addon_name                  = "aws-network-flow-monitoring-agent"
}
```

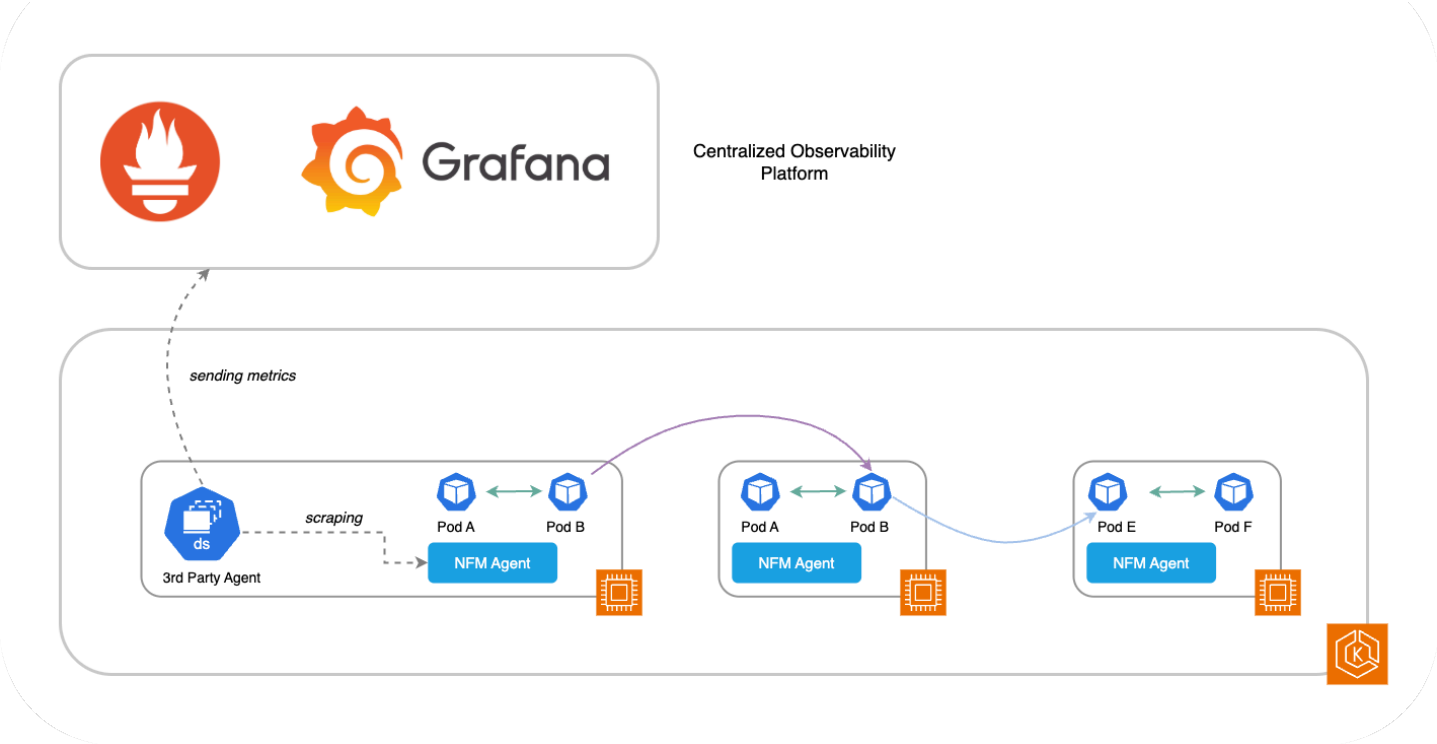
Fonctionnement

Métriques de performances

Métriques du système

Si vous utilisez des outils tiers (3P) pour surveiller votre environnement EKS (tels que Prometheus et Grafana), vous pouvez récupérer les métriques système prises en charge directement à partir de l'agent Network Flow Monitor. Ces mesures peuvent être envoyées à votre système de surveillance pour étendre la mesure des performances réseau de votre système au niveau du module et du nœud

de travail. Les métriques disponibles sont répertoriées dans le tableau, sous Métriques système prises en charge.



Pour activer ces métriques, remplacez les variables d'environnement suivantes à l'aide des variables de configuration lors du processus d'installation (voir : https://aws.amazon.com/blogs/ containers/ - advanced-configuration/amazon-eks-add-ons) :

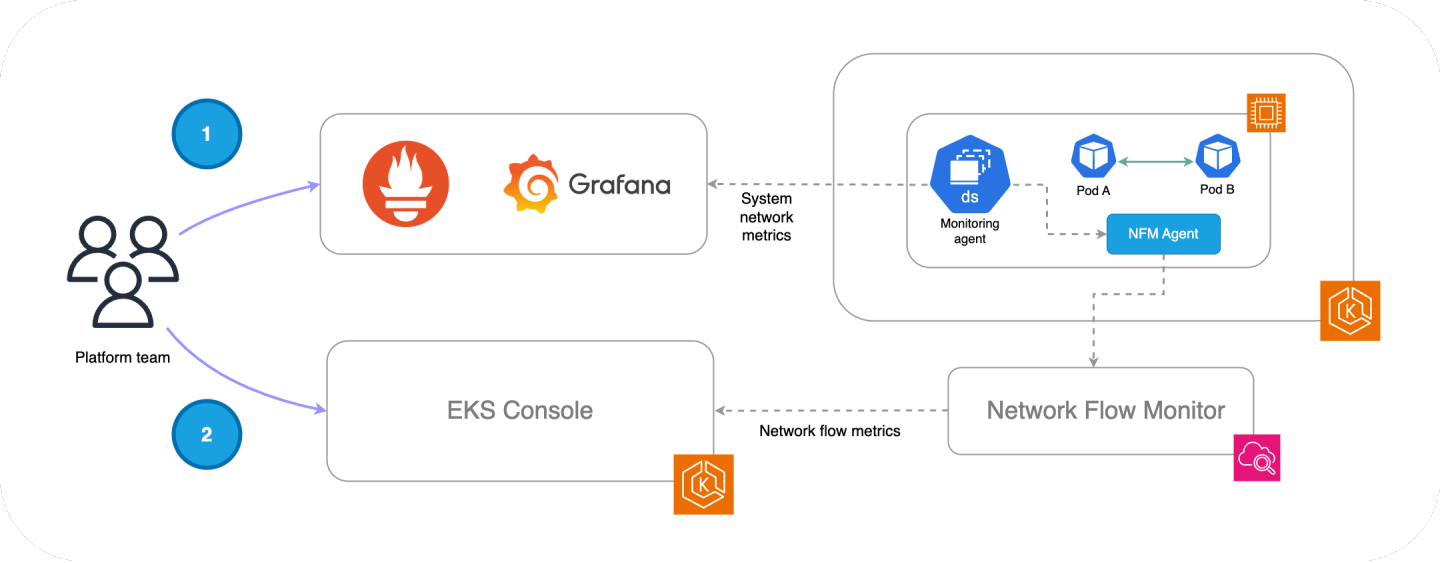
```
OPEN_METRICS:
    Enable or disable open metrics. Disabled if not supplied
    Type: String
    Values: [“on”, “off”]
OPEN_METRICS_ADDRESS:
    Listening IP address for open metrics endpoint. Defaults to 127.0.0.1 if not
 supplied
    Type: String
OPEN_METRICS_PORT:
    Listening port for open metrics endpoint. Defaults to 80 if not supplied
    Type: Integer
    Range: [0..65535]
```

Mesures du niveau de débit

En outre, Network Flow Monitor capture les données de flux réseau ainsi que les mesures de niveau de débit : retransmissions, délais de retransmission et données transférées. Ces données sont

traitées par Network Flow Monitor et visualisées dans la console EKS pour faire apparaître le trafic dans l'environnement de votre cluster et ses performances en fonction de ces mesures de niveau de flux.

Le schéma ci-dessous décrit un flux de travail dans lequel les deux types de métriques (système et niveau de flux) peuvent être exploités pour obtenir une meilleure intelligence opérationnelle.



1. L'équipe de la plateforme peut collecter et visualiser les métriques du système dans sa pile de surveillance. Une fois les alertes en place, ils peuvent détecter les anomalies du réseau ou les problèmes affectant les pods ou les nœuds de travail à l'aide des métriques du système fournies par l'agent NFM.
2. À l'étape suivante, les équipes chargées de la plateforme peuvent tirer parti des visualisations natives de la console EKS pour réduire davantage le champ d'investigation et accélérer le dépannage en fonction des représentations des flux et des indicateurs associés.

Remarque importante : l'extraction des métriques du système depuis l'agent NFM et le processus par lequel l'agent NFM envoie les métriques de niveau de flux vers le backend NFM sont des processus indépendants.

Métriques du système prises en charge

Remarque importante : les métriques du système sont exportées au OpenMetricsformat.

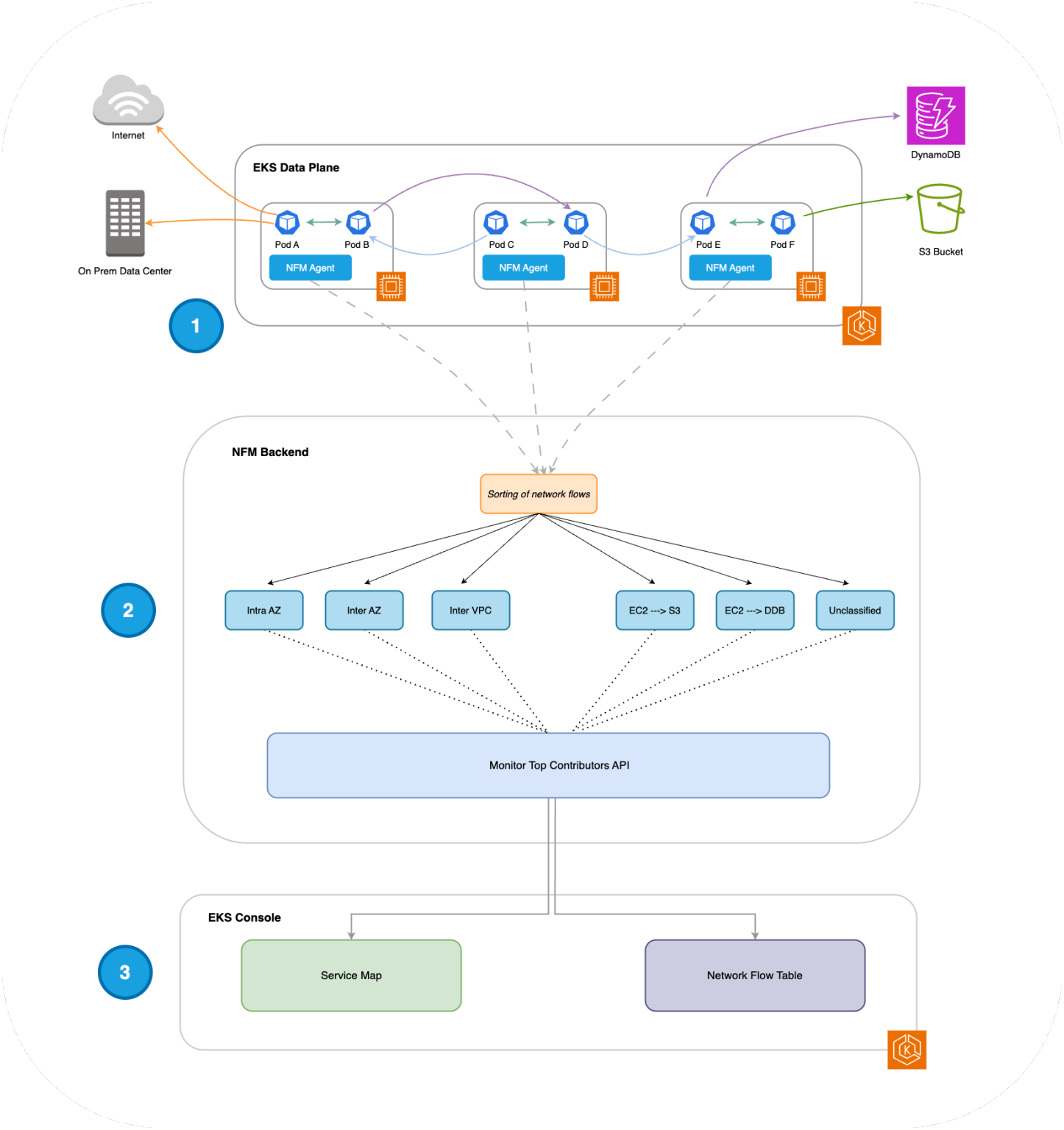
Nom de la métrique	Type	Dimensions	Description
flux d'entrée	Jauge	instance_id, iface, pod, espace de noms, nœud	Nombre de flux TCP d'entrée () TcpPassiveOpens
flux de sortie	Jauge	instance_id, iface, pod, espace de noms, nœud	Nombre de flux TCP de sortie () TcpActiveOpens
paquets d'entrée	Jauge	instance_id, iface, pod, espace de noms, nœud	Nombre de paquets entrants (delta)
paquets de sortie	Jauge	instance_id, iface, pod, espace de noms, nœud	Nombre de paquets de sortie (delta)
octets d'entrée	Jauge	instance_id, iface, pod, espace de noms, nœud	Nombre d'octets d'entrée (delta)
octets de sortie	Jauge	instance_id, iface, pod, espace de noms, nœud	Nombre d'octets de sortie (delta)
bw_in_allowance_exceeded	Jauge	instance_id, eni, nœud	Paquets queued/dropped dus à une limite de bande passante entrante
bw_out_allowance_exceeded	Jauge	instance_id, eni, nœud	Paquets queued/dropped dus à une limite de bande passante sortante
pps_allowance_dépassé	Jauge	instance_id, eni, nœud	Paquets queued/dropped dus à une

Nom de la métrique	Type	Dimensions	Description
			limite PPS bidirecti onnelle
conntrack_allowanc e_exceeded	Jauge	instance_id, eni, nœud	Paquets abandonné s en raison de la limite de suivi des connexions
linklocal_allowanc e_exceeded	Jauge	instance_id, eni, nœud	Paquets abandonnés en raison de la limite PPS du service proxy local

Mesures de niveau de débit prises en charge

Nom de la métrique	Type	Description
Retransmissions TCP	Compteur	Nombre de fois qu'un expéditeur renvoie un paquet perdu ou endommagé pendant la transmission.
Délais de retransmission TCP	Compteur	Nombre de fois qu'un expéditeur a entamé une période d'attente pour déterminer si un paquet avait été perdu pendant le transport.
Données (octets) transférées	Compteur	Volume de données transféré entre une source et une destination pour un flux donné.

Carte des services et tableau des flux

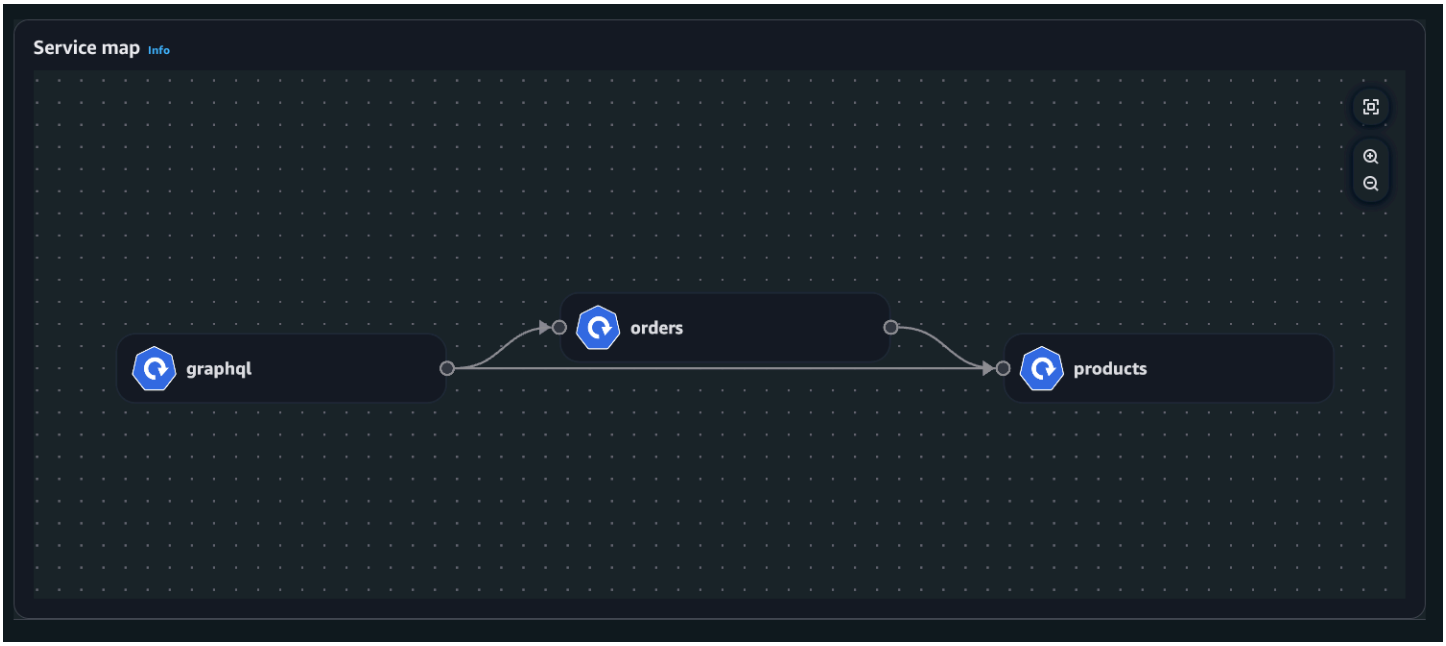


1. Une fois installé, l'agent Network Flow Monitor s'exécute DaemonSet sur chaque nœud de travail et collecte les 500 principaux flux réseau (en fonction du volume de données transférées) toutes les 30 secondes.
2. Ces flux réseau sont classés dans les catégories suivantes : Intra AZ, Inter AZ, EC2 → S3, EC2 → DynamoDB (DDB) et Non classé. Chaque flux est associé à 3 métriques : les retransmissions, les délais de retransmission et les données transférées (en octets).
   - Intra AZ : le réseau circule entre les pods d'une même zone
   - Inter AZ - le réseau circule entre les pods de différentes manières AZs
   - EC2 → S3 - flux réseau des pods vers S3
   - EC2 → DDB - flux réseau des pods vers DDB
   - Non classé : flux réseau des modules vers Internet ou sur site
3. Les flux réseau issus de l'API Top Contributors de Network Flow Monitor sont utilisés pour alimenter les expériences suivantes dans la console EKS :
   - Carte des services : visualisation des flux réseau au sein du cluster (Intra AZ et Inter AZ).
   - Tableau des flux : tableau présentant les flux réseau au sein du cluster (Intra AZ et Inter AZ), des pods aux AWS services (EC2 → S3 et EC2 → DDB), et des pods vers les destinations externes (non classés).

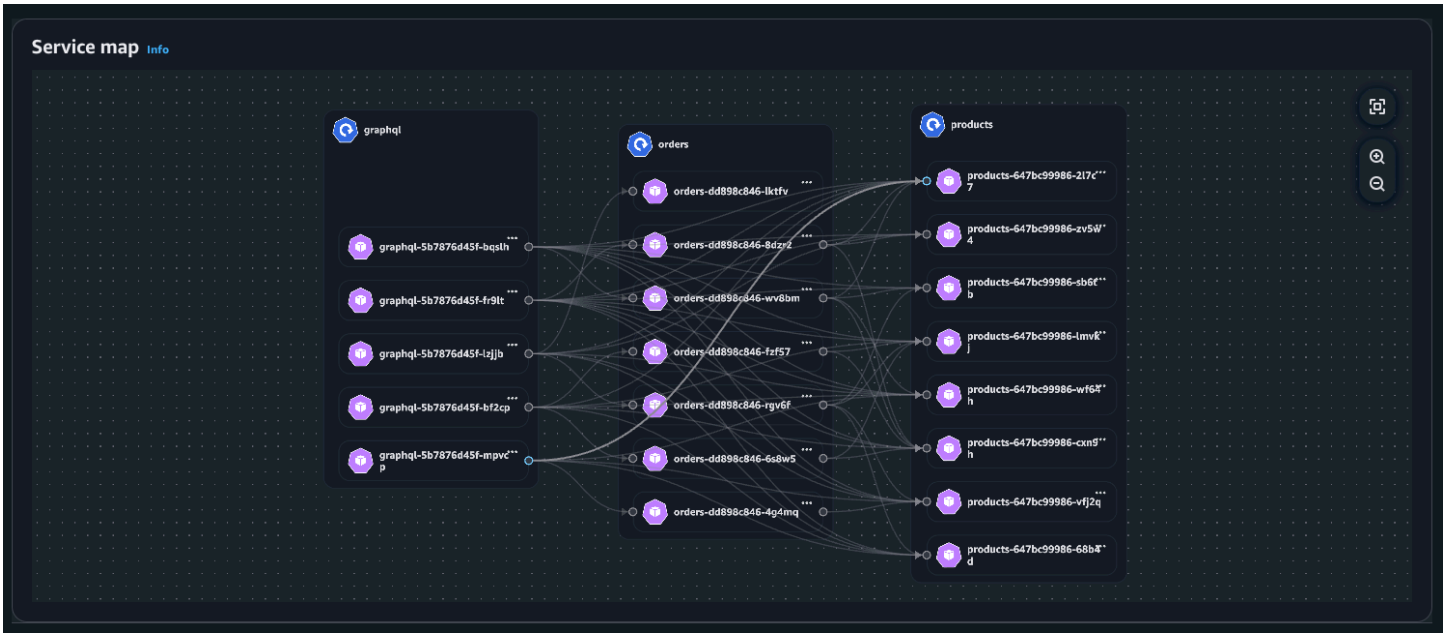
Les flux réseau extraits de l'API Top Contributors sont limités à une heure et peuvent inclure jusqu'à 500 flux de chaque catégorie. Pour la carte des services, cela signifie que jusqu'à 1 000 flux peuvent être sélectionnés et présentés à partir des catégories de flux Intra AZ et Inter AZ sur une plage horaire d'une heure. Pour le tableau des flux, cela signifie que jusqu'à 3 000 flux réseau peuvent être obtenus et présentés à partir des 6 catégories de flux réseau sur une période de 2 heures.

Exemple : carte des services

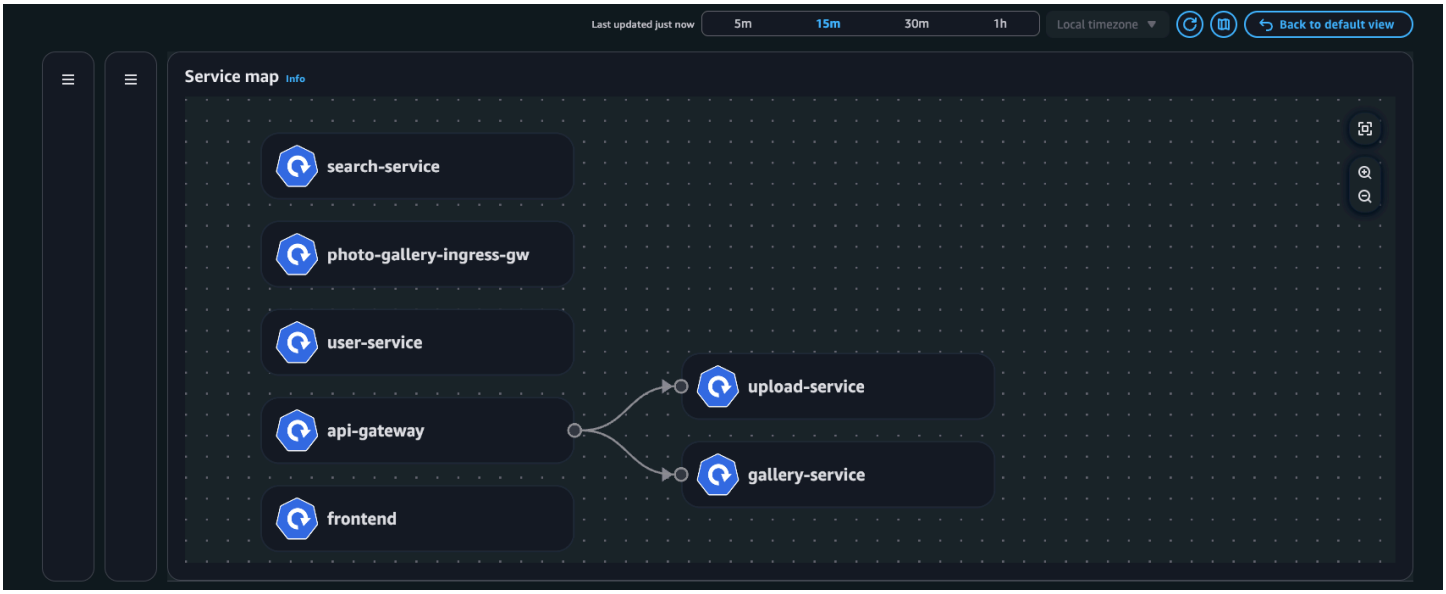
Vue du déploiement



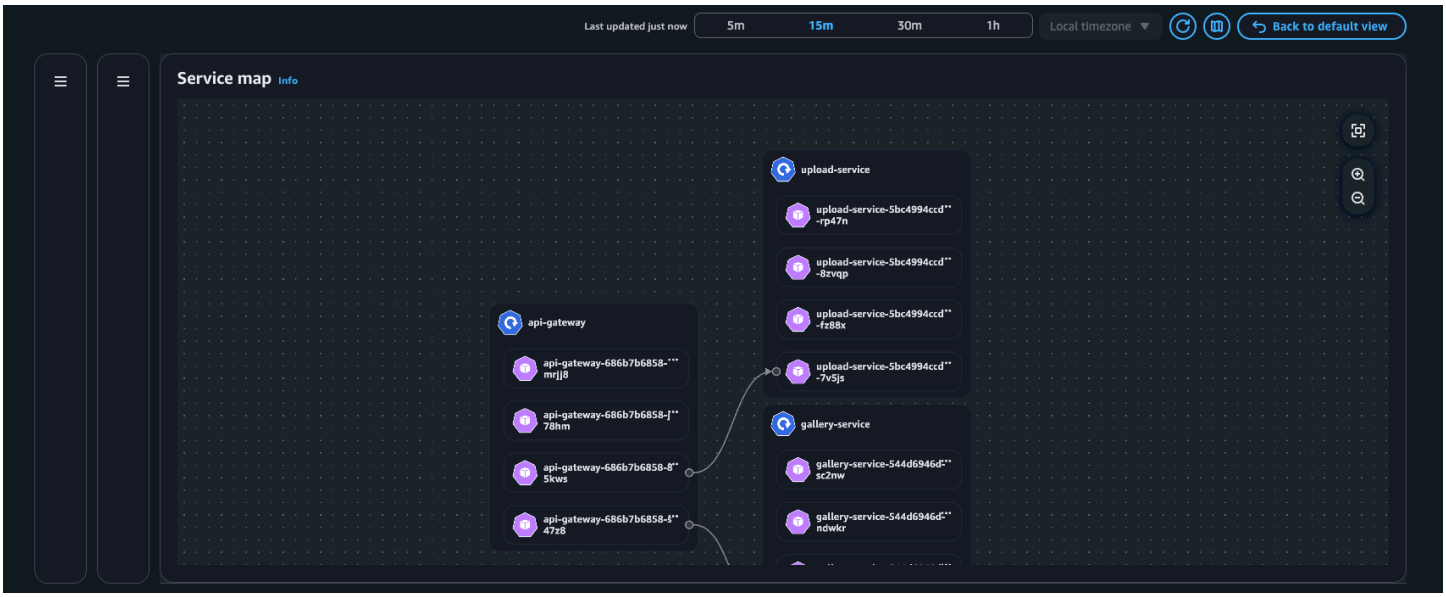
Vue du pod



Vue du déploiement

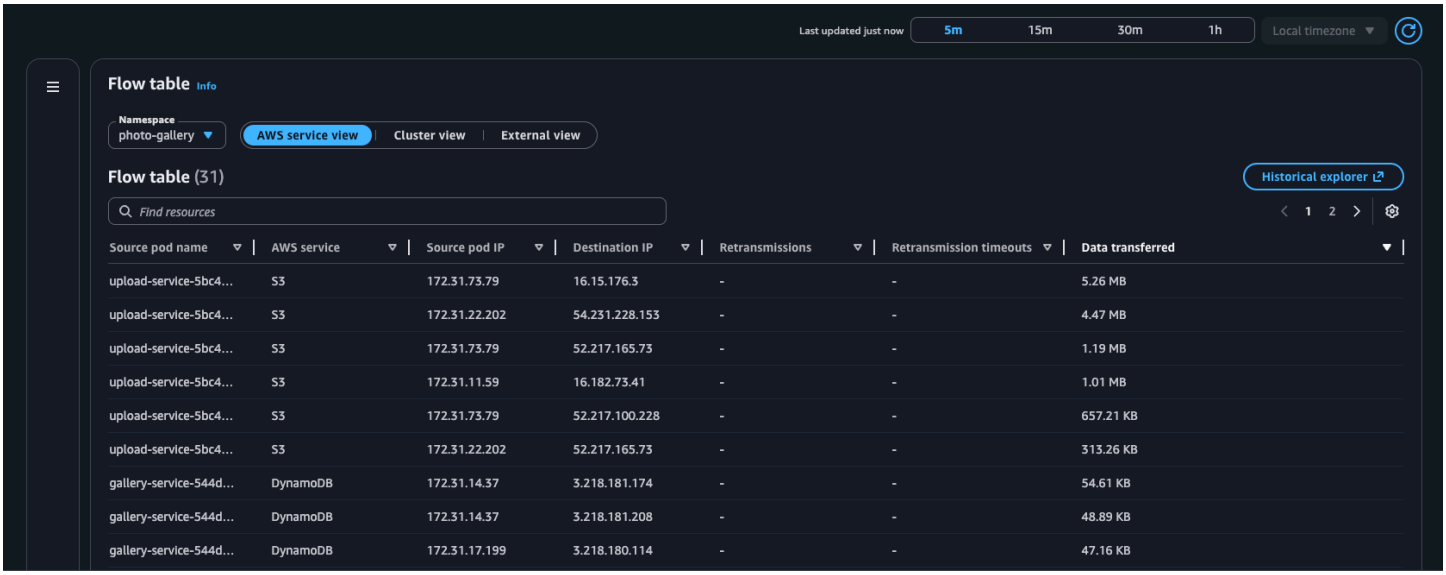


Vue du pod

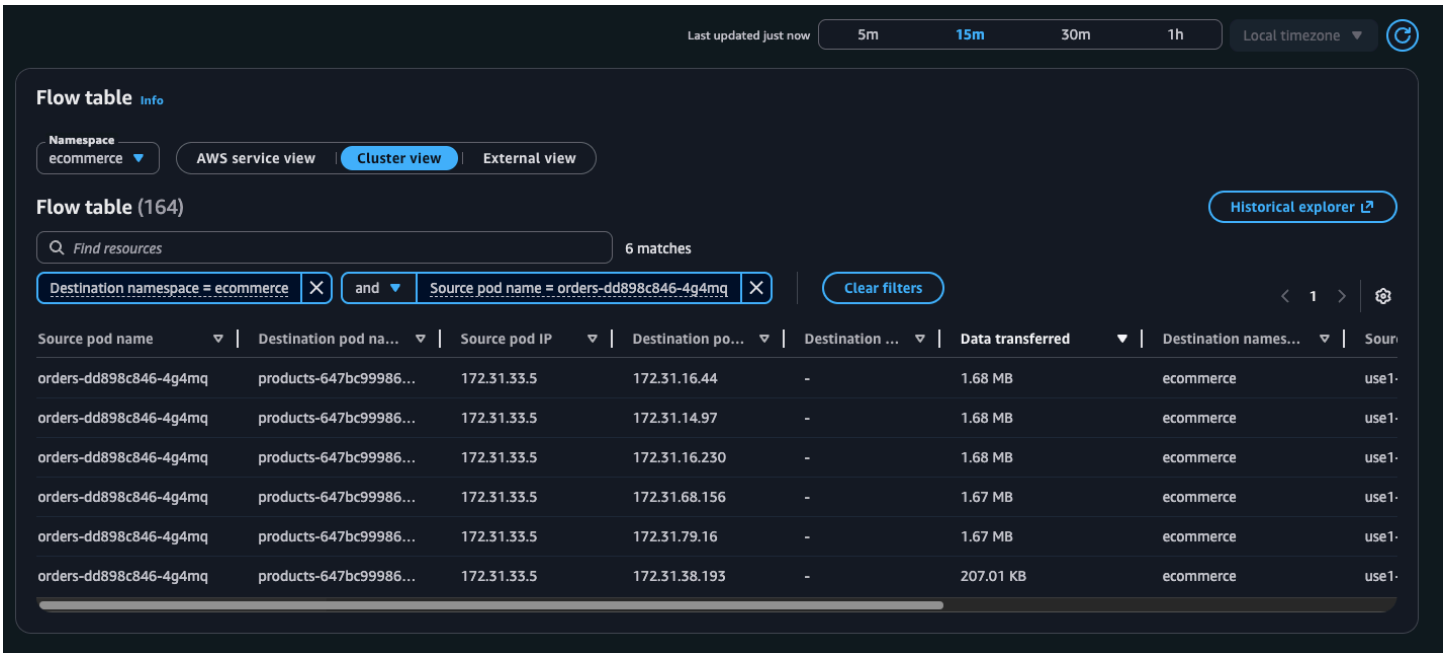


Exemple : tableau de flux

AWS vue du service



Vue du cluster



Considérations et restrictions

- L'observabilité du réseau de conteneurs dans EKS n'est disponible que dans les régions où Network Flow Monitor est pris en charge.
- Les métriques système prises en charge sont OpenMetrics au format et peuvent être directement extraites de l'agent Network Flow Monitor (NFM).

- Pour activer l'observabilité du réseau de conteneurs dans EKS à l'aide de l'infrastructure en tant que code (iAc) comme Terraform, vous devez définir et créer les dépendances suivantes dans vos configurations : portée NFM, moniteur NFM et agent NFM.
- Network Flow Monitor prend en charge jusqu'à environ 5 millions de flux par minute. Il s'agit d'environ 5 000 EC2 instances (nœuds de travail EKS) sur lesquelles l'agent Network Flow Monitor est installé. L'installation d'agents sur plus de 5 000 instances peut affecter les performances de surveillance jusqu'à ce que des capacités supplémentaires soient disponibles.
- Vous devez exécuter une version minimale de 1.1.0 pour le module complémentaire EKS de l'agent NFM.
- Vous devez utiliser la version 6.21.0 ou supérieure du AWS fournisseur Terraform pour prendre en charge les ressources du Network Flow Monitor.
- Pour enrichir les flux réseau avec les métadonnées des pods, vos pods doivent s'exécuter dans leur propre espace de noms réseau isolé, et non dans l'espace de noms du réseau hôte.

Surveiller les métriques de votre cluster avec Prometheus

Prometheus est une base de données de surveillance et de séries chronologiques qui récupère les points de terminaison. Elle permet d'interroger, d'agréger et de stocker les données collectées. Vous pouvez également l'utiliser pour les alertes et l'agrégation des alertes. Cette rubrique explique comment configurer Prometheus en tant qu'option gérée ou open source. La surveillance des métriques du plan de contrôle d'Amazon EKS est un cas d'utilisation courant.

Amazon Managed Service for Prometheus est un service de surveillance et d'alerte compatible avec Prometheus qui facilite la surveillance des applications et infrastructures conteneurisées à grande échelle. Il s'agit d'un service entièrement géré qui met automatiquement à l'échelle l'ingestion, le stockage, l'interrogation et l'alerte de vos métriques. Il s'intègre également aux services de sécurité AWS pour permettre un accès rapide et sécurisé à vos données. Vous pouvez utiliser le langage de requête open source PromQL pour interroger vos métriques et émettre des alertes à leur sujet. Vous pouvez également utiliser le gestionnaire d'alertes dans le service géré Amazon pour Prometheus pour configurer des règles d'alerte pour les alertes critiques. Vous pouvez ensuite envoyer ces alertes critiques sous forme de notifications à une rubrique Amazon SNS.

Il existe plusieurs options différentes pour utiliser Prometheus avec Amazon EKS :

- Vous pouvez activer les métriques Prometheus lors de la création initiale d'un cluster Amazon EKS ou vous pouvez créer votre propre scraper Prometheus pour les clusters existants. Ces deux options sont abordées dans cette rubrique.

- Vous pouvez déployer Prometheus à l'aide de Helm. Pour de plus amples informations, consultez the section called "Pour déployer à l'aide de Helm".
- Vous pouvez afficher les métriques brutes du plan de contrôle au format Prometheus. Pour de plus amples informations, consultez the section called "Plan de contrôle".

Étape 1 : activer les métriques Prometheus

> **Important**
>
> Les ressources du service géré Amazon pour Prometheus ne font pas partie du cycle de vie du cluster et doivent être gérées indépendamment de celui-ci. Lorsque vous supprimez votre cluster, veillez à supprimer également tous les scrapers concernés afin de mettre fin aux coûts applicables. Pour plus d'informations, consultez Rechercher et supprimer des scrapers dans le Guide de l'utilisateur du Service géré Amazon pour Prometheus.

Prometheus détecte et collecte les métriques de votre cluster via un modèle basé sur le pull appelé scraping. Les scrapers sont configurés pour collecter des données à partir de l'infrastructure de votre cluster et des applications conteneurisées. Lorsque vous activez l'option d'envoi des métriques Prometheus, le service géré Amazon pour Prometheus fournit un scraper sans agent entièrement géré.

Si vous n'avez pas encore créé le cluster, vous pouvez activer l'option d'envoi des métriques à Prometheus lors de la création initiale du cluster. Dans la console Amazon EKS, cette option se trouve dans l'étape Configurer l'observabilité de la création d'un nouveau cluster. Pour de plus amples informations, consultez the section called "Créer un cluster".

Si vous disposez déjà d'un cluster, vous pouvez créer votre propre scraper Prometheus. Pour ce faire dans la console Amazon EKS, accédez à l'onglet Observabilité de votre cluster et sélectionnez le bouton Ajouter un scraper. Si vous préférez utiliser l'API AWS ou l'interface AWS CLI, consultez Créer un scraper dans le Guide de l'utilisateur du Service géré Amazon pour Prometheus.

Les options suivantes sont disponibles lors de la création du scraper avec la console Amazon EKS.

Alias du scraper

(Facultatif) Saisissez un alias unique pour le scraper.

Destination (Destination)

Choisissez un espace de travail Amazon Managed Service for Prometheus. Un espace de travail est un espace logique dédié au stockage et à l'interrogation des métriques Prometheus. Avec cet espace de travail, vous pourrez afficher les métriques Prometheus sur tous les comptes qui y ont accès. L'option Créer un nouvel espace de travail indique à Amazon EKS de créer un espace de travail en votre nom à l'aide de l'Alias d'espace de travail que vous fournissez. L'option Sélectionner un espace de travail existant vous permet de sélectionner un espace de travail existant dans une liste déroulante. Pour plus d'informations sur les espaces de travail, consultez Gestion des espaces de travail dans le Guide de l'utilisateur Amazon Managed Service for Prometheus.

Accès à un service

Cette section résume les autorisations que vous accordez lorsque vous envoyez des métriques Prometheus :

- Autoriser Amazon Managed Service for Prometheus à décrire le cluster Amazon EKS récupéré
- Autoriser l'écriture à distance dans l'espace de travail Amazon Managed Prometheus

Si `AmazonManagedScraperRole` existe déjà, le scraper l'utilise. Choisissez le lien `AmazonManagedScraperRole` pour voir les Détails de l'autorisation. Si le rôle `AmazonManagedScraperRole` n'existe pas encore, sélectionnez le lien Afficher les détails des autorisations pour voir les autorisations spécifiques que vous accordez en envoyant des métriques Prometheus.

Sous-réseaux

Modifiez les sous-réseaux que le scraper héritera selon vos besoins. Si vous devez ajouter une option de sous-réseau grisée, revenez à l'étape Spécifier le réseau de la création du cluster.

Configuration du scraper

Modifiez la configuration du scraper au format YAML si nécessaire. Pour ce faire, utilisez le formulaire ou chargez un fichier YAML de remplacement. Pour plus d'informations, consultez Configuration du scraper dans le Guide de l'utilisateur Amazon Managed Service for Prometheus.

Amazon Managed Service for Prometheus fait référence au scraper sans agent créé aux côtés du cluster en tant que collecteur géré AWS. Pour plus d'informations sur les collecteurs gérés par AWS, consultez Ingérer des métriques avec les collecteurs gérés par AWS dans le Guide de l'utilisateur du Service géré Amazon pour Prometheus.

⚠ Important

- Si vous créez un scraper Prometheus à l'aide de l'interface AWS CLI ou de l'API AWS, vous devez ajuster sa configuration pour lui attribuer des autorisations au sein du cluster. Pour plus d'informations, consultez Configuration de votre cluster Amazon EKS dans le Guide de l'utilisateur Amazon Managed Service for Prometheus.
- Si vous disposez d'un scraper Prometheus créé avant le 11 novembre 2024 qui utilise le `aws-auth ConfigMap` au lieu des entrées d'accès, vous devez le mettre à jour pour accéder à des métriques supplémentaires à partir du plan de contrôle du cluster Amazon EKS. Pour la configuration mise à jour, consultez Configuration manuelle d'Amazon EKS pour l'accès au scraper dans le Guide de l'utilisateur du Service géré Amazon pour Prometheus.

Étape 2 : utiliser les métriques Prometheus

Pour plus d'informations sur l'utilisation des métriques Prometheus après les avoir activées pour votre cluster, consultez le Guide de l'utilisateur du Service géré Amazon pour Prometheus.

Étape 3 : gérer les scrapers Prometheus

Pour gérer les scrapers, sélectionnez l'onglet Observabilité dans la console Amazon EKS. Un tableau affiche la liste des scrappers du cluster, avec des informations telles que l'ID de scraper, l'alias, le statut et la date de création. Vous pouvez ajouter d'autres scrapers, modifier des scrapers, supprimer des scrapers ou afficher plus d'informations sur les scrapers actuels.

Pour obtenir plus de détails sur un scraper, sélectionnez le lien ID du scraper. Vous pouvez par exemple afficher l'ARN, l'environnement, l'ID de l'espace de travail, le rôle IAM, la configuration et les informations réseau. Vous pouvez utiliser l'ID du scraper comme entrée pour les opérations d'API du service géré Amazon pour Prometheus telles que `DescribeScraper`, `UpdateScraper` et `DeleteScraper`. Pour plus d'informations sur l'utilisation de l'API Prometheus, consultez la référence d'API du service géré Amazon pour Prometheus.

Déployer Prometheus à l'aide de Helm

Au lieu d'utiliser le service géré Amazon pour Prometheus, vous pouvez déployer Prometheus dans votre cluster avec Helm. Si Helm est déjà installé, vous pouvez vérifier votre version avec la commande `helm version`. Helm est un gestionnaire de package pour les clusters Kubernetes.

Pour de plus amples informations sur Helm et son installation, veuillez consulter the section called "Déployer des applications avec Helm".

Une fois que vous avez configuré Helm pour votre cluster Amazon EKS, vous pouvez l'utiliser pour déployer Prometheus avec les étapes suivantes.

1. Créez un espace de noms Prometheus.

```
kubectl create namespace prometheus
```

2. Ajoutez le référentiel de graphiques `prometheus-community`.

```
helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
```

3. Déployez Prometheus.

```
helm upgrade -i prometheus prometheus-community/prometheus \
    --namespace prometheus \
    --set alertmanager.persistence.storageClass="gp2" \
    --set server.persistentVolume.storageClass="gp2"
```

> **Note**
>
> Si vous obtenez l'erreur `Error: failed to download "stable/prometheus" (hint: running helm repo update may help)` lors de l'exécution de cette commande, exécutez `helm repo update prometheus-community`, puis réessayez d'exécuter la commande Étape 2.

   Si vous obtenez l'erreur `Error: rendered manifests contain a resource that already exists`, exécutez `helm uninstall` *`your-release-name`* `-n` *`namespace`* , puis réessayez d'exécuter la commande Étape 3.

4. Vérifiez que tous les pods dans l'espace de noms `prometheus` sont à l'état `READY`.

```
kubectl get pods -n prometheus
```

   L'exemple qui suit illustre un résultat.

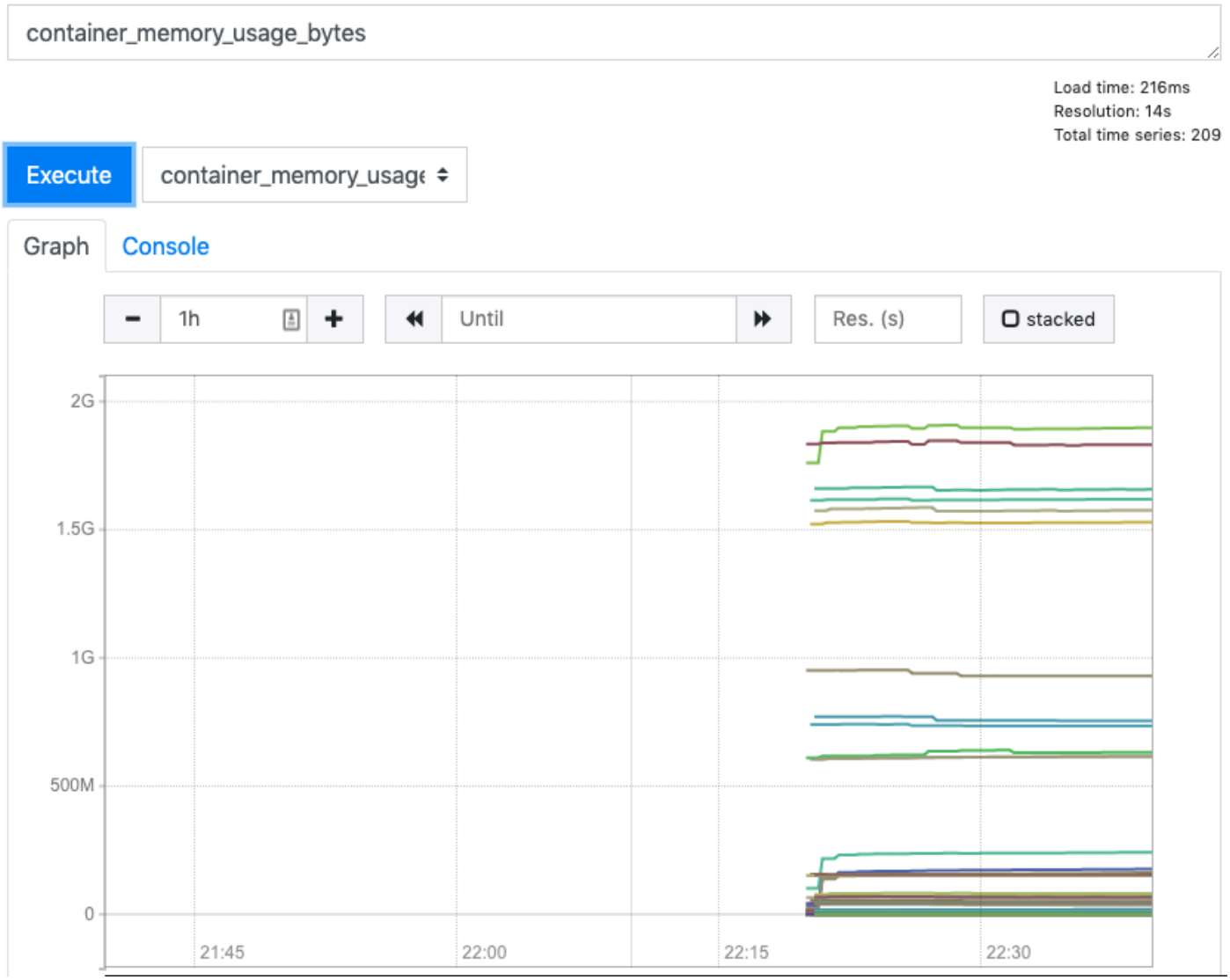
```
NAME                                             READY   STATUS    RESTARTS   AGE
```

```
prometheus-alertmanager-59b4c8c744-r7bgp          1/2     Running   0          48s
prometheus-kube-state-metrics-7cfd87cf99-jkz2f    1/1     Running   0          48s
prometheus-node-exporter-jcjqz                    1/1     Running   0          48s
prometheus-node-exporter-jxv2h                    1/1     Running   0          48s
prometheus-node-exporter-vbdks                    1/1     Running   0          48s
prometheus-pushgateway-76c444b68c-82tnw           1/1     Running   0          48s
prometheus-server-775957f748-mmht9                1/2     Running   0          48s
```

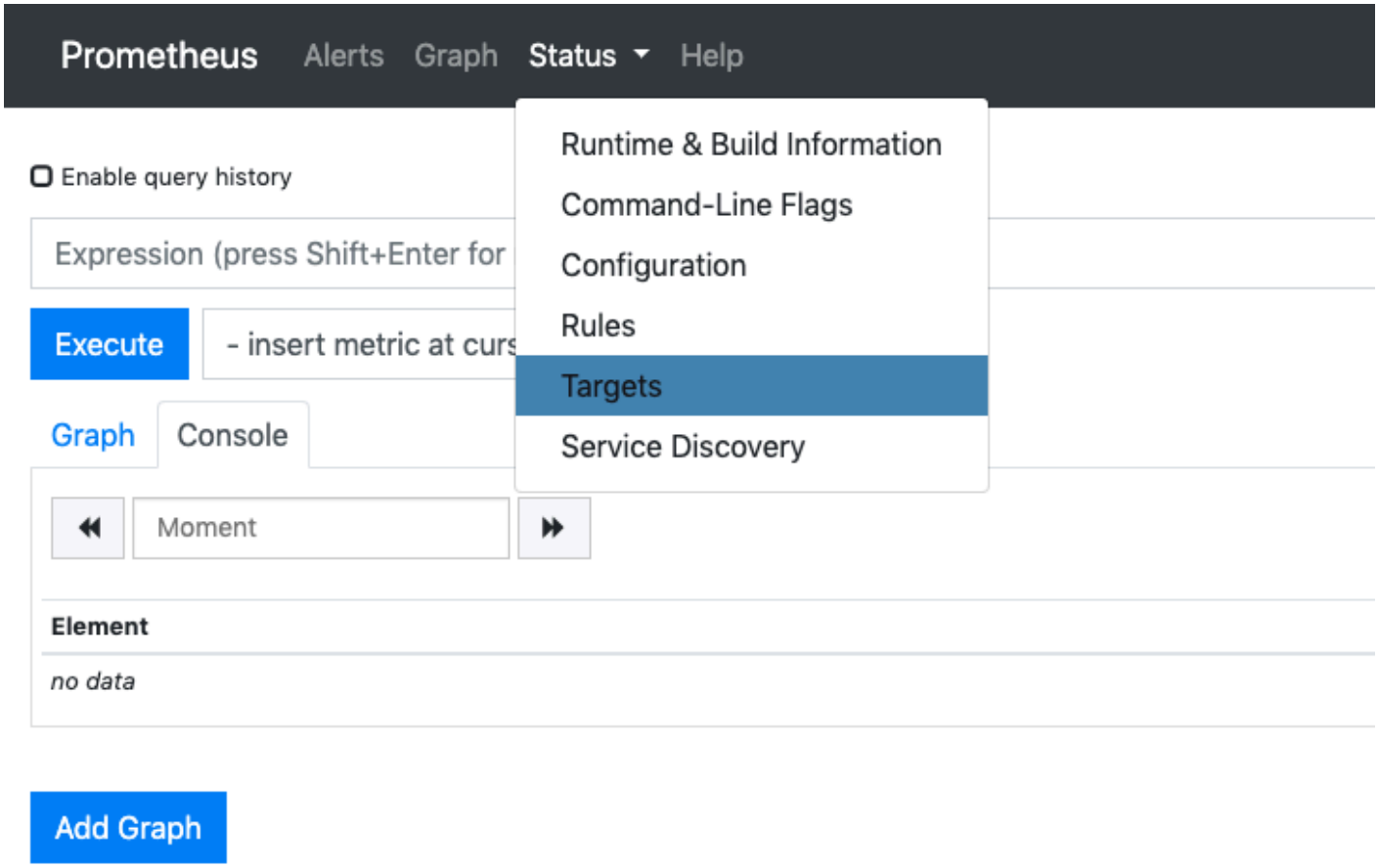
5. Utilisez `kubectl` pour transférer la console Prometheus vers votre ordinateur local.

```
kubectl --namespace=prometheus port-forward deploy/prometheus-server 9090
```

6. Ouvrez un navigateur Web sur `http://localhost:9090` pour afficher la console Prometheus.
7. Choisissez une métrique dans le menu - insert metric at cursor (insérer une métrique sur le curseur), puis choisissez Execute (Exécuter). Choisissez l'onglet Graph (Graphique) pour afficher la métrique au fil du temps. L'image suivante montre `container_memory_usage_bytes` au fil du temps.



8. Dans la barre de navigation supérieure, choisissez Status (Statut), puis Targets (Cibles).



Tous les points de terminaison Kubernetes qui sont connectés à Prometheus à l'aide de la découverte de service sont affichés.

Récupérer les métriques brutes du plan de contrôle au format Prometheus

Le plan de contrôle Kubernetes expose un certain nombre de métriques représentées au format Prometheus. Ces métriques sont utiles pour la surveillance et l'analyse. Elles sont exposées en interne via des points de terminaison de métriques et sont accessibles sans avoir à déployer Prometheus dans son intégralité. Cependant, le déploiement de Prometheus facilite l'analyse des métriques au fil du temps.

Pour afficher les métriques brutes, remplacez `endpoint` et exécutez la commande suivante.

```
kubectl get --raw endpoint
```

Cette commande vous permet de passer n'importe quel chemin d'accès au point de terminaison et renvoie la réponse brute. La sortie répertorie différentes métriques ligne par ligne, chaque ligne comprenant un nom de métrique, des balises et une valeur.

```
metric_name{tag="value"[,...]} value
```

Récupérer les métriques à partir du serveur API

Le point de terminaison général du serveur API est exposé sur le plan de contrôle Amazon EKS. Ce point de terminaison est particulièrement utile lorsque vous recherchez une métrique spécifique.

```
kubectl get --raw /metrics
```

L'exemple qui suit illustre un résultat.

```
[...]
# HELP rest_client_requests_total Number of HTTP requests, partitioned by status code,
 method, and host.
# TYPE rest_client_requests_total counter
rest_client_requests_total{code="200",host="127.0.0.1:21362",method="POST"} 4994
rest_client_requests_total{code="200",host="127.0.0.1:443",method="DELETE"} 1
rest_client_requests_total{code="200",host="127.0.0.1:443",method="GET"} 1.326086e+06
rest_client_requests_total{code="200",host="127.0.0.1:443",method="PUT"} 862173
rest_client_requests_total{code="404",host="127.0.0.1:443",method="GET"} 2
rest_client_requests_total{code="409",host="127.0.0.1:443",method="POST"} 3
rest_client_requests_total{code="409",host="127.0.0.1:443",method="PUT"} 8
# HELP ssh_tunnel_open_count Counter of ssh tunnel total open attempts
# TYPE ssh_tunnel_open_count counter
ssh_tunnel_open_count 0
# HELP ssh_tunnel_open_fail_count Counter of ssh tunnel failed open attempts
# TYPE ssh_tunnel_open_fail_count counter
ssh_tunnel_open_fail_count 0
```

Cette sortie brute renvoie tel quel ce que le serveur d'API expose.

Récupérer les métriques du plan de contrôle avec **`metrics.eks.amazonaws.com`**

Pour les clusters Kubernetes version `1.28` et supérieure, Amazon EKS expose également les métriques sous le groupe API `metrics.eks.amazonaws.com`. Ces métriques incluent les composants du plan de contrôle tels que `kube-scheduler` et `kube-controller-manager`.

> **Note**
>
> Si vous disposez d'une configuration de webhook susceptible de bloquer la création de la nouvelle ressource `APIService` `v1.metrics.eks.amazonaws.com` sur votre cluster, la fonctionnalité du point de terminaison des métriques peut ne pas être disponible. Vous pouvez le vérifier dans le journal d'audit `kube-apiserver` en recherchant le mot-clé `v1.metrics.eks.amazonaws.com`.

Récupérer les métriques **`kube-scheduler`**

Pour récupérer les métriques `kube-scheduler`, utilisez la commande suivante.

```
kubectl get --raw "/apis/metrics.eks.amazonaws.com/v1/ksh/container/metrics"
```

L'exemple qui suit illustre un résultat.

```
# TYPE scheduler_pending_pods gauge
scheduler_pending_pods{queue="active"} 0
scheduler_pending_pods{queue="backoff"} 0
scheduler_pending_pods{queue="gated"} 0
scheduler_pending_pods{queue="unschedulable"} 18
# HELP scheduler_pod_scheduling_attempts [STABLE] Number of attempts to successfully
 schedule a pod.
# TYPE scheduler_pod_scheduling_attempts histogram
scheduler_pod_scheduling_attempts_bucket{le="1"} 79
scheduler_pod_scheduling_attempts_bucket{le="2"} 79
scheduler_pod_scheduling_attempts_bucket{le="4"} 79
scheduler_pod_scheduling_attempts_bucket{le="8"} 79
scheduler_pod_scheduling_attempts_bucket{le="16"} 79
scheduler_pod_scheduling_attempts_bucket{le="+Inf"} 81
[...]
```

Récupérer les métriques **`kube-controller-manager`**

Pour récupérer les métriques `kube-controller-manager`, utilisez la commande suivante.

```
kubectl get --raw "/apis/metrics.eks.amazonaws.com/v1/kcm/container/metrics"
```

L'exemple qui suit illustre un résultat.

```
[...]
workqueue_work_duration_seconds_sum{name="pvprotection"} 0
workqueue_work_duration_seconds_count{name="pvprotection"} 0
workqueue_work_duration_seconds_bucket{name="replicaset",le="1e-08"} 0
workqueue_work_duration_seconds_bucket{name="replicaset",le="1e-07"} 0
workqueue_work_duration_seconds_bucket{name="replicaset",le="1e-06"} 0
workqueue_work_duration_seconds_bucket{name="replicaset",le="9.999999999999999e-06"} 0
workqueue_work_duration_seconds_bucket{name="replicaset",le="9.999999999999999e-05"} 19
workqueue_work_duration_seconds_bucket{name="replicaset",le="0.001"} 109
workqueue_work_duration_seconds_bucket{name="replicaset",le="0.01"} 139
workqueue_work_duration_seconds_bucket{name="replicaset",le="0.1"} 181
workqueue_work_duration_seconds_bucket{name="replicaset",le="1"} 191
workqueue_work_duration_seconds_bucket{name="replicaset",le="10"} 191
workqueue_work_duration_seconds_bucket{name="replicaset",le="+Inf"} 191
workqueue_work_duration_seconds_sum{name="replicaset"} 4.265655885000002
[...]
```

Comprendre les métriques du planificateur et du gestionnaire de contrôleurs

Le tableau suivant décrit les métriques du planificateur et du gestionnaire de contrôleurs qui sont disponibles pour le scraping de type Prometheus. Pour plus d'informations sur ces métriques, consultez Référence des métriques Kubernetes dans la documentation Kubernetes.

Métrique	Composant du plan de contrôle	Description
scheduler_pending_pods	planificateur	Nombre de pods en attente d'être planifiés sur un nœud pour exécution.
scheduler_schedule_attempts_total	planificateur	Nombre de tentatives effectuées pour planifier des pods.
scheduler_preemption_attempts_total	planificateur	Nombre de tentatives effectuées par le planificateur pour planifier des pods de priorité supérieure en évictant ceux de priorité inférieure.

Métrique	Composant du plan de contrôle	Description
scheduler_preemption_victims	planificateur	Nombre de pods sélectionnés pour être évictés afin de libérer de l'espace pour des pods de priorité supérieure.
scheduler_pod_scheduling_attempts	planificateur	Nombre de tentatives pour planifier avec succès un pod.
scheduler_scheduling_attempt_duration_seconds	planificateur	Indique la rapidité ou la lenteur avec laquelle le planificateur est capable de trouver un emplacement approprié pour l'exécution d'un pod en fonction de divers facteurs tels que la disponibilité des ressources et les règles de planification.
scheduler_pod_scheduling_sli_duration_seconds	planificateur	Latence de bout en bout pour un pod en cours de planification, à partir du moment où le pod entre dans la file d'attente de planification. Cela peut impliquer plusieurs tentatives de planification.
cronjob_controller_job_creation_skew_duration_seconds	gestionnaire de contrôleur	Temps entre le moment où une tâche cron est planifiée pour être exécutée et le moment où la tâche correspondante est créée.
workqueue_depth	gestionnaire de contrôleur	La profondeur actuelle de la file d'attente.

Métrique	Composant du plan de contrôle	Description
workqueue_adds_total	gestionnaire de contrôleur	Le nombre total d'ajouts traités par la file d'attente de travail.
workqueue_queue_duration_seconds	gestionnaire de contrôleur	Le temps en secondes pendant lequel un élément reste dans la file d'attente de travail avant d'être demandé.
workqueue_work_duration_seconds	gestionnaire de contrôleur	Le temps en secondes nécessaire au traitement d'un élément de la file d'attente de travail.

Déployer un scraper Prometheus pour récupérer les métriques de manière cohérente

Pour déployer un scraper Prometheus afin de récupérer les métriques de manière cohérente, utilisez la configuration suivante :

```
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: prometheus-conf
data:
  prometheus.yml: |-
    global:
      scrape_interval: 30s
    scrape_configs:
    # apiserver metrics
    - job_name: apiserver-metrics
      kubernetes_sd_configs:
      - role: endpoints
      scheme: https
      tls_config:
        ca_file: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
        insecure_skip_verify: true
      bearer_token_file: /var/run/secrets/kubernetes.io/serviceaccount/token
```

```
      relabel_configs:
      - source_labels:
          [
            __meta_kubernetes_namespace,
            __meta_kubernetes_service_name,
            __meta_kubernetes_endpoint_port_name,
          ]
        action: keep
        regex: default;kubernetes;https
    # Scheduler metrics
    - job_name: 'ksh-metrics'
      kubernetes_sd_configs:
      - role: endpoints
      metrics_path: /apis/metrics.eks.amazonaws.com/v1/ksh/container/metrics
      scheme: https
      tls_config:
        ca_file: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
        insecure_skip_verify: true
      bearer_token_file: /var/run/secrets/kubernetes.io/serviceaccount/token
      relabel_configs:
      - source_labels:
          [
            __meta_kubernetes_namespace,
            __meta_kubernetes_service_name,
            __meta_kubernetes_endpoint_port_name,
          ]
        action: keep
        regex: default;kubernetes;https
    # Controller Manager metrics
    - job_name: 'kcm-metrics'
      kubernetes_sd_configs:
      - role: endpoints
      metrics_path: /apis/metrics.eks.amazonaws.com/v1/kcm/container/metrics
      scheme: https
      tls_config:
        ca_file: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
        insecure_skip_verify: true
      bearer_token_file: /var/run/secrets/kubernetes.io/serviceaccount/token
      relabel_configs:
      - source_labels:
          [
            __meta_kubernetes_namespace,
            __meta_kubernetes_service_name,
            __meta_kubernetes_endpoint_port_name,
```

```
          ]
        action: keep
        regex: default;kubernetes;https
---
apiVersion: v1
kind: Pod
metadata:
  name: prom-pod
spec:
  containers:
  - name: prom-container
    image: prom/prometheus
    ports:
    - containerPort: 9090
    volumeMounts:
    - name: config-volume
      mountPath: /etc/prometheus/
  volumes:
  - name: config-volume
    configMap:
      name: prometheus-conf
```

L'autorisation suivante est requise pour que le pod puisse accéder au nouveau point de terminaison des métriques.

```
{
  "effect": "allow",
  "apiGroups": [
    "metrics.eks.amazonaws.com"
  ],
  "resources": [
    "kcm/metrics",
    "ksh/metrics"
  ],
  "verbs": [
    "get"
  ] },
```

Pour appliquer un correctif au rôle utilisé, vous pouvez utiliser la commande suivante.

```
kubectl patch clusterrole <role-name> --type=json -p='[
  {
    "op": "add",
```

```
    "path": "/rules/-",
    "value": {
      "verbs": ["get"],
      "apiGroups": ["metrics.eks.amazonaws.com"],
      "resources": ["kcm/metrics", "ksh/metrics"]
    }
  }
]'
```

Vous pouvez ensuite afficher le tableau de bord Prometheus en redirigeant le port du scraper Prometheus vers votre port local.

```
kubectl port-forward pods/prom-pod 9090:9090
```

Pour votre cluster Amazon EKS, les métriques du plan de contrôle Kubernetes sont également ingérées dans Amazon CloudWatch Metrics sous l'espace de noms `AWS/EKS`. Pour les afficher, ouvrez la console CloudWatch et sélectionnez Toutes les métriques dans le panneau de navigation gauche. Sur la page de sélection Métriques, choisissez l'espace de noms `AWS/EKS` et une dimension de métrique pour votre cluster.

Surveillez les données du cluster avec Amazon CloudWatch

Amazon CloudWatch est un service de surveillance qui collecte des statistiques et des journaux à partir de vos ressources cloud. CloudWatch fournit gratuitement des métriques Amazon EKS de base lorsque vous utilisez un nouveau cluster de version `1.28` ou supérieure. Toutefois, lorsque vous utilisez l'opérateur CloudWatch d'observabilité en tant que module complémentaire Amazon EKS, vous pouvez bénéficier de fonctionnalités d'observabilité améliorées.

Statistiques de base sur Amazon CloudWatch

Pour les clusters de versions `1.28` Kubernetes ou supérieures, vous pouvez obtenir des métriques gratuites CloudWatch dans l'espace de noms. `AWS/EKS` Le tableau ci-dessous présente les principales métriques disponibles pour les versions prises en charge. Chaque métrique est fournie avec une fréquence d'une minute.

Nom des métriques	Description
`apiserver_flowcontrol_current_executing_seats`	Le nombre de postes actuellement utilisés pour exécuter les demandes d'API. L'attribution des

Nom des métriques	Description
	sièges est déterminée par les configurations priority_level et flow_schema dans la fonctionnalité Priority and Fairness de l'API Kubernetes. Unités : nombre Statistiques valides : somme
scheduler_schedule_attempts_total	Nombre total de tentatives effectuées par le planificateur pour planifier les pods dans le cluster sur une période donnée. Cette métrique permet de surveiller la charge de travail du planificateur et peut indiquer une pression de planification ou des problèmes potentiels d'affectation des pods. Unités : nombre Statistiques valides : somme
scheduler_schedule_attempts_SCHEDULED	Nombre de tentatives réussies du planificateur pour affecter des pods à des nœuds du cluster sur une période donnée. Unités : nombre Statistiques valides : somme
scheduler_schedule_attempts_UNSCHEDULABLE	Nombre de tentatives de planification de pods qui n'ont pas pu être planifiés sur une période donnée en raison de contraintes valides, telles que l'insuffisance du processeur ou de la mémoire sur un nœud. Unités : nombre Statistiques valides : somme

Nom des métriques	Description
scheduler_schedule_attempts_ERROR	Nombre de tentatives de planification de pods ayant échoué pendant une période donnée en raison d'un problème interne du planificateur lui-même, par exemple, des problèmes de connectivité avec le serveur API. Unités : nombre Statistiques valides : somme
scheduler_pending_pods	Nombre total de pods en attente de planification par le planificateur dans le cluster pendant une période donnée. Unités : nombre Statistiques valides : somme
scheduler_pending_pods_ACTIVEQ	Nombre de pods en attente dans activeQ, qui attendent d'être planifiés dans le cluster pendant une période donnée. Unités : nombre Statistiques valides : somme
scheduler_pending_pods_UNSCHEDULABLE	Nombre de pods en attente pour lesquels le planificateur a tenté la planification, mais a échoué, et qui restent dans un état non planifiable en attente d'une nouvelle tentative. Unités : nombre Statistiques valides : somme

Nom des métriques	Description
`scheduler_pending_pods_BACKOFF`	Nombre de pods en attente dans `backoffQ`, en état de backoff, qui attendent la fin de leur période de backoff. Unités : nombre Statistiques valides : somme
`scheduler_pending_pods_GATED`	Nombre de pods en attente dans un état d'accès contrôlé, qui ne peuvent pas être planifiés tant qu'ils ne remplissent pas certaines conditions requises. Unités : nombre Statistiques valides : somme
`apiserver_request_total`	Nmbre de requêtes HTTP adressées à tous les serveurs API du cluster. Unités : nombre Statistiques valides : somme
`apiserver_request_total_4XX`	Nombre de requêtes HTTP adressées à tous les serveurs API du cluster ayant renvoyé un code d'état `4XX` (erreur client). Unités : nombre Statistiques valides : somme

Nom des métriques	Description
apiserver_request_total_429	Nombre de requêtes HTTP adressées à tous les serveurs API du cluster ayant renvoyé un code d'état 429, qui se produit lorsque les clients dépassent les seuils de limitation du débit. Unités : nombre Statistiques valides : somme
apiserver_request_total_5XX	Nombre de requêtes HTTP adressées à tous les serveurs API du cluster ayant renvoyé un code d'état 5XX (erreur serveur). Unités : nombre Statistiques valides : somme
apiserver_request_total_LIST_PODS	Nombre de requêtes des pods LIST adressées à tous les serveurs API du cluster. Unités : nombre Statistiques valides : somme
apiserver_request_duration_seconds_PUT_P99	99e percentile de la latence des requêtes PUT calculé à partir de toutes les requêtes adressées à tous les serveurs API du cluster. Représente le temps de réponse en dessous duquel 99 % des requêtes PUT sont traitées. Unités : secondes Statistiques valides : moyenne

Nom des métriques	Description
apiserver_request_duration_seconds_PATCH_P99	99e percentile de la latence des requêtes PATCH calculé à partir de toutes les requêtes adressées à tous les serveurs API du cluster. Représente le temps de réponse en dessous duquel 99 % des requêtes PATCH sont traitées. Unités : secondes Statistiques valides : moyenne
apiserver_request_duration_seconds_POST_P99	99e percentile de la latence des requêtes POST calculé à partir de toutes les requêtes adressées à tous les serveurs API du cluster. Représente le temps de réponse en dessous duquel 99 % des requêtes POST sont traitées. Unités : secondes Statistiques valides : moyenne
apiserver_request_duration_seconds_GET_P99	99e percentile de la latence des requêtes GET calculé à partir de toutes les requêtes adressées à tous les serveurs API du cluster. Représente le temps de réponse en dessous duquel 99 % des requêtes GET sont traitées. Unités : secondes Statistiques valides : moyenne

Nom des métriques	Description
apiserver_request_duration_seconds_LIST_P99	99e percentile de la latence des requêtes LIST calculé à partir de toutes les requêtes adressées à tous les serveurs API du cluster. Représente le temps de réponse en dessous duquel 99 % des requêtes LIST sont traitées. Unités : secondes Statistiques valides : moyenne
apiserver_request_duration_seconds_DELETE_P99	99e percentile de la latence des requêtes DELETE calculé à partir de toutes les requêtes adressées à tous les serveurs API du cluster. Représente le temps de réponse en dessous duquel 99 % des requêtes DELETE sont traitées. Unités : secondes Statistiques valides : moyenne
apiserver_current_inflight_requests_MUTATING	Nombre de requêtes mutantes (POST, PUT, DELETE, PATCH) en cours de traitement sur l'ensemble des serveurs API du cluster. Cette métrique représente les requêtes en cours, dont le traitement n'est pas encore terminé. Unités : nombre Statistiques valides : somme

Nom des métriques	Description
`apiserver_current_inflight_requests_READONLY`	Nombre de requêtes en lecture seule (`GET`, `LIST`) actuellement traitées sur tous les serveurs API du cluster. Cette métrique représente les requêtes en cours, dont le traitement n'est pas encore terminé. Unités : nombre Statistiques valides : somme
`apiserver_admission_webhook_request_total`	Nombre de requêtes adressées aux webhooks d'admission sur tous les serveurs API du cluster. Unités : nombre Statistiques valides : somme
`apiserver_admission_webhook_request_total_ADMIT`	Nombre de requêtes adressées aux webhooks d'admission mutants sur tous les serveurs API du cluster. Unités : nombre Statistiques valides : somme
`apiserver_admission_webhook_request_total_VALIDATING`	Nombre de requêtes adressées aux webhooks d'admission validants sur tous les serveurs API du cluster. Unités : nombre Statistiques valides : somme

Nom des métriques	Description
`apiserver_admission_webhook_rejection_count`	Nombre de requêtes adressées aux webhooks d'admission sur tous les serveurs API du cluster qui ont été rejetées. Unités : nombre Statistiques valides : somme
`apiserver_admission_webhook_rejection_count_ADMIT`	Nombre de requêtes adressées aux webhooks d'admission mutants sur tous les serveurs API du cluster qui ont été rejetées. Unités : nombre Statistiques valides : somme
`apiserver_admission_webhook_rejection_count_VALIDATING`	Nombre de requêtes adressées aux webhooks d'admission validants sur tous les serveurs API du cluster qui ont été rejetées. Unités : nombre Statistiques valides : somme
`apiserver_admission_webhook_admission_duration_seconds`	Le 99e centile de la latence des requêtes adressées aux webhooks d'admission validants , calculé à partir de toutes les requêtes sur l'ensemble des serveurs API du cluster. Représente le temps de réponse en dessous duquel 99 % de toutes les requêtes adressées aux webhooks d'admission validants sont terminées. Unités : secondes Statistiques valides : moyenne

Nom des métriques	Description
`apiserver_admission_webhook_admission_duration_seconds_ADMIT_P99`	Le 99e centile de la latence des requêtes adressées aux webhooks d'admission mutants, calculé à partir de toutes les requêtes sur l'ensemble des serveurs API du cluster. Représente le temps de réponse en dessous duquel 99 % de toutes les requêtes adressées aux webhooks d'admission mutants sont terminées. Unités : secondes Statistiques valides : moyenne
`apiserver_admission_webhook_admission_duration_seconds_VALIDATING_P99`	Le 99e centile de la latence des requêtes adressées aux webhooks d'admission tiers, calculé à partir de toutes les requêtes sur l'ensemble des serveurs API du cluster. Représente le temps de réponse en dessous duquel 99 % de toutes les requêtes adressées aux webhooks d'admission tiers sont terminées . Unités : secondes Statistiques valides : moyenne
`apiserver_storage_size_bytes`	Taille physique, en octets, du fichier de base de données etcd utilisé par les serveurs API du cluster. Cette métrique représente l'espace disque réel alloué au stockage. Unités : octets Statistiques valides : maximum

Opérateur Amazon CloudWatch Observability

Amazon CloudWatch Observability collecte des journaux, des métriques et des données de suivi en temps réel. Il les envoie à Amazon CloudWatch et à AWS X-Ray. Vous pouvez installer ce module complémentaire pour activer à la fois CloudWatch Application Signals et CloudWatch Container Insights avec une observabilité améliorée pour Amazon EKS. Cela vous aide à surveiller l'état et les performances de votre infrastructure et de vos applications conteneurisées. L'opérateur Amazon CloudWatch Observability est conçu pour installer et configurer les composants nécessaires.

Amazon EKS prend en charge l'opérateur CloudWatch d'observabilité en tant que module complémentaire Amazon EKS. Ce module complémentaire active Container Insights sur les composants master Linux et Windows du cluster. Pour activer Container Insights sous Windows, la version du module complémentaire Amazon EKS doit être `1.5.0` ou supérieure. Actuellement, CloudWatch Application Signals n'est pas pris en charge sur Amazon EKS Windows.

Les rubriques ci-dessous décrivent comment commencer à utiliser CloudWatch Observability Operator pour votre cluster Amazon EKS.

- Pour obtenir des instructions sur l'installation de ce module complémentaire, consultez Installer l' CloudWatch agent avec le module complémentaire Amazon CloudWatch Observability EKS ou le tableau Helm dans le guide de l' CloudWatch utilisateur Amazon.
- Pour plus d'informations sur les signaux CloudWatch d'application, consultez la section Signaux d'application dans le guide de CloudWatch l'utilisateur Amazon.
- Pour plus d'informations sur Container Insights, consultez la section Utilisation de Container Insights dans le guide de CloudWatch l'utilisateur Amazon.

Envoyer les journaux du plan de contrôle à CloudWatch Logs

La journalisation du plan de contrôle Amazon EKS fournit des journaux d'audit et de diagnostic directement depuis le plan de contrôle Amazon EKS vers CloudWatch les journaux de votre compte. Ces journaux vous permettent de sécuriser et d'exécuter facilement vos clusters. Vous pouvez sélectionner les types de journaux exacts dont vous avez besoin, et les journaux sont envoyés sous forme de flux de journaux à un groupe pour chaque cluster Amazon EKS inclus CloudWatch. Vous pouvez utiliser des filtres CloudWatch d'abonnement pour analyser les journaux en temps réel ou pour les transmettre à d'autres services (les journaux seront codés en Base64 et compressés au format gzip). Pour plus d'informations, consultez Amazon CloudWatch logging.

> **Note**
>
> Les journaux du plan de contrôle Amazon EKS sont transmis à CloudWatch Logs en quelques minutes. Cependant, la livraison des journaux constitue le meilleur effort.

Vous pouvez commencer à utiliser la journalisation de plan de contrôle d'Amazon EKS en choisissant les types de journal que vous voulez activer pour chaque cluster Amazon EKS nouveau ou existant. Vous pouvez activer ou désactiver chaque type de journal par cluster à l'aide de la AWS Management Console AWS CLI (version `1.16.139` ou supérieure) ou via l'API Amazon EKS. Lorsque cette option est activée, les journaux sont automatiquement envoyés depuis le cluster Amazon EKS aux CloudWatch journaux du même compte.

Lorsque vous utilisez la journalisation de plan de contrôle d'Amazon EKS, des frais standards Amazon EKS vous sont facturés pour chaque cluster que vous exécutez. Les frais standard d'ingestion et de stockage des données CloudWatch Logs vous sont facturés pour tous les journaux envoyés à CloudWatch Logs depuis vos clusters. Toutes les AWS ressources, telles que les instances Amazon EC2 ou les volumes Amazon EBS, que vous fournissez dans le cadre de votre cluster vous sont également facturées.

Les types de journal de plan de contrôle de cluster suivants sont disponibles. Chaque type de journal correspond à un composant du plan de contrôle Kubernetes. Pour en savoir plus sur ces composants, consultez Kubernetes Components dans la documentation Kubernetes.

Serveur d'API**`api`** ()

Le serveur API de votre cluster est le composant du plan de contrôle qui expose l'API Kubernetes. Si vous activez les journaux du serveur d'API au moment du lancement du cluster, ou peu après, les journaux incluent les indicateurs qui ont été utilisés pour démarrer le serveur d'API. Pour plus d'informations, consultez kube-apiserver et la stratégie d'audit dans la documentation Kubernetes.

Audit**`audit`** ()

Les journaux d'audit Kubernetes fournissent un enregistrement des utilisateurs individuels, des administrateurs ou des composants système qui ont affecté votre cluster. Pour plus d'informations, consultez la section Audit dans la documentation Kubernetes.

Authentificateur**`authenticator`** ()

Les journaux d'authentification sont propres à Amazon EKS. Ces journaux représentent le composant de plan de contrôle utilisé par Amazon EKS pour l'authentification de contrôle d'accès

basé sur les rôles (RBAC) Kubernetes à l'aide d'informations d'identification IAM. Pour de plus amples informations, veuillez consulter Gestion du cluster.

Gestionnaire de contrôleur**`controllerManager`** ()

Le gestionnaire de contrôleurs gère les boucles de contrôle principales fournies avec Kubernetes. Pour plus d'informations, consultez kube-controller-manager dans la documentation Kubernetes.

Planificateur**`scheduler`** ()

Le composant planificateur gère quand et où exécuter les pods dans votre cluster. Pour plus d'informations, consultez kube-scheduler dans la documentation Kubernetes.

Activer ou désactiver les journaux du plan de contrôle

Par défaut, les journaux du plan de contrôle du cluster ne sont pas envoyés à CloudWatch Logs. Vous devez activer chaque type de journal individuellement pour envoyer des journaux pour votre cluster. CloudWatch Les taux d'ingestion des journaux, de stockage des archives et d'analyse des données s'appliquent aux journaux du plan de contrôle activés. Pour en savoir plus, consultez PricingCloudWatch (Tarification).

Pour mettre à jour la configuration de la journalisation du plan de contrôle, Amazon EKS a besoin d'un maximum de cinq adresses IP disponibles dans chaque sous-réseau. Lorsque vous activez un type de journal, les journaux sont envoyés avec un niveau de détail de journal 2.

Vous pouvez activer ou désactiver les journaux du plan de contrôle à l'aide de AWS Management Console ou de l'interface CLI AWS.

AWS Management Console

1. Ouvrez la console Amazon EKS.
2. Choisissez le nom du cluster pour afficher les informations le concernant.
3. Sélectionnez l'onglet Observabilité.
4. Dans la section Journalisation du plan de contrôle, sélectionnez Gérer la journalisation.
5. Pour chaque type de journal, choisissez si celui-ci doit être activé ou désactivé. Par défaut, chaque type de journal est désactivé.
6. Choisissez Save changes (Enregistrer les modifications) pour terminer.

AWS CLI

1. Vérifiez la version de votre AWS CLI à l'aide de la commande suivante.

```
aws --version
```

   Si la version de votre AWS CLI est antérieure à`1.16.139`, vous devez d'abord effectuer la mise à jour vers la dernière version. Pour installer ou mettre à niveau la AWS CLI, reportez-vous à la section Installation de l'interface de ligne de AWS commande dans le Guide de l'utilisateur de l'interface de ligne de AWS commande.

2. Mettez à jour la configuration d'exportation du journal du plan de contrôle de votre cluster à l'aide de la commande AWS CLI suivante. Remplacez *`my-cluster`* par le nom de votre cluster et spécifiez les valeurs d'accès au point de terminaison souhaitées.

> **Note**
>
> La commande suivante envoie tous les types de journaux disponibles à CloudWatch Logs.

```
aws eks update-cluster-config \
    --region region-code \
    --name my-cluster \
    --logging '{"clusterLogging":[{"types":
["api","audit","authenticator","controllerManager","scheduler"],"enabled":true}]}'
```

   L'exemple qui suit illustre un résultat.

```
{
    "update": {
        "id": "883405c8-65c6-4758-8cee-2a7c1340a6d9",
        "status": "InProgress",
        "type": "LoggingUpdate",
        "params": [
            {
                "type": "ClusterLogging",
                "value": "{\"clusterLogging\":[{\"types\":[\"api\",\"audit\",
\"authenticator\",\"controllerManager\",\"scheduler\"],\"enabled\":true}]}"
            }
        ],
```

```
        "createdAt": 1553271814.684,
        "errors": []
    }
}
```

3. Surveillez l'état de la mise à jour de votre configuration de journal avec la commande suivante, en indiquant le nom de votre cluster et l'ID de mise à jour qui ont été renvoyés par la commande précédente. Votre mise à jour est terminée lorsqu'elle affiche l'état `Successful`.

```
aws eks describe-update \
    --region region-code\
    --name my-cluster \
    --update-id 883405c8-65c6-4758-8cee-2a7c1340a6d9
```

L'exemple qui suit illustre un résultat.

```
{
    "update": {
        "id": "883405c8-65c6-4758-8cee-2a7c1340a6d9",
        "status": "Successful",
        "type": "LoggingUpdate",
        "params": [
            {
                "type": "ClusterLogging",
                "value": "{\"clusterLogging\":[{\"types\":[\"api\",\"audit\",
\"authenticator\",\"controllerManager\",\"scheduler\"],\"enabled\":true}]}"
            }
        ],
        "createdAt": 1553271814.684,
        "errors": []
    }
}
```

Affichage des journaux de plan de contrôle de cluster

Une fois que vous avez activé l'un des types de journaux du plan de contrôle pour votre cluster Amazon EKS, vous pouvez les consulter sur la CloudWatch console.

Pour en savoir plus sur l'affichage, l'analyse et la gestion des connexions CloudWatch, consultez le guide de l'utilisateur Amazon CloudWatch Logs.

1. Ouvrez la CloudWatch console. Ce lien ouvre la console et affiche vos groupes de journaux disponibles actuels et les filtre avec le préfixe `/aws/eks`.
2. Choisissez le cluster pour lequel vous souhaitez afficher les journaux. Le format du nom de groupe de journaux est `/aws/eks/my-cluster/cluster`.
3. Choisissez le flux de journal à afficher. La liste suivante décrit le format de nom de flux du journal pour chaque type de journal.

   > **Note**
   >
   > À mesure que les données de flux de journal augmentent, les noms de flux de journal font l'objet d'une rotation. Lorsqu'il existe plusieurs flux de journaux pour un type de journal particulier, vous pouvez afficher le dernier flux en recherchant son nom avec l'heure du dernier événement (Last event time).

   - Journaux de composant de serveur d'API Kubernetes (**`api`**) : `kube-apiserver-1234567890abcdef01234567890abcde`
   - Audit (**`audit`** : `kube-apiserver-audit-1234567890abcdef01234567890abcde`
   - Authentificateur (**`authenticator`**) : `authenticator-1234567890abcdef01234567890abcde`
   - Gestionnaire de contrôleur (**`controllerManager`**) : `kube-controller-manager-1234567890abcdef01234567890abcde`
   - Planificateur (**`scheduler`**) : `kube-scheduler-1234567890abcdef01234567890abcde`
4. Parcourez les événements du flux de journaux.

   Par exemple, vous devez voir les indicateurs de serveur d'API initiaux correspondant au cluster lorsque vous consultez le haut de `kube-apiserver-1234567890abcdef01234567890abcde` .

   > **Note**
   >
   > Si vous ne voyez pas les journaux du serveur d'API au début du flux de journaux, il est probable que le fichier journal du serveur d'API a été l'objet d'une rotation sur le serveur avant d'activer la journalisation du serveur API sur le serveur. Les fichiers journaux qui font

l'objet d'une rotation avant que la journalisation du serveur d'API ne soit activée ne peuvent pas être exportés vers CloudWatch.

Toutefois, vous pouvez créer un nouveau cluster avec la même version de Kubernetes et activer la journalisation du serveur API lorsque vous créez le cluster. Les clusters avec la même version de plateforme ont les mêmes indicateurs activés, donc vos indicateurs doivent correspondre aux indicateurs du nouveau cluster. Lorsque vous avez fini de visualiser les drapeaux du nouveau cluster dans CloudWatch, vous pouvez supprimer le nouveau cluster.

Journaliser les appels d'API en tant qu'événements AWS CloudTrail

Amazon EKS est intégré à AWS CloudTrail. CloudTrail est un service qui fournit un enregistrement des actions effectuées par un utilisateur, un rôle ou un service AWS dans Amazon EKS. CloudTrail capture tous les appels d'API pour Amazon EKS en tant qu'événements. Cela inclut les appels de la console Amazon EKS et les appels de code aux opérations de l'API Amazon EKS.

Si vous créez un journal d'activité, vous pouvez activer la livraison continue d'événements CloudTrail à un compartiment Amazon S3, y compris des événements pour Amazon S3. Cela inclut les événements pour Amazon EKS. Si vous ne configurez pas de piste, vous pouvez toujours afficher les événements les plus récents dans la console CloudTrail dans Historique des événements. Grâce aux informations collectées par CloudTrail, vous pouvez déterminer plusieurs détails sur une demande. Par exemple, vous pouvez déterminer quand la demande a été envoyée à Amazon EKS, l'adresse IP à partir de laquelle la demande a été effectuée et qui a effectué la demande.

Pour en savoir plus sur CloudTrail, consultez le Guide de l'utilisateur AWS CloudTrail.

Rubriques

- Voir les références utiles pour AWS CloudTrail
- Analyser les entrées du fichier journal AWS CloudTrail
- Afficher les métriques pour les groupes Amazon EC2 Auto Scaling

Voir les références utiles pour AWS CloudTrail

Lorsque vous créez votre compte AWS, CloudTrail est également activé dans votre compte AWS. Lorsqu'une activité se produit dans Amazon EKS, cette activité est enregistrée dans un

événement CloudTrail avec d'autres événements de service AWS dans Event history (Historique des événements). Vous pouvez afficher, rechercher et télécharger les événements récents dans votre compte AWS. Pour plus d'informations, consultez Affichage des événements avec l'historique des événements CloudTrail.

Pour enregistrer en continu des événements de votre compte AWS, y compris les événements pour Amazon EKS, créez un journal d'activité. Un journal d'activité permet à CloudTrail d'envoyer des fichiers journaux dans un compartiment Amazon S3. Par défaut, lorsque vous créez un journal de suivi dans la console, il s'applique à toutes les régions AWS. Le journal d'activité consigne les événements de toutes les Régions dans la partition AWSAWS et transfère les fichiers journaux dans le compartiment Amazon S3 de votre choix. En outre, vous pouvez configurer d'autres services AWS pour analyser plus en détail les données d'événement collectées dans les journaux CloudTrail et agir en conséquence. Pour plus d'informations, veuillez consulter les ressources suivantes.

- Présentation de la création d'un journal de suivi
- Intégrations et services pris en charge par CloudTrail
- Configuration des notifications Amazon SNS pour CloudTrail
- Réception de fichiers journaux CloudTrail de plusieurs régions et Réception de fichiers journaux CloudTrail de plusieurs comptes

Toutes les actions Amazon EKS sont consignées par CloudTrail et documentées dans la référence d'API Amazon EKS. Par exemple, les appels aux sections CreateCluster, ListClusters et DeleteCluster génèrent des entrées dans les fichiers journaux CloudTrail.

Chaque événement ou entrée de journal contient des détails sur le type d'identité IAM ayant effectué la demande, ainsi que les informations d'identification qui ont été utilisées. Si des informations d'identification temporaires ont été utilisées, l'entrée montre comment ces informations d'identification ont été obtenues.

Pour plus d'informations, consultez l'élément userIdentity CloudTrail.

Analyser les entrées du fichier journal AWS CloudTrail

Un journal de suivi est une configuration qui permet d'envoyer des événements sous forme de fichiers journaux à un compartiment Amazon S3 de votre choix. Les fichiers journaux CloudTrail peuvent contenir une ou plusieurs entrées. Un événement représente une demande émise par une source et comprend des informations sur l'action demandée. Cela comprend des informations comme la date et l'heure de l'action et les paramètres de demande qui ont été utilisés. Les fichiers journaux

CloudTrail ne constituent pas une série ordonnée retraçant les appels d'API publics. Ils ne suivent aucun ordre précis.

L'exemple suivant montre une entrée de journal CloudTrail qui illustre l'action `CreateCluster`.


```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AKIAIOSFODNN7EXAMPLE",
    "arn": "arn:aws:iam::111122223333:user/username",
    "accountId": "111122223333",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "userName": "username"
  },
  "eventTime": "2018-05-28T19:16:43Z",
  "eventSource": "eks.amazonaws.com",
  "eventName": "CreateCluster",
  "awsRegion": "region-code",
  "sourceIPAddress": "205.251.233.178",
  "userAgent": "PostmanRuntime/6.4.0",
  "requestParameters": {
    "resourcesVpcConfig": {
      "subnetIds": [
        "subnet-a670c2df",
        "subnet-4f8c5004"
      ]
    },
    "roleArn": "arn:aws:iam::111122223333:role/AWSServiceRoleForAmazonEKS-
CAC1G1VH3ZKZ",
    "clusterName": "test"
  },
  "responseElements": {
    "cluster": {
      "clusterName": "test",
      "status": "CREATING",
      "createdAt": 1527535003.208,
      "certificateAuthority": {},
      "arn": "arn:aws:eks:region-code:111122223333:cluster/test",
      "roleArn": "arn:aws:iam::111122223333:role/AWSServiceRoleForAmazonEKS-
CAC1G1VH3ZKZ",
      "version": "1.10",
      "resourcesVpcConfig": {
        "securityGroupIds": [],
```

```
        "vpcId": "vpc-21277358",
        "subnetIds": [
          "subnet-a670c2df",
          "subnet-4f8c5004"
        ]
      }
    }
  },
  "requestID": "a7a0735d-62ab-11e8-9f79-81ce5b2b7d37",
  "eventID": "eab22523-174a-499c-9dd6-91e7be3ff8e3",
  "readOnly": false,
  "eventType": "AwsApiCall",
  "recipientAccountId": "111122223333"
}
```

Entrées de journal des rôles liés à un service Amazon EKS

Les rôles liés à un service Amazon EKS effectuent des appels d'API vers les ressources AWS. Des entrées de journal CloudTrail avec `username: AWSServiceRoleForAmazonEKS` et `username: AWSServiceRoleForAmazonEKSNodegroup` apparaissent pour des appels effectués par les rôles liés à un service Amazon EKS. Pour plus d'informations sur Amazon EKS et les rôles liés aux services, consultez the section called “Rôles liés à un service”.

L'exemple suivant montre une entrée du journal CloudTrail qui illustre une action `DeleteInstanceProfile` effectuée par le rôle lié à un service `AWSServiceRoleForAmazonEKSNodegroup`, noté dans le `sessionContext`.

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROA3WHGPEZ7SJ2CW55C5:EKS",
        "arn": "arn:aws:sts::111122223333:assumed-role/
AWSServiceRoleForAmazonEKSNodegroup/EKS",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROA3WHGPEZ7SJ2CW55C5",
                "arn": "arn:aws:iam::111122223333:role/aws-service-role/eks-
nodegroup.amazonaws.com/AWSServiceRoleForAmazonEKSNodegroup",
```

```
                "accountId": "111122223333",
                "userName": "AWSServiceRoleForAmazonEKSNodegroup"
            },
            "webIdFederationData": {},
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2020-02-26T00:56:33Z"
            }
        },
        "invokedBy": "eks-nodegroup.amazonaws.com"
    },
    "eventTime": "2020-02-26T00:56:34Z",
    "eventSource": "iam.amazonaws.com",
    "eventName": "DeleteInstanceProfile",
    "awsRegion": "region-code",
    "sourceIPAddress": "eks-nodegroup.amazonaws.com",
    "userAgent": "eks-nodegroup.amazonaws.com",
    "requestParameters": {
        "instanceProfileName": "eks-11111111-2222-3333-4444-abcdef123456"
    },
    "responseElements": null,
    "requestID": "11111111-2222-3333-4444-abcdef123456",
    "eventID": "11111111-2222-3333-4444-abcdef123456",
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```


Afficher les métriques pour les groupes Amazon EC2 Auto Scaling

Les groupes de nœuds gérés Amazon EKS disposent par défaut des métriques du groupe Amazon EC2 Auto Scaling, sans frais supplémentaires. Le groupe Auto Scaling envoie des données échantillonnées à Amazon CloudWatch toutes les minutes. Ces métriques peuvent être affinées par le nom des groupes Auto Scaling. Ils vous offrent une visibilité continue sur l'historique du groupe Auto Scaling qui alimente vos groupes de nœuds gérés, comme les changements de taille du groupe au fil du temps. Les métriques du groupe Auto Scaling sont disponibles dans Amazon CloudWatch ou dans la console Auto Scaling. Pour plus d'informations, consultez la section Surveiller les métriques CloudWatch pour vos groupes et instances Auto Scaling.

Grâce à la collecte des métriques du groupe Auto Scaling, vous pouvez surveiller la mise à l'échelle des groupes de nœuds gérés. Les métriques de groupe Auto Scaling indiquent les tailles minimale, maximale et souhaitée d'un groupe Auto Scaling. Vous pouvez créer une alarme si le nombre de

nœuds d'un groupe de nœuds est inférieur à la taille minimale, ce qui indique que le groupe est défectueux. Le suivi de la taille des groupes de nœuds permet également d'ajuster le nombre maximal afin que votre plan de données ne se retrouve pas à court de capacité.

Si vous préférez que ces mesures ne soient pas collectées, vous pouvez choisir de désactiver toutes ou seulement certaines d'entre elles. Par exemple, vous pouvez procéder ainsi pour éviter le bruit dans vos tableaux de bord CloudWatch. Pour plus d'informations, consultez Métriques Amazon CloudWatch pour Amazon EC2 Auto Scaling.

Envoyer des données métriques et de suivi avec l'opérateur ADOT

Amazon EKS prend en charge l'utilisation de la AWS Management Console, de l'AWS CLI et de l'API Amazon EKS pour installer et gérer l'opérateur AWS Distro for OpenTelemetry (ADOT). Cela facilite l'envoi de données métriques et de trace par vos applications exécutées sur Amazon EKS vers plusieurs options de service de surveillance telles que Amazon CloudWatch, Prometheus et X-Ray.

Pour plus d'informations, consultez Prise en main d'AWS Distro for OpenTelemetry à l'aide des modules complémentaires EKS dans la documentation AWS Distro for OpenTelemetry.

Affichage des données agrégées des ressources du cluster dans le tableau de bord EKS

image::images/eks-dashboard.png[screenshot of account level cluster metrics]

Qu'est-ce que le tableau de bord Amazon EKS◦?

Le tableau de bord Amazon EKS fournit une visibilité consolidée sur vos clusters Kubernetes dans plusieurs AWS régions et comptes. AWS Grâce à ce tableau de bord, vous pouvez :

- Suivez les clusters dont les end-of-support mises à niveau automatiques sont prévues dans les 90 prochains jours.
- Estimer les coûts du plan de contrôle EKS pour les clusters bénéficiant d'une prise en charge étendue.
- Vérifier les clusters présentant des informations nécessitant votre attention avant une mise à niveau.
- Identifier les groupes de nœuds gérés exécutant des versions spécifiques d'AMI.

- Surveiller la répartition des types de prise en charge des clusters (standard par rapport à la prise en charge étendue).

Le tableau de bord EKS s'intègre à EKS Cluster Insights pour détecter les problèmes liés à vos clusters, tels que l'utilisation de Kubernetes obsolètes. APIs Pour de plus amples informations, veuillez consulter the section called “Informations sur le cluster”.

> **Note**
>
> Le tableau de bord EKS n'est pas en temps réel ; il est mis à jour toutes les 12 heures. Pour la surveillance des clusters en temps réel, consultez Grappes de moniteurs

Comment le tableau de bord utilise-t-il AWS les Organizations ?

Le tableau de bord Amazon EKS nécessite l'intégration AWS des Organizations pour fonctionner. Il permet aux AWS Organisations de collecter en toute sécurité des informations sur les clusters entre les comptes. Cette intégration permet une gestion et une gouvernance centralisées à mesure que votre AWS infrastructure évolue.

Si l'option AWS Organizations n'est pas activée pour votre infrastructure, consultez le Guide de l'utilisateur AWS des organisations pour obtenir des instructions de configuration.

Accès interrégions et accès intercomptes

Le tableau de bord EKS peut voir les ressources du cluster dans n'importe quel compte membre de l' AWS organisation. Pour générer une liste des AWS comptes de votre organisation, voir Exporter les détails de tous les comptes d'une organisation.

La région AWS us-east-1 génère le tableau de bord. Vous devez vous connecter à cette région pour voir le tableau de bord. Le tableau de bord agrège les données entre AWS les régions, mais cela n'inclut pas les `GovCloud` régions de Chine.

Termes clés

- AWS Organisation : structure de gestion unifiée pour plusieurs AWS comptes.
- Compte de gestion : compte principal qui contrôle l' AWS organisation.
- Compte membre : n'importe quel compte au sein de l'organisation, à l'exception du compte de gestion.

- Administrateur délégué : compte membre à qui sont accordées des autorisations d'administration intercomptes. Dans le compte de gestion, vous pouvez sélectionner un compte d'administrateur délégué par AWS service.
- Accès approuvé : autorisation permettant au tableau de bord EKS d'accéder aux informations des clusters dans les comptes de l'organisation.
- Rôle lié à un service (SLRs) : type unique de rôle IAM directement lié à un service. AWS Le tableau de bord EKS utilise un SLR pour lire les informations de vos comptes et de votre organisation.
- Pour plus d'informations, reportez-vous à la section Terminologie et concepts du Guide de l'utilisateur AWS des Organizations.

Présentation générale

1. Accédez au compte de gestion de votre AWS organisation.
   - Les étapes pour accéder au compte de gestion dépendent de la manière dont vous avez configuré votre AWS organisation. Par exemple, vous pouvez accéder au compte de gestion via AWS Identity Center ou Okta.
2. Activez l'accès approuvé depuis la console EKS.
3. Désignez un administrateur délégué à l'aide de son identifiant de AWS compte.
4. Basculez vers le compte administrateur délégué.
5. Accédez à la console EKS améliorée, avec visibilité sur l'ensemble de l'organisation.

Activez le tableau de bord EKS à l'aide de la AWS console

> **Important**
>
> Vous devez être connecté au compte de gestion de votre AWS organisation pour activer le tableau de bord EKS.

Accès aux paramètres du tableau de bord EKS

1. Assurez-vous des points suivants :
   a. AWS Organizations est activée et configurée.
   b. Vous êtes connecté au compte de gestion de l'organisation.

c. Vous consultez la console de AWS gestion dans la région us-east-1.

2. Accédez à la console EKS.
3. Dans la barre latérale gauche, ouvrez Paramètres du tableau de bord.

Configuration de l'accès au tableau de bord Amazon EKS

1. Trouvez l'identifiant du AWS AWS compte que vous souhaitez autoriser à consulter le tableau de bord EKS.
   a. Cette étape est facultative, mais recommandée. Si vous ne la réalisez pas, l'accès au tableau de bord ne sera possible que depuis le compte de gestion. En tant que bonne pratique, vous devez limiter l'accès au compte de gestion.
2. Cliquez sur Activer l'accès approuvé.
   a. Vous pouvez désormais consulter le tableau de bord depuis le compte de gestion.
3. Cliquez sur Enregistrer un administrateur délégué et saisissez l'identifiant du AWS compte que vous utiliserez pour consulter le tableau de bord.
   a. Vous pouvez désormais afficher le tableau de bord depuis le compte administrateur délégué ou depuis le compte de gestion.

Pour plus d'informations sur les autorisations requises pour activer le tableau de bord, consultez Politiques IAM minimales requises.

Affichage du tableau de bord EKS

1. Connectez-vous au compte administrateur délégué (recommandé) ou au compte de gestion.
2. Connexion à la région us-east-1.
3. Accédez au service EKS, puis sélectionnez Tableau de bord dans la barre latérale gauche.

> **Note**
>
> Vérifiez les autorisations IAM requises pour consulter le tableau de bord EKS.

Configuration du tableau de bord

Vous pouvez configurer l'affichage du tableau de bord et filtrer les ressources.

Ressources disponibles

- Clusters : affichez des informations agrégées sur l'état et l'emplacement des clusters EKS.
  - Clusters présentant des problèmes d'état.
  - Clusters sous EKS Extended Support.
  - Répartition des clusters par version de Kubernetes.
- Groupes de nœuds gérés : passez en revue les groupes de nœuds gérés et EC2 les instances.
  - Répartition des groupes par type d'AMI, comme Amazon Linux ou Bottlerocket.
  - Problèmes d'état des groupes de nœuds.
  - Répartition par type d'instance.
- Modules complémentaires : informations sur les modules complémentaires Amazon EKS installés et leur état.
  - Nombre d'installations par module complémentaire.
  - Modules complémentaires présentant des problèmes d'état.
  - Répartition des versions par module complémentaire.

Vues disponibles

- Vue graphique
  - Une vue personnalisable sous forme de widgets affichant des graphiques et visualisations pour la ressource sélectionnée.
  - Les modifications apportées à la vue graphique (par exemple, suppression d'un widget) sont visibles par tous les utilisateurs du tableau de bord EKS.
- Vue Ressource
  - Vue sous forme de liste de la ressource sélectionnée, avec prise en charge des filtres.
- Vue Carte
  - Affiche la répartition géographique de la ressource sélectionnée.

Filtrage du tableau de bord EKS

Vous pouvez filtrer le tableau de bord EKS selon :

- AWS Compte
- Unité organisationnelle, définie par AWS Organizations
- AWS Région

Désactiver le tableau de bord EKS à l'aide de la AWS console

1. Assurez-vous des points suivants :
   a. AWS Organizations est activée et configurée.
   b. Vous êtes connecté au compte de gestion de l’organisation.
   c. Vous consultez la console de AWS gestion dans la région us-east-1.
2. Accédez à la console EKS.
3. Dans la barre latérale gauche, ouvrez Paramètres du tableau de bord.
4. Cliquez sur Désactiver l’accès approuvé.

Résolution des problèmes du tableau de bord EKS

Problème d’activation du tableau de bord EKS

- Vous devez être connecté au compte de gestion d'une AWS organisation.
  - Si vous n'avez pas d' AWS organisation, créez-en une. Découvrez comment Créer et configurer une organisation.
  - Si votre AWS compte est déjà membre d'une AWS organisation, identifiez l'administrateur de l'organisation.
- Vous devez être connecté au AWS compte avec des autorisations IAM suffisantes pour créer et mettre à jour les ressources AWS des Organizations.

Problème d’affichage du tableau de bord EKS

- Vous devez être connecté à l'un des AWS comptes suivants :
  - Le compte de gestion de l' AWS Organisation

  - Le compte administrateur délégué, défini dans les paramètres du tableau de bord EKS depuis le compte de gestion.
- Si vous venez d’activer le tableau de bord EKS, notez que la première collecte des données peut prendre jusqu’à 12 heures.
- Essayez de réactiver le tableau de bord à l’aide de la CLI, en vous assurant notamment de créer le rôle lié au service.

Déplacement inattendu des widgets du tableau de bord

- Le tableau de bord EKS enregistre la vue du widget configurable au niveau du AWS compte. Si vous modifiez l'affichage du widget, les autres utilisateurs du même AWS compte verront les modifications.

Configurer l'intégration du tableau de bord EKS avec les AWS Organizations

Cette section fournit des step-by-step instructions pour configurer l'intégration du tableau de bord EKS avec AWS Organizations. Vous apprendrez à activer ou désactiver l’accès approuvé entre services, ainsi qu’à enregistrer ou annuler l’enregistrement de comptes administrateurs délégués. Chaque tâche de configuration peut être exécutée à l'aide de la AWS console ou de la AWS CLI.

Permettre un accès sécurisé

L’accès approuvé autorise le tableau de bord EKS à accéder de manière sécurisée aux informations des clusters dans tous les comptes de votre organisation.

Utilisation de la AWS console

1. Connectez-vous au compte de gestion de votre AWS organisation.
2. Accédez à la console EKS dans la région us-east-1.
3. Dans la barre latérale gauche, sélectionnez Paramètres du tableau de bord.
4. Cliquez sur Activer l’accès approuvé.

> **Note**
>
> Lorsque vous activez l'accès approuvé depuis la console EKS, le système crée automatiquement le rôle lié à un service `AWSServiceRoleForAmazonEKSDashboard`. Cette création automatique ne se produit pas si vous activez l'accès sécurisé à l'aide de la AWS CLI ou de la console AWS Organizations.

Utilisation de la AWS CLI

1. Connectez-vous au compte de gestion de votre AWS organisation.
2. Exécutez les commandes suivantes :

```
aws iam create-service-linked-role --aws-service-name dashboard.eks.amazonaws.com
aws organizations enable-aws-service-access --service-principal eks.amazonaws.com
```

Désactiver l'accès approuvé

La désactivation de l'accès approuvé révoque l'autorisation du tableau de bord EKS d'accéder aux informations des clusters dans les comptes de votre organisation.

Utilisation de la AWS console

1. Connectez-vous au compte de gestion de votre AWS organisation.
2. Accédez à la console EKS dans la région us-east-1.
3. Dans la barre latérale gauche, sélectionnez Paramètres du tableau de bord.
4. Cliquez sur Désactiver l'accès approuvé.

Utilisation de la AWS CLI

1. Connectez-vous au compte de gestion de votre AWS organisation.
2. Exécutez la commande suivante :

```
aws organizations disable-aws-service-access --service-principal eks.amazonaws.com
```

Activer un compte administrateur délégué

Un administrateur délégué est un compte membre autorisé à accéder au tableau de bord EKS.

Utilisation de la AWS console

1. Connectez-vous au compte de gestion de votre AWS organisation.
2. Accédez à la console EKS dans la région us-east-1.
3. Dans la barre latérale gauche, sélectionnez Paramètres du tableau de bord.
4. Cliquez sur Enregistrer un administrateur délégué.
5. Entrez l'ID de compte du AWS compte que vous souhaitez choisir comme administrateur délégué.
6. Confirmez l’enregistrement.

Utilisation de la AWS CLI

1. Connectez-vous au compte de gestion de votre AWS organisation.
2. Exécutez la commande suivante, en remplaçant `123456789012` par votre ID de compte :

```
aws organizations register-delegated-administrator --account-id 123456789012 --
service-principal eks.amazonaws.com
```

Désactivation d’un compte administrateur délégué

La désactivation d’un administrateur délégué supprime l’autorisation de ce compte d’accéder au tableau de bord EKS.

Utilisation de la AWS console

1. Connectez-vous au compte de gestion de votre AWS organisation.
2. Accédez à la console EKS dans la région us-east-1.
3. Dans la barre latérale gauche, sélectionnez Paramètres du tableau de bord.
4. Localisez l’administrateur délégué dans la liste.
5. Cliquez sur Désenregistrer à côté du compte que vous souhaitez supprimer en tant qu’administrateur délégué.

Utilisation de la AWS CLI

1. Connectez-vous au compte de gestion de votre AWS organisation.
2. Exécutez ensuite la commande suivante, en remplaçant `123456789012` par l'ID du compte de l'administrateur délégué :

```
aws organizations deregister-delegated-administrator --account-id 123456789012 --
service-principal eks.amazonaws.com
```

Politiques IAM minimales nécessaires

Cette section décrit les politiques IAM minimales requises pour permettre un accès sécurisé et déléguer un administrateur pour l'intégration du tableau de bord EKS avec les AWS Organizations.

Politique d'activation de l'accès approuvé

Pour activer un accès sécurisé entre EKS Dashboard et AWS Organizations, vous devez disposer des autorisations suivantes :

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "organizations:EnableAWSServiceAccess",
                "organizations:DescribeOrganization",
                "organizations:ListAWSServiceAccessForOrganization"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": "arn:aws:iam::*:role/aws-service-role/
dashboard.eks.amazonaws.com/AWSServiceRoleForAmazonEKSDashboard"
        }
    ]
}
```

Politique de délégation d'un administrateur

Pour enregistrer ou désenregistrer un administrateur délégué pour le tableau de bord EKS, vous devez disposer des autorisations suivantes :

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "organizations:RegisterDelegatedAdministrator",
                "organizations:DeregisterDelegatedAdministrator",
                "organizations:ListDelegatedAdministrators"
            ],
            "Resource": "*"
        }
    ]
}
```

Politique d'accès au tableau de bord EKS

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AmazonEKSDashboardReadOnly",
            "Effect": "Allow",
            "Action": [
                "eks:ListDashboardData",
                "eks:ListDashboardResources",
                "eks:DescribeClusterVersions"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AmazonOrganizationsReadOnly",
            "Effect": "Allow",
            "Action": [
                "organizations:DescribeOrganization",
                "organizations:ListAWSServiceAccessForOrganization",
                "organizations:ListRoots",
                "organizations:ListAccountsForParent",
```

```
                "organizations:ListOrganizationalUnitsForParent"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AmazonOrganizationsDelegatedAdmin",
            "Effect": "Allow",
            "Action": [
                "organizations:ListDelegatedAdministrators"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringEquals": {
                    "organizations:ServicePrincipal": "eks.amazonaws.com"
                }
            }
        }
    ]
}
```

> **Note**
>
> Ces politiques doivent être associées au principal IAM (utilisateur ou rôle) dans le compte de gestion de votre AWS organisation. Les comptes membres ne peuvent pas activer l'accès approuvé ni désigner d'administrateurs délégués.

Améliorez EKS avec des AWS services intégrés

Outre les services décrits dans d'autres sections, Amazon EKS propose d'autres AWS services pour fournir des solutions supplémentaires. Cette rubrique identifie certains des autres services qui utilisent Amazon EKS pour ajouter des fonctionnalités, ou des services qu'Amazon EKS utilise pour effectuer des tâches.

Rubriques

- Sauvegardez vos clusters EKS avec AWS Backup
- Créer des ressources Amazon EKS avec CloudFormation AWS
- Connectez-vous aux référentiels Git avec AWS CodeConnections
- Analysez les événements de sécurité sur EKS avec Amazon Detective
- Détectez les menaces avec Amazon GuardDuty
- Lancer des clusters EKS à faible latence avec AWS Local Zones
- Évaluez la résilience du cluster EKS avec le Hub de Résilience AWS
- Gérez les secrets des applications avec AWS Secrets Manager
- Centralisez et analysez les données de sécurité EKS avec Security Lake.
- Activez une connectivité inter-clusters sécurisée avec Amazon VPC Lattice.

Sauvegardez vos clusters EKS avec AWS Backup

AWS Backup prend en charge les sauvegardes des clusters Amazon EKS, y compris l'état du cluster Kubernetes et le stockage persistant attaché au cluster EKS via une réclamation de volume persistant (volumes EBS, systèmes de fichiers EFS et compartiments S3). Une sauvegarde Amazon EKS créera un point de restauration composite, où un point de restauration secondaire sera créé pour chaque ressource sauvegardée.

- Comment sauvegarder des ressources : Premiers pas avec AWS Backup
- Création de sauvegardes par type de ressource : Amazon EKS Backups
- Comment restaurer des clusters Amazon EKS : restauration des sauvegardes Amazon EKS

Pour démarrer via la console EKS, la console AWS Backup ou la CLI, assurez-vous que votre rôle IAM dispose des autorisations suivantes :

- Vous pouvez les trouver dans la politique gérée de AWS Backup AWSBackupServiceRolePolicyForBackup. Il contient les autorisations requises pour sauvegarder votre cluster Amazon EKS et votre stockage persistant EBS et EFS
- Si votre cluster EKS contient un compartiment S3, vous devez vous assurer que les politiques et conditions préalables suivantes pour votre compartiment S3 sont ajoutées et activées, comme indiqué dans la documentation :
- AWSBackupServiceRolePolicyForSauvegarde S3
- Conditions préalables pour les sauvegardes S3

Assurez-vous que vos clusters EKS possèdent les paramètres suivants :

- Le mode d'autorisation du cluster EKS est défini sur API ou API_AND _CONFIG_MAP pour AWS Backup afin de créer des entrées d'accès permettant d'accéder au cluster EKS.

Créer des ressources Amazon EKS avec CloudFormation AWS

Amazon EKS est intégré à CloudFormation AWS, un service qui vous aide à modéliser et à configurer vos ressources AWS afin que vous puissiez passer moins de temps à créer et à gérer vos ressources et votre infrastructure. Vous créez un modèle qui décrit toutes les ressources AWS dont vous avez besoin, par exemple un cluster Amazon EKS, et CloudFormation AWS se charge de provisionner et de configurer ces ressources pour vous.

Lorsque vous utilisez CloudFormation AWS, vous pouvez réutiliser votre modèle pour configurer vos ressources Amazon EKS de manière cohérente et répétée. Il suffit de décrire vos ressources une fois, puis d'allouer les mêmes ressources à l'infini dans plusieurs comptes et régions AWS.

Amazon EKS et modèles CloudFormation AWS

Pour provisionner et configurer des ressources pour Amazon EKS et les services associés, vous devez comprendre les modèles CloudFormation AWS. Les modèles sont des fichiers texte formatés en JSON ou YAML. Ces modèles décrivent les ressources que vous souhaitez provisionner dans vos piles CloudFormation AWS. Si vous n'êtes pas familier avec JSON ou YAML, vous pouvez utiliser CloudFormation Designer AWS pour vous aider à démarrer avec les modèles CloudFormation AWS. Pour plus d'informations, consultez la section Qu'est-ce que CloudFormation Designer ?AWS dans le Guide de l'utilisateur CloudFormation AWS.

Amazon EKS prend en charge la création de clusters et de groupes de nœuds dans CloudFormation AWS. Pour plus d'informations, y compris des exemples de modèles JSON et YAML pour vos ressources Amazon EKS, consultez la référence sur les types de ressources Amazon EKS dans le Guide de l'utilisateur CloudFormation AWS.

En savoir plus sur CloudFormation AWS

Pour en savoir plus sur CloudFormation AWS, consultez les ressources suivantes :

- AWS CloudFormation
- AWS Guide de l'utilisateur CloudFormation
- Guide de l'utilisateur de l'interface de ligne de commande CloudFormation AWS

Connectez-vous aux référentiels Git avec AWS CodeConnections

AWS CodeConnectionsfournit un moyen sécurisé de connecter AWS des services à des référentiels de code source tiers. AWS CodeConnections prend en charge GitHub GitLab, Bitbucket et d'autres fournisseurs. Pour en savoir plus et commencer, consultez la section Utilisation des connexions.

À utiliser AWS CodeConnections avec Argo CD

Lorsque vous utilisez la fonctionnalité EKS pour Argo CD, vous pouvez choisir d'activer l'authentification sécurisée auprès AWS CodeConnections des référentiels Git sans gérer les informations d'identification à long terme ou les jetons d'accès personnels. AWS CodeConnections gère le flux OAuth d'authentification et gère la connexion à votre fournisseur Git, en fournissant une approche sécurisée et gérable pour accéder à vos GitOps référentiels et manifestes d'applications stockés chez des fournisseurs Git tiers.

Conditions préalables

- Création d'un cluster Amazon EKS doté de la fonctionnalité Argo CD
- Une connexion créée dans AWS CodeConnections votre fournisseur Git
- Autorisations IAM configurées pour qu'Argo CD utilise la connexion

CodeConnections Pour configurer l'accès au référentiel Argo CD

1. Créez une connexion dans la CodeConnections console :

a. Ouvrez la CodeConnections console.

b. Choisissez Créer une connexion.

c. Sélectionnez votre fournisseur (GitHub, GitLab, ou Bitbucket) et suivez le flux d'authentification.

d. Notez l'ARN de connexion à utiliser dans la configuration de votre Argo CD.

2. Assurez-vous que le rôle de fonctionnalité Argo CD est autorisé à utiliser la connexion avec une politique basée sur les ressources :

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codeconnections:UseConnection",
        "codeconnections:GetConnection"
      ],
      "Resource": "arn:aws:codeconnections:region:account-id:connection/connection-
id"
    }
  ]
}
```

3. Configurez Argo CD pour référencer la CodeConnections ressource lors de l'ajout d'un référentiel, en utilisant le point de terminaison de la CodeConnections ressource comme URL du référentiel. La fonctionnalité Argo CD utilise la connexion pour s'authentifier auprès de votre fournisseur Git sans avoir besoin d'informations d'identification de longue durée.

Considérations relatives à l'utilisation CodeConnections avec Argo CD

Lorsque vous utilisez AWS CodeConnections le CD EKS Capability for Argo, gardez à l'esprit les points suivants :

- La CodeConnections connexion doit se trouver dans la même AWS région que votre cluster EKS
- Le rôle de capacité Argo CD doit comporter `codeconnections:UseConnection` des autorisations `codeconnections:GetConnection`
- CodeConnections gère automatiquement le OAuth flux et le cycle de vie des informations d'identification

Pour plus d'informations sur la configuration de l'accès au référentiel avec Argo CD, consultezthe section called “Configuration des référentiels”.

Analysez les événements de sécurité sur EKS avec Amazon Detective

Amazon Detective vous permet d'analyser, d'enquêter et d'identifier rapidement la cause racine des résultats de sécurité ou des activités suspectes. Detective collecte automatiquement les données du journal à partir de vos AWS ressources. Detective utilise ensuite le machine learning, l'analyse statistique et la théorie des graphes pour générer des visualisations qui vous aideront à mener des investigations de sécurité plus rapides et plus efficaces. Les agrégations de données, les résumés et le contexte prédéfinis de Detective vous aident à analyser et à déterminer rapidement la nature et l'étendue des éventuels problèmes de sécurité. Pour plus d'informations, consultez le Guide de l'utilisateur Amazon Detective.

Detective organise Kubernetes et les AWS données en résultats tels que :

- Les détails du cluster Amazon EKS, y compris l'identité IAM qui a créé le cluster et la fonction du service du cluster. Vous pouvez étudier l'activité de ces identités IAM AWS et l'activité de l'API Kubernetes avec Detective.
- Les détails du conteneur, tels que l'image et le contexte de sécurité. Vous pouvez également examiner les détails des pods terminés.
- Vous pouvez examiner l'activité de l'API Kubernetes de ces identités IAM à l'aide de Detective. Par exemple, vous pouvez afficher le nombre d'appels API Kubernetes réussis et échoués qui ont été émis pendant une période sélectionnée. En outre, la section sur les appels d'API récemment observés peut être utile pour identifier une activité suspecte.

Les journaux d'audit Amazon EKS sont un package de source de données facultatif qui peut être ajouté à votre graphe de comportement Detective. Vous pouvez afficher les packages source facultatifs disponibles et leur état dans votre compte. Pour plus d'informations, veuillez consulter la rubrique Journaux d'audit d'Amazon EKS pour Detective dans le Guide de l'utilisateur Amazon Detective.

Utilisation d'Amazon Detective avec Amazon EKS

Avant de pouvoir examiner les résultats, Detective doit être activé pendant au moins 48 heures dans la même AWS région que celle dans laquelle se trouve votre cluster. Pour plus d'informations,

veuillez consulter la rubrique Configuration d'Amazon Detective dans le Guide de l'utilisateur Amazon Detective.

1. Ouvrez la console Detective à l'adresse https://console.aws.amazon.com/detective/.
2. Dans le volet de navigation de gauche, sélectionnez Search (Rechercher).
3. Sélectionnez Choose type (Choisir le type), puis sélectionnez EKS cluster.
4. Saisissez le nom du cluster ou l'ARN, puis choisissez Search (Rechercher).
5. Dans les résultats de la recherche, sélectionnez le nom du cluster dont vous souhaitez consulter l'activité. Pour plus d'informations sur ce que vous pouvez consulter, consultez la section Activité globale de l'API Kubernetes impliquant un cluster Amazon EKS dans le Guide de l'utilisateur Amazon Detective.

Détectez les menaces avec Amazon GuardDuty

Amazon GuardDuty est un service de détection des menaces qui vous aide à protéger vos comptes, vos conteneurs, vos charges de travail et les données de votre environnement AWS. À l'aide de modèles de machine learning (ML) et de capacités de détection des anomalies et des menaces, GuardDuty surveille en permanence les différentes sources de journaux et les activités d'exécution afin d'identifier et de hiérarchiser les risques de sécurité potentiels et les activités malveillantes dans votre environnement.

Entre autres fonctionnalités, GuardDuty offre les deux fonctionnalités suivantes qui détectent les menaces potentielles pour vos clusters EKS : Protection EKS et Surveillance du temps d'exécution.

> **Note**
>
> Nouveau : le mode automatique Amazon EKS s'intègre à GuardDuty.

EKS Protection

Cette fonctionnalité offre une couverture de détection des menaces pour vous aider à protéger les clusters Amazon EKS en surveillant les journaux d'audit Kubernetes associés. Les journaux d'audit Kubernetes enregistrent les actions séquentielles au sein de votre cluster, y compris les activités des utilisateurs, des applications utilisant l'API Kubernetes et du plan de contrôle. Par exemple, GuardDuty peut identifier que des API appelées pour potentiellement altérer des ressources dans un cluster Kubernetes ont été invoquées par un utilisateur non authentifié.

Lorsque vous activez EKS Protection, GuardDuty pourra accéder à vos journaux d'audit Amazon EKS uniquement à des fins de détection continue des menaces. Si GuardDuty identifie une menace potentielle pour votre cluster, il génère un résultat de journal d'audit Kubernetes associé d'un type spécifique. Pour plus d'informations sur les types de résultats disponibles à partir des journaux d'audit Kubernetes, consultez la section Types de résultats des journaux d'audit Kubernetes dans le Guide de l'utilisateur Amazon GuardDuty.

Pour plus d'informations, consultez la section Protection EKS dans le Guide de l'utilisateur Amazon GuardDuty.

Surveillance d'exécution

Cette fonctionnalité surveille et analyse les événements au niveau du système d'exploitation, du réseau et des fichiers afin de vous aider à détecter les menaces potentielles dans des charges de travail AWS spécifiques de votre environnement.

Lorsque vous activez la surveillance d'exécution et installez l'agent GuardDuty dans vos clusters Amazon EKS, GuardDuty commence à surveiller les événements d'exécution associés à ce cluster. Notez que l'agent GuardDuty et la surveillance d'exécution ne sont pas disponibles pour les nœuds hybrides Amazon EKS. La surveillance d'exécution n'est donc pas disponible pour les événements d'exécution qui se produisent sur vos nœuds hybrides. Si GuardDuty identifie une menace potentielle pour votre cluster, il génère un résultat associé dans la surveillance d'exécution. Par exemple, une menace peut potentiellement commencer par compromettre un seul conteneur qui exécute une application web vulnérable. Cette application Web peut disposer d'autorisations d'accès aux conteneurs et charges de travail sous-jacents. Dans ce scénario, des identifiants mal configurés pourraient potentiellement conduire à un accès plus large au compte et aux données qui y sont stockées.

Pour configurer la surveillance d'exécution, vous installez l'agent GuardDuty sur votre cluster en tant que module complémentaire Amazon EKS. Pour plus d'informations sur les modules complémentaires, consultez the section called " AWS modules complémentaires".

Pour plus d'informations, consultez la section Surveillance d'exécution dans le Guide de l'utilisateur Amazon GuardDuty.

Lancer des clusters EKS à faible latence avec AWS Local Zones

Une AWS Local Zone est une extension d'une région AWS géographiquement proche de vos utilisateurs. Les Local Zones disposent de leurs propres connexions à Internet et prennent en charge

AWS Direct Connect. Les ressources créées dans une zone locale peuvent servir les utilisateurs locaux avec des communications à faible latence. Pour plus d'informations, consultez le Guide d'utilisation AWS Local Zones et la section Local Zones du Guide de l'utilisateur Amazon EC2.

Amazon EKS prend en charge certaines ressources dans Local Zones. Cela inclut les groupes de nœuds gérés, les nœuds Amazon EC2 autogérés, les volumes Amazon EBS et les Application Load Balancer (ALB). Nous vous recommandons de tenir compte des points suivants lorsque vous utilisez Local Zones dans le cadre de votre cluster Amazon EKS.

- Vous ne pouvez pas créer de nœuds Fargate dans les zones locales avec Amazon EKS.
- Le plan de contrôle Kubernetes géré par Amazon EKS s'exécute toujours dans la région AWS. Le plan de contrôle Kubernetes géré par Amazon EKS ne peut pas fonctionner dans la zone locale. Étant donné que les Local Zones apparaissent comme un sous-réseau au sein de votre VPC, Kubernetes voit vos ressources de zone locale comme faisant partie de ce sous-réseau.
- Le cluster Amazon EKS Kubernetes communique avec les instances Amazon EC2 que vous exécutez dans la région ou la Local Zone AWS en utilisant les interfaces de réseau Elastic gérées par Amazon EKS Pour en savoir plus sur l'architecture réseau Amazon EKS, consultez Configuration de la mise en réseau.
- Contrairement aux sous-réseaux régionaux, Amazon EKS ne peut pas placer d'interfaces réseau dans vos sous-réseaux Local Zone. Cela signifie que vous ne devez pas spécifier de sous-réseaux de Local Zone lorsque vous créez votre cluster. Cependant, vous pouvez avoir des composants master dans plusieurs Local Zones différentes connectées au même cluster.

Évaluez la résilience du cluster EKS avec le Hub de Résilience AWS

Le Hub de Résilience AWS évalue la résilience d'un cluster Amazon EKS en analysant son infrastructure AWS. Hub de Résilience utilise la configuration du contrôle d'accès basé sur les rôles (RBAC) de Kubernetes pour évaluer les charges de travail Kubernetes déployées sur votre cluster. Pour plus d'informations, consultez la section Activation de l'accès du Hub de Résilience AWS à votre cluster Amazon EKS dans le Guide de l'utilisateur du Hub de Résilience AWS.

Gérez les secrets des applications avec AWS Secrets Manager

AWS Secrets Manager vous permet de gérer, d'accéder et de faire pivoter les informations d'identification, les clés d'API et les autres secrets tout au long de leur cycle de vie. Avec Secrets

Manager, vous pouvez sécuriser et gérer les secrets utilisés pour accéder aux ressources dans AWS, sur des services tiers et sur site. Pour plus d'informations, consultez le guide de l'utilisateur AWS Secrets Manager.

Lorsque vous utilisez la fonctionnalité EKS pour Argo CD, Secrets Manager fournit un moyen sécurisé de stocker et de récupérer les informations d'identification du référentiel Git sans coder en dur des données sensibles dans la configuration et les ressources de votre Argo CD. Cette intégration est particulièrement utile pour gérer les jetons d'accès aux référentiels privés et les clés SSH utilisés par Argo CD pour synchroniser les applications depuis les référentiels Git.

Utiliser AWS Secrets Manager avec Argo CD

Lorsque vous utilisez la fonctionnalité EKS pour Argo CD, vous pouvez stocker les informations d'identification du référentiel Git dans Secrets Manager et configurer Argo CD pour les récupérer. Cette approche est plus sûre que le stockage des informations d'identification directement dans la configuration d'Argo CD ou que l'utilisation de jetons d'accès personnels à longue durée de vie.

Conditions préalables

- Un cluster Amazon EKS avec la fonctionnalité Argo CD activée
- Informations d'identification du référentiel Git stockées dans AWS Secrets Manager
- Autorisations IAM configurées pour qu'Argo CD accède à Secrets Manager

Pour configurer Argo CD afin d'utiliser Secrets Manager pour les informations d'identification du référentiel

1. Stockez vos identifiants Git dans Secrets Manager. Par exemple, pour stocker un jeton d'accès GitHub personnel :

```
aws secretsmanager create-secret \
  --name argocd/github-token \
  --secret-string '{"username":"git","password":"ghp_xxxxxxxxxxxx"}'
```

2. Assurez-vous que le rôle de capacité Argo CD est autorisé à récupérer le secret :

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

```
      "Effect": "Allow",
      "Action": [
        "secretsmanager:GetSecretValue",
        "secretsmanager:DescribeSecret"
      ],
      "Resource": "arn:aws:secretsmanager:region:account-id:secret:argocd/github-
token*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "kms:Decrypt"
      ],
      "Resource": "arn:aws:kms:region:account-id:key/*",
      "Condition": {
        "StringLike": {
          "kms:EncryptionContext:SecretARN": "arn:aws:secretsmanager:region:account-
id:secret:argocd/*",
          "kms:ViaService": "secretsmanager.*.amazonaws.com"
        }
      }
    }
  ]
}
```

> **Note**
>
> L'autorisation de déchiffrement KMS est requise car Secrets Manager chiffre tous les secrets à l'aide AWS de KMS. La condition limite le déchiffrement aux seuls secrets avec le `argocd/` préfixe. Si vous utilisez la clé AWS gérée par défaut pour Secrets Manager, cette autorisation est suffisante. Pour les clés KMS gérées par le client, mettez à jour le `Resource` champ avec l'ARN de votre clé spécifique.

3. Configurez Argo CD pour utiliser les informations d'identification de Secrets Manager. Pour plus d'informations sur la synchronisation des secrets de Secrets Manager vers des secrets Kubernetes auxquels Argo CD peut faire référence, consultez la section Gestion des secrets dans la documentation d'Argo CD.
4. Créez une configuration de référentiel Argo CD qui fait référence à l'ARN secret :

```
apiVersion: v1
kind: Secret
```

```
metadata:
  name: private-repo
  namespace: argocd
  labels:
    argocd.argoproj.io/secret-type: repository
stringData:
  type: git
  url: https://github.com/org/repo
  secretArn: arn:aws:secretsmanager:region:account-id:secret:argocd/github-token
```

Pour plus d'informations sur la configuration de l'accès au référentiel avec Argo CD, consultezthe section called “Configuration des référentiels”.

Centralisez et analysez les données de sécurité EKS avec Security Lake.

Amazon Security Lake est un service de lac de données de sécurité entièrement géré qui vous permet de centraliser les données de sécurité provenant de diverses sources, y compris Amazon EKS. En intégrant Amazon EKS à Security Lake, vous pouvez obtenir des informations plus détaillées sur les activités effectuées sur vos ressources Kubernetes et renforcer la sécurité de vos clusters Amazon EKS.

> **Note**
>
> Pour plus d’informations sur l’utilisation de Security Lake avec Amazon EKS et la configuration des sources de données, consultez la documentation Amazon Security Lake.

Avantages de l’utilisation de Security Lake avec Amazon Amazon EKS

Données de sécurité centralisées : Security Lake collecte et centralise automatiquement les données de sécurité de vos clusters Amazon EKS, ainsi que les données provenant d'autres AWS services, de fournisseurs de SaaS, de sources sur site et de sources tierces. Cela fournit une vue complète de votre état de sécurité dans l’ensemble de votre organisation dans.

Format de données normalisé : Security Lake convertit les données collectées au format Open Cybersecurity Schema Framework (OCSF), qui est un schéma open source standard. Cette normalisation facilite l’analyse et l’intégration avec d’autres outils et services de sécurité.

Détection améliorée des menaces : En analysant les données de sécurité centralisées, y compris les journaux du plan de contrôle Amazon EKS, vous pouvez détecter plus efficacement les activités potentiellement suspectes au sein de vos clusters Amazon EKS. Cela permet d'identifier les incidents de sécurité et d'y répondre rapidement.

Gestion simplifiée des données : Security Lake gère le cycle de vie de vos données de sécurité grâce à des paramètres de conservation et de réplication personnalisables. Cela simplifie les tâches de gestion des données et garantit que vous conservez les données nécessaires à des fins de conformité et d'audit.

Activation de Security Lake pour Amazon EKS

1. Activez la journalisation de plan de contrôle Amazon EKS pour vos clusters EKS. Reportez-vous à la section Activation et désactivation des journaux du plan de contrôle pour obtenir des instructions détaillées.
2. Ajoutez les journaux d'audit Amazon EKS en tant que source dans Security Lake. Security Lake commencera alors à collecter des informations détaillées sur les activités effectuées sur les ressources Kubernetes exécutées dans vos clusters EKS.
3. Configurez les paramètres de conservation et de réplication de vos données de sécurité dans Security Lake en fonction de vos besoins.
4. Utilisez les données OCSF normalisées stockées dans Security Lake pour la réponse aux incidents, les analyses de sécurité et l'intégration avec d'autres AWS services ou outils tiers. Par exemple, vous pouvez générer des informations de sécurité à partir des données Amazon Security Lake à l'aide d'Amazon OpenSearch Ingestion.

Analyse des journaux EKS dans Security Lake

Security Lake normalise les événements de journal EKS au format OCSF, ce qui facilite l'analyse et la corrélation des données avec d'autres événements de sécurité. Vous pouvez utiliser différents outils et services, tels qu'Amazon Athena, Amazon QuickSight ou des outils d'analyse de sécurité tiers, pour interroger et visualiser les données normalisées.

Pour plus d'informations sur le mappage OCSF pour les événements du journal EKS, consultez les journaux d'audit https://github.com/ocsf/examples/tree/main/mappings/markdown/AWS/v1.1.0/EKS [référence de mappage] dans le référentiel OCSF. GitHub

Activez une connectivité inter-clusters sécurisée avec Amazon VPC Lattice.

Amazon VPC Lattice est un service de mise en réseau d'applications entièrement géré intégré directement à l'infrastructure réseau AWS que vous pouvez utiliser pour connecter, sécuriser et surveiller vos services sur plusieurs comptes et clouds privés virtuels (VPC). Avec Amazon EKS, vous pouvez tirer parti d'Amazon VPC Lattice en utilisant le Contrôleur d'API Gateway AWS, une implémentation de l'API Gateway Kubernetes. Grâce à Amazon VPC Lattice, vous pouvez configurer une connectivité inter-clusters avec la sémantique Kubernetes standard de manière simple et cohérente. Pour commencer à utiliser Amazon VPC Lattice avec Amazon EKS, consultez le Guide de l'utilisateur du contrôleur d'API Gateway AWS.

Connexion d'un cluster Kubernetes à une console de gestion Amazon EKS à l'aide du connecteur Amazon EKS

Vous pouvez utiliser Amazon EKS Connector pour enregistrer et connecter tout cluster Kubernetes conforme à AWS et le visualiser dans la console Amazon EKS. Une fois qu'un cluster est connecté, vous pouvez voir le statut, la configuration et les charges de travail de ce cluster dans la console Amazon EKS. Vous pouvez utiliser cette fonction pour afficher les clusters connectés dans la console Amazon EKS, mais vous ne pouvez pas les gérer. Le connecteur Amazon EKS est également un agent qui est un projet open source sur Github. Pour obtenir du contenu technique supplémentaire, y compris les questions fréquentes et le dépannage, veuillez consulter the section called "Dépannage d'EKS Connector".

Le connecteur Amazon EKS peut connecter les types de clusters Kubernetes suivants à Amazon EKS.

- Clusters Kubernetes sur site
- Clusters autogérés qui s'exécutent sur Amazon EC2
- Clusters gérés par d'autres fournisseurs de cloud

Considérations relatives à Amazon EKS Connector

Avant d'utiliser Amazon EKS Connector, comprenez les points suivants :

- Vous devez disposer des privilèges d'administration sur le cluster Kubernetes pour connecter le cluster à Amazon EKS.
- Le cluster Kubernetes doit comporter des composants master Linux 64 bits (x86) avant de se connecter. Les composants master ARM ne sont pas pris en charge.
- Vous devez disposer de composants master dans votre cluster Kubernetes qui disposent d'un accès sortant aux points de terminaison Systems Manager `ssm.` et `ssmmessages.`. Pour plus d'informations, consultez la section Points de terminaison Systems Manager dans référence générale AWS.
- Par défaut, vous pouvez connecter jusqu'à 10 clusters dans une région. Vous pouvez demander une augmentation par le biais de la console de quota de service. Consultez Demander une augmentation de quota pour plus d'informations.

- Seules les API `RegisterCluster`, `ListClusters`, `DescribeCluster` et `DeregisterCluster` Amazon EKS sont prises en charge pour les clusters Kubernetes externes.
- Vous devez disposer des autorisations suivantes pour enregistrer un cluster :
  - eks:RegisterCluster
  - ssm:CreateActivation
  - ssm:DeleteActivation
  - iam:PassRole
- Vous devez disposer des autorisations suivantes pour annuler l'enregistrement d'un cluster :
  - eks:DeregisterCluster
  - ssm:DeleteActivation
  - ssm:DeregisterManagedInstance

Rôles IAM requis pour Amazon EKS Connector

L'utilisation du Amazon EKS Connector nécessite les deux rôles IAM suivants :

- Le rôle lié à un service du Amazon EKS Connector est créé lorsque vous enregistrez un cluster pour la première fois.
- Vous devez créer le rôle IAM de l'agent Amazon EKS Connector. Consultez the section called "Rôle IAM du connecteur" pour plus de détails.

Pour activer l'autorisation d'affichage du cluster et de la charge de travail pour des principaux IAM, appliquez le `eks-connector` et les rôles de cluster Amazon EKS Connector à votre cluster. Suivez les étapes décrites dans Accorder l'accès pour afficher les ressources du cluster Kubernetes sur une console Amazon EKS.

Connectez un cluster Kubernetes externe à la console de gestion Amazon EKS

Vous pouvez connecter un cluster Kubernetes externe à Amazon EKS à l'aide de plusieurs méthodes dans le processus suivant. Ce processus comporte deux étapes : enregistrement du cluster auprès d'Amazon EKS et installation de l'agent `eks-connector` dans le cluster.

> **Important**
>
> Vous devez effectuer la deuxième étape dans les 3 jours suivant la fin de la première étape, avant l'expiration de l'enregistrement.

Considérations

Vous pouvez utiliser des manifestes YAML lors de l'installation de l'agent. Vous pouvez également utiliser Helm si vous enregistrez le cluster avec l'interface de ligne de commande AWS Management Console ou AWS. Cependant, vous ne pouvez pas utiliser Helm pour installer l'agent si vous enregistrez le cluster avec `eksctl`.

Prérequis

- Assurez-vous que le rôle d'agent Amazon EKS Connector est créé. Suivez les étapes décrites dans la section Création du rôle d'agent du connecteur Amazon EKS.
- Vous devez disposer des autorisations suivantes pour enregistrer un cluster :
  - `eks:RegisterCluster`
  - `ssm:CreateActivation`
  - `ssm:DeleteActivation`
  - `iam:PassRole`

Étape 1 : Enregistrement du cluster

Pour enregistrer un cluster sur le connecteur Amazon EKS, vous pouvez utiliser l'un des outils suivants :

- the section called "AWS CLI"
- the section called "AWS Management Console"
- the section called "`eksctl`"

AWS CLI

1. La CLI AWS doit être installé. Pour l'installer ou la mettre à niveau, consultez Installation de la CLI AWS.

2. Pour la configuration du connecteur, spécifiez votre rôle IAM d'agent Amazon EKS Connector. Pour de plus amples informations, consultez the section called "Rôles IAM requis pour Amazon EKS Connector".

```
aws eks register-cluster \
     --name my-first-registered-cluster \
     --connector-config roleArn=arn:aws:iam::111122223333:role/
AmazonEKSConnectorAgentRole,provider="OTHER" \
     --region aws-region
```

L'exemple qui suit illustre un résultat.

```
{
    "cluster": {
        "name": "my-first-registered-cluster",
        "arn": "arn:aws:eks:region:111122223333:cluster/my-first-registered-cluster",
        "createdAt": 1627669203.531,
        "ConnectorConfig": {
            "activationId": "xxxxxxxxACTIVATION_IDxxxxxxxx",
            "activationCode": "xxxxxxxxACTIVATION_CODExxxxxxxx",
            "activationExpiry": 1627672543.0,
            "provider": "OTHER",
            "roleArn": "arn:aws:iam::111122223333:role/AmazonEKSConnectorAgentRole"
        },
        "status": "CREATING"
    }
}
```

Vous utilisez les valeurs `aws-region`, `activationId` et `activationCode` à l'étape suivante.

AWS Management Console

1. Ouvrez la console Amazon EKS.
2. Sélectionnez Add cluster (Ajouter un cluster), puis Register (Enregistrer) pour afficher la page de configuration.
3. Sur la page Configurer le cluster, renseignez les champs suivants :
   - Nom : nom unique de votre cluster.

- Provider (Fournisseur) : sélectionnez cet élément pour afficher la liste déroulante des fournisseurs de cluster Kubernetes. Si vous ne connaissez pas le fournisseur spécifique, sélectionnez Autre.
- Rôle EKS Connector : sélectionnez le rôle à utiliser pour connecter le cluster.

4. Sélectionnez Register cluster (Enregistrer le cluster).
5. La page Présentation du cluster s'affiche. Si vous souhaitez utiliser les Charts de Helm, copiez la commande `helm install` et passez à l'étape suivante. Si vous souhaitez utiliser le manifeste YAML, sélectionnez Télécharger le fichier YAML pour télécharger le fichier manifeste sur votre lecteur local.

   > **Important**
   >
   > C'est le seul moment où vous pouvez copier la commande `helm install` ou télécharger ce fichier. Ne quittez pas cette page, car le lien ne sera plus accessible et vous devrez désenregistrer le cluster et recommencer la procédure depuis le début.

   La commande ou le fichier manifeste ne peut être utilisé(e) qu'une seule fois pour le cluster enregistré. Si vous supprimez des ressources du cluster Kubernetes, vous devez réenregistrer le cluster et obtenir un nouveau fichier manifeste.

Passez à l'étape suivante pour appliquer le fichier manifeste à votre cluster Kubernetes.

`eksctl`

1. `eksctl` version `0.68` ou une version ultérieure doit être installée. Pour l'installer ou la mettre à niveau, veuillez consulter la rubrique the section called “Créer un cluster (eksctl)”.
2. Enregistrez le cluster en fournissant un nom, un fournisseur et une région.

   ```
   eksctl register cluster --name my-cluster --provider my-provider --region region-code
   ```

   Exemple de sortie :

   ```
   2021-08-19 13:47:26 [#]  creating IAM role "eksctl-20210819194112186040"
   2021-08-19 13:47:26 [#]  registered cluster "<name>" successfully
   2021-08-19 13:47:26 [#]  wrote file eks-connector.yaml to <current directory>
   ```

```
2021-08-19 13:47:26 [#]  wrote file eks-connector-clusterrole.yaml to <current
 directory>
2021-08-19 13:47:26 [#]  wrote file eks-connector-console-dashboard-full-access-
group.yaml to <current directory>
2021-08-19 13:47:26 [!]  note: "eks-connector-clusterrole.yaml" and "eks-connector-
console-dashboard-full-access-group.yaml" give full EKS Console access to IAM
 identity "<aws-arn>", edit if required; read https://eksctl.io/usage/eks-connector
 for more info
2021-08-19 13:47:26 [#]  run `kubectl apply -f eks-connector.yaml,eks-connector-
clusterrole.yaml,eks-connector-console-dashboard-full-access-group.yaml` before
 expiry> to connect the cluster
```

Cela crée des fichiers sur votre ordinateur local. Ces fichiers doivent être appliqués au cluster externe dans un délai de 3 jours, sinon l'enregistrement expire.

3. Dans un terminal pouvant accéder au cluster, appliquez le fichier `eks-connector-binding.yaml` :

```
kubectl apply -f eks-connector-binding.yaml
```

Étape 2 : Installation de l'agent **`eks-connector`**

Pour installer l'agent `eks-connector`, utilisez l'un des outils suivants :

- the section called "Helm"
- the section called "yaml"

Helm

> **Note**
>
> Si vous avez enregistré le cluster avec `eksctl`, utilisez la méthode du manifeste YAML plutôt que la méthode des Charts de Helm.

1. Si vous avez utilisé l'interface CLI AWS à l'étape précédente, remplacez `ACTIVATION_CODE` et `ACTIVATION_ID` dans la commande suivante par les valeurs `activationId` et `activationCode`, respectivement. Remplacez `aws-region` par la région AWS que vous

avez utilisée à l'étape précédente. Ensuite, exécutez la commande pour installer l'agent `eks-connector` dans le cluster d'enregistrement :

```
$ helm install eks-connector \
  --namespace eks-connector \
  oci://public.ecr.aws/eks-connector/eks-connector-chart \
  --set eks.activationCode=ACTIVATION_CODE \
  --set eks.activationId=ACTIVATION_ID \
  --set eks.agentRegion=aws-region
```

Si vous avez utilisé la AWS Management Console à l'étape précédente, utilisez la commande que vous avez copiée à l'étape précédente et qui contient ces valeurs.

2. Vérifiez l'état de santé du déploiement du `eks-connector` installé et attendez que le statut du cluster enregistré dans Amazon EKS soit `ACTIVE`.

yaml

Effectuez la connexion en appliquant le fichier manifeste Amazon EKS Connector à votre cluster Kubernetes. Pour ce faire, vous devez utiliser les méthodes décrites précédemment. Si le manifeste n'est pas appliqué dans un délai de trois jours, l'enregistrement du connecteur Amazon EKS expire. Si la connexion de cluster expire, le cluster doit être désenregistré avant de reconnecter le cluster.

1. Téléchargez le fichier YAML Amazon EKS Connector.

```
curl -O https://amazon-eks.s3.us-west-2.amazonaws.com/eks-connector/manifests/eks-connector/latest/eks-connector.yaml
```

2. Modifiez le fichier YAML Amazon EKS Connector pour remplacer toutes les références de `%AWS_REGION%`, `%EKS_ACTIVATION_ID%`, `%EKS_ACTIVATION_CODE%` par la `aws-region`, l'`activationId` et le `activationCode` à partir de la sortie de l'étape antérieure.

   L'exemple de commande suivant peut remplacer ces valeurs.

```
sed -i "s~%AWS_REGION%~$aws-region~g; s~%EKS_ACTIVATION_ID%~$EKS_ACTIVATION_ID~g; s~%EKS_ACTIVATION_CODE%~$(echo -n $EKS_ACTIVATION_CODE | base64)~g" eks-connector.yaml
```

> **Important**
>
> Assurez-vous que votre code d'activation est au format base64.

3. Dans un terminal pouvant accéder au cluster, vous pouvez appliquer le fichier manifeste mis à jour en exécutant la commande suivante :

```
kubectl apply -f eks-connector.yaml
```

4. Après l'application du manifeste Amazon EKS Connector et des fichiers YAML de liaison de rôle à votre cluster Kubernetes, vérifiez que le cluster est maintenant connecté.

```
aws eks describe-cluster \
     --name "my-first-registered-cluster" \
     --region AWS_REGION
```

   La sortie doit inclure `status=ACTIVE`.

5. (Facultatif) Attribuez des balises à votre cluster. Pour de plus amples informations, consultez the section called “Balisage de vos ressources ”.

Étapes suivantes

Si vous rencontrez des problèmes lors de ces étapes, consultez la rubrique the section called “Dépannage d’EKS Connector”.

Pour accorder à d’autres principaux IAM l’accès à la console Amazon EKS afin de consulter les ressources Kubernetes dans un cluster connecté, consultez the section called “Accorder l’accès à tous les clusters”.

Autoriser l’accès à l’affichage des ressources du cluster Kubernetes sur une console Amazon EKS

Accordez à des principaux IAM l’accès à la console Amazon EKS pour afficher des informations sur les ressources Kubernetes exécutées sur votre cluster connecté.

Prérequis

Le principal IAM que vous utilisez pour accéder à la AWS Management Console doit répondre aux exigences suivantes :

- Il doit bénéficier de l'autorisation IAM d'`eks:AccessKubernetesApi`.
- Le compte de service Amazon EKS Connector peut emprunter l'identité du principal IAM dans le cluster. Cela permet au connecteur Amazon EKS de mapper le principal IAM à un utilisateur Kubernetes.

Pour créer et appliquer le rôle de cluster Amazon EKS Connector

1. Téléchargez le modèle de rôle de cluster `eks-connector`.

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/eks-connector/manifests/eks-connector-console-roles/eks-connector-clusterrole.yaml
```

2. Modifiez le fichier YAML du modèle de rôle de cluster. Remplacez les références de `%IAM_ARN%` par l'Amazon Resource Name (ARN) de votre principal IAM.
3. Appliquez le rôle de cluster Amazon EKS Connector YAML à votre cluster Kubernetes.

```
kubectl apply -f eks-connector-clusterrole.yaml
```

Pour qu'un principal IAM puisse afficher les ressources Kubernetes dans la console Amazon EKS, il doit être associé à un Kubernetes `role` ou `clusterrole` avec les autorisations nécessaires pour lire les ressources. Pour plus d'informations, consultez Utilisation de l'autorisation RBAC dans la documentation Kubernetes.

Pour configurer un principal IAM pour accéder au cluster connecté

1. Vous pouvez télécharger n'importe lequel de ces exemples de fichier manifeste pour créer un `clusterrole` et une `clusterrolebinding` ou un `role` et une `rolebinding` :

   Afficher les ressources Kubernetes dans tous les espaces de noms

   - Le rôle de cluster `eks-connector-console-dashboard-full-access-clusterrole` donne accès à tous les espaces de noms et toutes les ressources qui peuvent être visualisés dans la console. Vous pouvez changer le nom du rôle `role`, du

`clusterrole` et de leur liaison correspondante avant de l'appliquer à votre cluster. Utilisez la commande suivante pour télécharger un exemple de fichier.

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/eks-connector/manifests/
eks-connector-console-roles/eks-connector-console-dashboard-full-access-
group.yaml
```

Afficher les ressources Kubernetes dans un espace de noms spécifique

- L'espace de nom dans ce fichier est `default`, donc si vous souhaitez spécifier un espace de noms différent, modifiez le fichier avant de l'appliquer à votre cluster. Utilisez la commande suivante pour télécharger un exemple de fichier.

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/eks-connector/manifests/
eks-connector-console-roles/eks-connector-console-dashboard-restricted-access-
group.yaml
```

2. Modifiez le fichier YAML à accès complet ou à accès restreint pour remplacer les références de `%IAM_ARN%` par l'Amazon Resource Name (ARN) de votre principal IAM.
3. Appliquez les fichiers YAML à accès complet ou à accès restreint à votre cluster Kubernetes. Remplacez la valeur du fichier YAML par la vôtre.

```
kubectl apply -f eks-connector-console-dashboard-full-access-group.yaml
```

Pour afficher les ressources Kubernetes dans votre cluster connecté, consultez the section called "Accéder aux ressources du cluster". Les données relatives à certains types de ressources de l'onglet Ressources ne sont pas disponibles pour les clusters connectés.

Désenregistrer un cluster Kubernetes depuis la console Amazon EKS

Si vous n'utilisez plus un cluster, vous pouvez annuler son enregistrement. Après l'annulation de son enregistrement, le cluster n'est plus visible dans la console Amazon EKS.

Vous devez disposer des autorisations suivantes pour appeler l'API deregisterCluster :

- `eks:DeregisterCluster`
- `ssm:DeleteActivation`

- `ssm:DeregisterManagedInstance`

Ce processus comporte deux étapes : annulation de l'enregistrement du cluster auprès d'Amazon EKS et désinstallation de l'agent eks-connector dans le cluster.

Désenregistrer le cluster Kubernetes

Pour désenregistrer un cluster du connecteur Amazon EKS, vous pouvez utiliser l'un des outils suivants :

- the section called "AWS CLI"
- the section called "AWS Management Console"
- the section called "`eksctl`"

AWS CLI

1. La CLI AWS doit être installé. Pour l'installer ou la mettre à niveau, consultez Installation de la CLI AWS.
2. Assurez-vous que le rôle d'agent Amazon EKS Connector est créé.
3. Annulez l'enregistrement du cluster connecté.

```
aws eks deregister-cluster \
    --name my-cluster \
    --region region-code
```

AWS Management Console

1. Ouvrez la console Amazon EKS.
2. Choisissez Clusters.
3. Sur la page Clusters, sélectionnez le cluster connecté et sélectionnez Deregister (Annuler l'enregistrer).
4. Confirmez que vous voulez annuler l'enregistrement du cluster.

`eksctl`

1. Installez la `eksctl` version `0.68` ou une version ultérieure. Pour l'installer ou la mettre à niveau, veuillez consulter la rubrique the section called “Créer un cluster (eksctl)”.
2. Assurez-vous que le rôle d'agent Amazon EKS Connector est créé.
3. Annulez l'enregistrement du cluster connecté :

```
eksctl deregister cluster --name my-cluster
```

Nettoyez les ressources de votre cluster Kubernetes

Pour désinstaller l'agent `eks-connector`, utilisez l'un des outils suivants :

- the section called “helm”
- the section called “yaml”

helm

Exécutez la commande suivante pour désinstaller l'agent.

```
helm -n eks-connector uninstall eks-connector
```

yaml

1. Supprimez le fichier YAML Amazon EKS Connector de votre cluster Kubernetes.

```
kubectl delete -f eks-connector.yaml
```

2. Si vous avez créé `clusterrole` ou `clusterrolebindings` pour ajouter des principaux IAM supplémentaires pour accéder au cluster, supprimez-les de votre cluster Kubernetes.

Dépannage des problèmes liés à Amazon EKS Connector

Cette rubrique couvre certaines des erreurs les plus courantes que vous pouvez rencontrer lors de l'utilisation d'Amazon EKS Connector, y compris des instructions sur la façon de les résoudre et des solutions de contournement.

Dépannage de base

Cette section décrit les étapes à suivre pour diagnostiquer les problèmes liés à Amazon EKS Connector.

Vérifier le statut d'Amazon EKS Connector

Pour vérifier le contrôle de statut d'Amazon EKS Connector, veuillez saisir :

```
kubectl get pods -n eks-connector
```

Inspecter les journaux d'Amazon EKS Connector

Le pod Amazon EKS Connector se compose de trois conteneurs. Pour récupérer les journaux complets de tous ces conteneurs afin que vous puissiez les inspecter, exécutez les commandes suivantes :

- `connector-init`

```
kubectl logs eks-connector-0 --container connector-init -n eks-connector
kubectl logs eks-connector-1 --container connector-init -n eks-connector
```

- `connector-proxy`

```
kubectl logs eks-connector-0 --container connector-proxy -n eks-connector
kubectl logs eks-connector-1 --container connector-proxy -n eks-connector
```

- `connector-agent`

```
kubectl exec eks-connector-0 --container connector-agent -n eks-connector -- cat /
var/log/amazon/ssm/amazon-ssm-agent.log
kubectl exec eks-connector-1 --container connector-agent -n eks-connector -- cat /
var/log/amazon/ssm/amazon-ssm-agent.log
```

Obtenir le nom effectif du cluster

Les clusters Amazon EKS sont identifiés de manière unique par `clusterName` au sein d'un seul compte AWS et d'une seule région AWS. Si vous disposez de plusieurs clusters connectés dans Amazon EKS, vous pouvez vérifier à quel cluster Amazon EKS le cluster Kubernetes actuel est

associé. Pour ce faire, saisissez la commande suivante pour connaître le `clusterName` du cluster actuel.

```
kubectl exec eks-connector-0 --container connector-agent -n eks-connector \
  -- cat /var/log/amazon/ssm/amazon-ssm-agent.log | grep -m1 -oE "eks_c:[a-zA-Z0-9_-]+"
 | sed -E "s/^.*eks_c:([a-zA-Z0-9_-]+)_[a-zA-Z0-9]+.*$/\1/"
kubectl exec eks-connector-1 --container connector-agent -n eks-connector \
  -- cat /var/log/amazon/ssm/amazon-ssm-agent.log | grep -m1 -oE "eks_c:[a-zA-Z0-9_-]+"
 | sed -E "s/^.*eks_c:([a-zA-Z0-9_-]+)_[a-zA-Z0-9]+.*$/\1/"
```

Commandes diverses

Les commandes suivantes sont utiles pour récupérer les informations dont vous avez besoin pour résoudre les problèmes.

- Utilisez la commande suivante pour collecter les images utilisées par les pods dans Amazon EKS Connector.

  ```
  kubectl get pods -n eks-connector -o jsonpath="{.items[*].spec.containers[*].image}"
   | tr -s '[[:space:]]' '\n'
  ```

- Utilisez la commande suivante pour rassembler les noms des nœuds sur lesquels Amazon EKS Connector est exécuté.

  ```
  kubectl get pods -n eks-connector -o jsonpath="{.items[*].spec.nodeName}" | tr -s
   '[[:space:]]' '\n'
  ```

- Exécutez la commande suivante pour obtenir les versions de votre client et serveur Kubernetes.

  ```
  kubectl version
  ```

- Exécutez la commande suivante pour obtenir des informations sur vos nœuds.

  ```
  kubectl get nodes -o wide --show-labels
  ```

Problème Helm : 403 Forbidden

Si vous avez reçu le message d'erreur suivant lors de l'exécution des commandes d'installation de Helm :

```
Error: INSTALLATION FAILED: unexpected status from HEAD request to https://
public.ecr.aws/v2/eks-connector/eks-connector-chart/manifests/0.0.6: 403 Forbidden
```

Vous pouvez exécuter la ligne suivante pour y remédier :

```
docker logout public.ecr.aws
```

Erreur de console : le cluster est bloqué dans l'état En attente

Si le cluster reste bloqué dans l'état `Pending` sur la console Amazon EKS après votre inscription, cela peut être dû au fait qu'Amazon EKS Connector n'a pas encore réussi à connecter le cluster à AWS. Pour un cluster enregistré, l'état `Pending` signifie que la connexion n'a pas été établie avec succès. Pour résoudre ce problème, assurez-vous d'avoir appliqué le manifeste au cluster Kubernetes cible. Si vous l'avez appliqué au cluster mais que celui-ci est toujours à l'état `Pending`, il est fort probable que le statefulset `eks-connector` ne soit pas sain. Pour résoudre ce problème, consultez the section called "Les pods du connecteur Amazon EKS sont en boucle de crash" dans cette rubrique.

Erreur de console : l'utilisateur system:serviceaccount:eks-connector:eks-connector ne peut pas usurper l'identité des utilisateurs de ressources dans le groupe d'API à portée du cluster

Amazon EKS Connector utilise l'emprunt d'identité des utilisateurs Kubernetes pour agir au nom des principaux IAM à partir de la AWS Management Console. Chaque principal qui accède à l'API Kubernetes à partir du compte de service AWS `eks-connector` doit disposer d'une autorisation pour usurper l'identité de l'utilisateur Kubernetes correspondant avec un ARN IAM comme nom d'utilisateur Kubernetes. Dans les exemples suivants, l'ARN IAM est mappé à un utilisateur Kubernetes.

- L'utilisateur IAM *`john`* du compte AWS *`111122223333`* est mappé à un utilisateur Kubernetes. Les bonnes pratiques IAM recommandent d'accorder des autorisations à des rôles plutôt qu'à des utilisateurs.

  ```
  arn:aws:iam::111122223333:user/john
  ```

- Le rôle IAM *`admin`* du compte AWS *`111122223333`* est mappé à un utilisateur Kubernetes :

```
arn:aws:iam::111122223333:role/admin
```

Le résultat est un ARN de rôle IAM, au lieu de l'ARN de session AWS STS.

Pour savoir comment configurer `ClusterRole` et `ClusterRoleBinding` afin d'accorder au compte de service `eks-connector` le privilège de compte de service pour usurper l'identité de l'utilisateur mappé, consultez the section called "Accorder l'accès à tous les clusters". Assurez-vous que, dans le modèle, `%IAM_ARN%` est remplacé par l'ARN IAM du principal IAM de la AWS Management Console.

Erreur de console : […] est interdit : l'utilisateur […] ne peut pas répertorier la ressource […] dans le groupe API à portée du cluster

Considérons le problème suivant. Le connecteur Amazon EKS a réussi à usurper l'identité du principal IAM de la AWS Management Console demandeur dans le cluster Kubernetes cible. Cependant, le principal usurpé ne dispose pas de l'autorisation RBAC pour les opérations API Kubernetes.

Pour résoudre ce problème, il existe deux méthodes pour accorder des autorisations à des utilisateurs supplémentaires. Si vous avez déjà installé eks-connector via les charts de Helm, vous pouvez facilement accorder l'accès aux utilisateurs en exécutant la commande suivante. Remplacez `userARN1` et `userARN2` par une liste des ARN des rôles IAM afin de donner accès à la visualisation des ressources Kubernetes :

```
helm upgrade eks-connector oci://public.ecr.aws/eks-connector/eks-connector-chart \
    --reuse-values \
    --set 'authentication.allowedUserARNs={userARN1,userARN2}'
```

Ou, en tant qu'administrateur du cluster, accordez le niveau approprié de privilèges RBAC aux utilisateurs Kubernetes individuels. Pour plus d'informations et d'exemples, consultez the section called "Accorder l'accès à tous les clusters".

Erreur de console : Amazon EKS ne peut pas communiquer avec le serveur API de votre cluster Kubernetes. Le cluster doit être dans un état ACTIF pour que la connexion soit réussie. Veuillez réessayer dans quelques minutes.

Si le service Amazon EKS ne parvient pas à communiquer avec le connecteur Amazon EKS dans le cluster cible, cela peut être dû à l'une des raisons suivantes :

- Amazon EKS Connector dans le cluster cible n'est pas sain.
- Une connectivité médiocre ou une connexion interrompue entre le cluster cible et la Région AWS.

Pour résoudre ce problème, consultez les journaux d'Amazon EKS Connector. Si vous ne voyez pas d'erreur pour le connecteur Amazon EKS, réessayez la connexion après quelques minutes. Si vous rencontrez régulièrement une latence élevée ou une connectivité intermittente pour le cluster cible, envisagez de réenregistrer le cluster dans une région AWS plus proche de vous.

Les pods du connecteur Amazon EKS sont en boucle de crash

De nombreuses raisons peuvent expliquer pourquoi un pod du connecteur Amazon EKS passe à l'état `CrashLoopBackOff`. Ce problème concerne probablement le conteneur `connector-init`. Vérifiez l'état du pod du connecteur Amazon EKS.

```
kubectl get pods -n eks-connector
```

L'exemple qui suit illustre un résultat.

```
NAME              READY   STATUS                  RESTARTS   AGE
eks-connector-0   0/2     Init:CrashLoopBackOff   1          7s
```

Si votre sortie est similaire à la sortie précédente, consultez la section the section called "Inspecter les journaux d'Amazon EKS Connector" pour résoudre le problème.

Échec de l'initialisation de eks-connector : InvalidActivation

Lorsque vous démarrez Amazon EKS Connector pour la première fois, il enregistre un `activationId` et `activationCode` avec Amazon Web Services. L'enregistrement peut échouer,

ce qui peut provoquer le crash du conteneur `connector-init` avec une erreur similaire à la suivante.

```
F1116 20:30:47.261469       1 init.go:43] failed to initiate eks-connector:
 InvalidActivation:
```

Pour résoudre ce problème, tenez compte des causes suivantes et des correctifs recommandés :

- L'enregistrement a peut-être échoué, car les éléments `activationId` et `activationCode` ne figurent pas dans votre fichier manifeste. Si c'est le cas, assurez-vous qu'il s'agit des valeurs correctes renvoyées par l'opération d'API `RegisterCluster` et que `activationCode` se trouve dans le fichier manifeste. Le `activationCode` est ajouté aux secrets Kubernetes, il doit donc être encodé en `base64`. Pour de plus amples informations, consultez the section called "Étape 1 : Enregistrement du cluster".
- L'enregistrement a peut-être échoué car votre activation a expiré. En effet, pour des raisons de sécurité, vous devez activer Amazon EKS Connector dans les trois jours suivant l'enregistrement du cluster. Pour résoudre ce problème, assurez-vous que le manifeste du connecteur Amazon EKS est appliqué au cluster Kubernetes cible avant la date et l'heure d'expiration. Pour confirmer la date d'expiration de votre activation, exécutez un appel à l'opération API `DescribeCluster`.

  ```
  aws eks describe-cluster --name my-cluster
  ```

  Dans l'exemple de réponse suivant, la date et l'heure d'expiration sont enregistrées sous la forme `2021-11-12T22:28:51.101000-08:00`.

  ```
  {
      "cluster": {
          "name": "my-cluster",
          "arn": "arn:aws:eks:region:111122223333:cluster/my-cluster",
          "createdAt": "2021-11-09T22:28:51.449000-08:00",
          "status": "FAILED",
          "tags": {
          },
          "connectorConfig": {
              "activationId": "00000000-0000-0000-0000-000000000000",
              "activationExpiry": "2021-11-12T22:28:51.101000-08:00",
              "provider": "OTHER",
              "roleArn": "arn:aws:iam::111122223333:role/my-connector-role"
          }
  ```

```
    }
}
```

Si l'`activationExpiry` est passée, désenregistrez le cluster et enregistrez-le à nouveau. Cela génère une nouvelle activation.

Le nœud du cluster manque de connectivité sortante

Pour fonctionner correctement, Amazon EKS Connector nécessite une connectivité sortante à plusieurs points de terminaison AWS. Vous ne pouvez pas connecter un cluster privé sans connectivité sortante à une région AWS cible. Pour résoudre ce problème, vous devez ajouter la connectivité sortante nécessaire. Pour plus d'informations sur les exigences de connecteur, consultez the section called “Considérations relatives à Amazon EKS Connector”.

Les pods du connecteur Amazon EKS sont dans l'état `ImagePullBackOff`

Si vous exécutez la commande `get pods` et que les pods sont dans l'état `ImagePullBackOff`, ils ne peuvent pas fonctionner correctement. Si les pods Amazon EKS Connector sont dans l'état `ImagePullBackOff`, ils ne peuvent pas fonctionner correctement. Vérifiez l'état de vos pods Amazon EKS Connector.

```
kubectl get pods -n eks-connector
```

L'exemple qui suit illustre un résultat.

```
NAME              READY   STATUS                  RESTARTS   AGE
eks-connector-0   0/2     Init:ImagePullBackOff   0          4s
```

Le fichier manifeste Amazon EKS Connector par défaut fait référence aux images de la galerie publique Amazon ECR. Il est possible que le cluster Kubernetes cible ne puisse pas extraire d'images de la galerie publique Amazon ECR. Veuillez soit résoudre le problème d'extraction d'images de la galerie publique Amazon ECR, soit envisager la mise en miroir des images dans le registre de conteneurs privé de votre choix.

Question fréquente sur le connecteur AWS

Q : Comment fonctionne la technologie sous-jacente à Amazon EKS Connector ?

R : le connecteur Amazon EKS est basé sur l'agent AWS Systems Manager (Systems Manager). Le connecteur Amazon EKS s'exécute en tant que `StatefulSet` sur votre cluster Kubernetes. Il établit une connexion et proxie la communication entre le serveur API de votre cluster et Amazon Web Services. Cela permet d'afficher les données du cluster dans la console Amazon EKS jusqu'à ce que vous déconnectiez le cluster de AWS. L'agent Systems Manager est un projet open source. Pour plus d'informations sur ce projet, consultez la page du projet sur GitHub.

Q : J'ai un cluster Kubernetes sur site que je souhaite connecter. Dois-je ouvrir des ports de pare-feu pour le connecter ?

R : non, vous n'avez pas besoin d'ouvrir de ports de pare-feu. Le cluster Kubernetes nécessite uniquement une connexion sortante vers les régions AWS. Les services AWS n'accèdent jamais aux ressources de votre réseau sur site. Amazon EKS Connector s'exécute sur votre cluster et initie la connexion à AWS. Lorsque l'enregistrement du cluster est terminé, AWS n'émet des commandes vers Amazon EKS Connector qu'après avoir lancé une action depuis la console Amazon EKS qui nécessite des informations du serveur API Kubernetes sur votre cluster.

Q : Quelles sont les données envoyées de mon cluster à AWS par Amazon EKS Connector ?

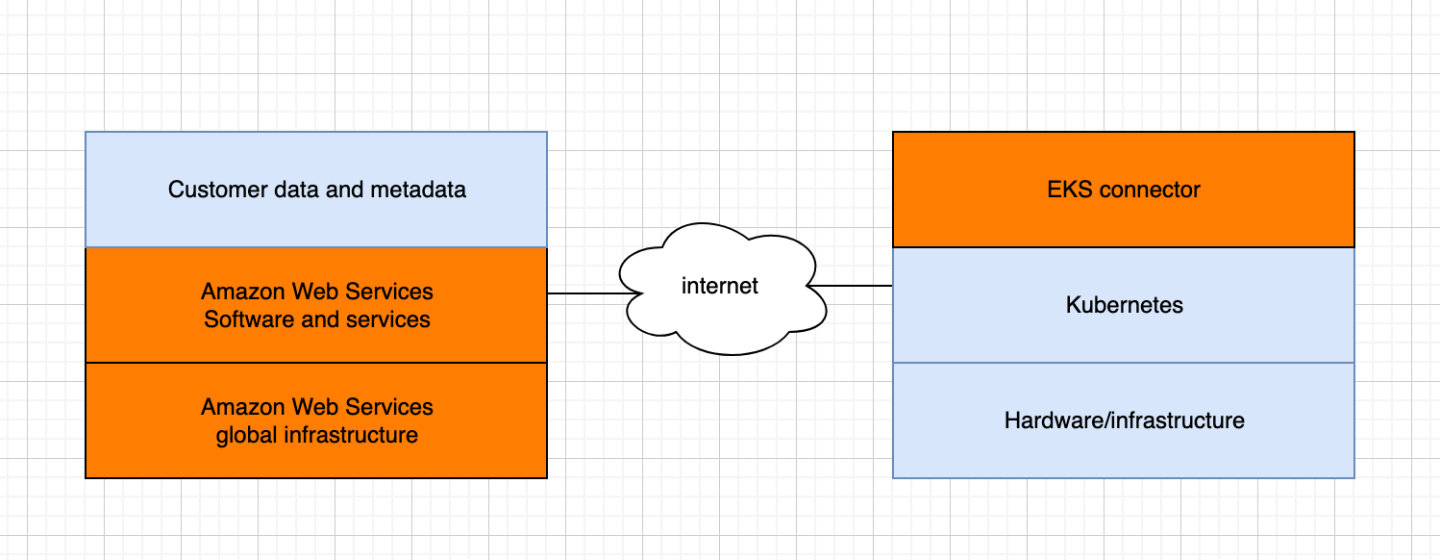
R : le connecteur Amazon EKS envoie les informations techniques nécessaires à l'enregistrement de votre cluster sur AWS. Il envoie également des métadonnées de cluster et de charge de travail pour les fonctions de la console Amazon EKS demandées par les clients. Amazon EKS Connector ne recueille ou n'envoie ces données que si vous lancez une action à partir de la console Amazon EKS ou de l'API Amazon EKS qui nécessite l'envoi de ces données à AWS. À l'exception du numéro de version Kubernetes, AWS ne stocke aucune donnée par défaut. Il ne stocke les données que si vous l'y autorisez.

Q : Puis-je connecter un cluster en dehors d'une région AWS ?

R : Oui, vous pouvez connecter un cluster depuis n'importe quel emplacement à Amazon EKS. De plus, votre service Amazon EKS peut être situé dans n'importe quelle région AWS publique commerciale AWS. Cela fonctionne avec une connexion réseau valide entre votre cluster et la région AWS cible. Nous vous recommandons de choisir une région AWS la plus proche de l'emplacement de votre cluster afin d'optimiser les performances de l'interface utilisateur. Par exemple, si vous avez un cluster fonctionnant à Tokyo, connectez votre cluster à la région AWS de Tokyo (c'est-à-dire la région AWS `ap-northeast-1`) pour bénéficier d'une faible latence. Vous pouvez connecter un cluster depuis n'importe quel emplacement à Amazon EKS dans n'importe quelle région AWS commerciale publique, à l'exception des régions AWS Chine ou GovCloud.

Comprendre la sécurité dans le connecteur Amazon EKS

Amazon EKS Connector est un composant open source qui s'exécute sur votre cluster Kubernetes. Ce cluster peut être situé en dehors de l'environnement AWS. Cela crée des considérations supplémentaires pour les responsabilités en matière de sécurité. Cette configuration peut être illustrée par le schéma suivant. L'orange représente les responsabilités AWS, et le bleu représente les responsabilités des clients :



Cette rubrique décrit les différences entre le modèle de responsabilité si le cluster connecté est externe à AWS.

Responsabilités AWS

- Maintenir, créer et fournir le connecteur Amazon EKS, qui est un composant open source qui s'exécute sur le cluster Kubernetes d'un client et communique avec AWS.
- Maintenir la sécurité des communications de la couche de transport et de la couche applicative entre le cluster Kubernetes connecté et les services AWS.

Responsabilités client

- Sécurité spécifique au cluster Kubernetes, notamment dans les domaines suivants :
  - Les secrets Kubernetes doivent être correctement cryptés et protégés.
  - Verrouillez l'accès à l'espace de noms `eks-connector`.

- Configurer des autorisations de contrôle d'accès basé sur les rôles (RBAC) pour gérer l'accès des principaux IAM depuis AWS. Pour obtenir des instructions, veuillez consulter the section called “Accorder l’accès à tous les clusters”.
- Installation et mise à niveau d'Amazon EKS Connector.
- Maintenir le matériel, des logiciels et de l'infrastructure prenant en charge le cluster Kubernetes connecté.
- Sécuriser leurs comptes AWS (par exemple, en utilisant des informations d'identification d'utilisateur root sécurisées).

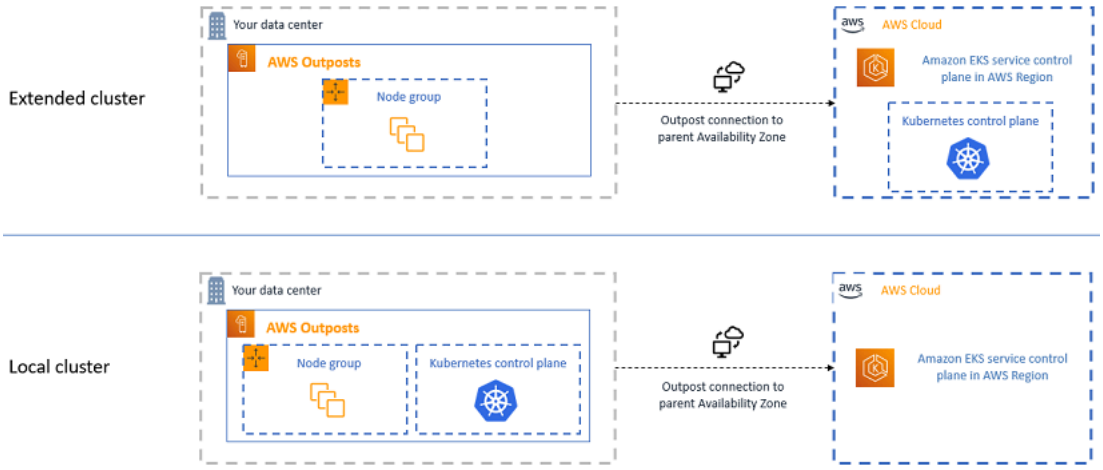
Déployez Amazon EKS sur site avec AWS Outposts

Vous pouvez utiliser Amazon EKS pour exécuter des applications Kubernetes sur site sur AWS Outposts. Vous pouvez déployer Amazon EKS sur des Outposts avec :

- Clusters étendus : exécutez le plan de contrôle Kubernetes dans une région AWS et les nœuds sur votre Outpost.
- Clusters locaux : exécutez le plan de contrôle Kubernetes et les nœuds sur votre Outpost.

Pour les deux options de déploiement, le plan de contrôle Kubernetes est entièrement géré par AWS. Vous pouvez utiliser les API, les outils et la console Amazon EKS que vous utilisez dans le cloud pour créer et exécuter Amazon EKS sur des Outposts.

Le schéma suivant illustre ces options de déploiement.



Quand utiliser chaque option de déploiement

Les clusters locaux et étendus sont des options de déploiement polyvalentes et peuvent être utilisées pour de nombreuses applications.

Avec les clusters locaux, vous pouvez exécuter l'intégralité du cluster Amazon EKS localement sur des Outposts. Cette option permet de réduire le risque d'interruption des applications qui pourrait résulter de déconnexions temporaires du réseau au cloud. Ces déconnexions du réseau peuvent être causées par des coupures de fibre ou des événements météorologiques. Étant donné que l'ensemble du cluster Amazon EKS s'exécute localement sur les Outposts, les applications restent disponibles. Vous pouvez effectuer des opérations de cluster en cas de déconnexion du réseau au cloud. Pour

de plus amples informations, consultez the section called "Préparer les déconnexions". Si vous êtes préoccupé par la qualité de la connexion réseau entre vos Outposts et la région parent AWS et que vous avez besoin d'une haute disponibilité en cas de déconnexion réseau, utilisez l'option de déploiement en cluster local.

Avec les clusters étendus, vous pouvez économiser de la capacité sur votre Outpost, car le plan de contrôle Kubernetes s'exécute dans la région parent AWS. Cette option est adaptée si vous pouvez investir dans une connectivité réseau fiable et redondante entre votre Outpost et la région AWS. La qualité de la connexion réseau est essentielle pour cette option. La manière dont Kubernetes gère les déconnexions réseau entre le plan de contrôle Kubernetes et les nœuds peut entraîner des durées d'indisponibilité des applications. Pour plus d'informations sur le comportement de Kubernetes, consultez les sections Planification, Préemption et Expulsion dans la documentation Kubernetes.

Comparaison des options de déploiement

Le tableau suivant compare les différences entre les deux options.

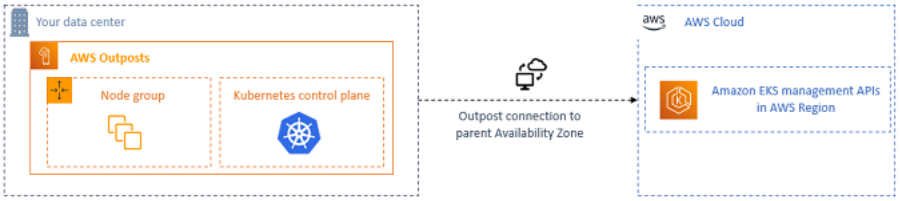
Fonctionnalité	Cluster étendu	Cluster local
Emplacement du plan de contrôle Kubernetes	Région AWS	Outpost
Compte du plan de contrôle Kubernetes	Compte AWS	Votre compte
Disponibilité par région	consultez points de terminaison de service	USA Est (Ohio), USA Est (Virginie du Nord), USA Ouest (Californie du Nord), USA Ouest (Oregon), Asie-Pacifique (Séoul), Asie-Pacifique (Singapour), Asie-Pacifique (Sydney), Asie-Pacifique (Tokyo), Canada (Centre), Europe (Francfort), Europe (Irlande), Europe (Londres), Moyen-Orient (Bahreïn) et Amérique du Sud (São Paulo)

Fonctionnalité	Cluster étendu	Cluster local
Versions Kubernetes mineures	eks/latest/userguide/kubernetes-versions.html[Supported Amazon EKS versions, type="documentation"].	eks/latest/userguide/kubernetes-versions.html[Supported Amazon EKS versions, type="documentation"].
Versions de plateforme	Consulter Version deplateforme EKS	Consultez the section called "Versions de la plateforme EKS"
Facteurs de forme Outpost	Racks Outpost	Racks Outpost
Interfaces utilisateur	AWS Management Console, AWS CLI, API Amazon EKS`eksctl`, AWS CloudFormation et Terraform	AWS Management Console, AWS CLI, API Amazon EKS`eksctl`, AWS CloudFormation et Terraform
Politiques gérées	AmazonEKSClusterPolicy et the section called "AWS politique gérée : Amazon EKSService RolePolicy"	AmazonEKSLocalOutpostClusterPolicy et the section called "AWS politique gérée : Amazon EKSLocal OutpostServiceRolePolicy"
VPC et sous-réseaux de cluster	Consultez the section called "Exigences requises pour le VPC et les sous-réseaux"	Consultez the section called "Créer un VPC et des sous-réseaux"
Accès au point de terminaison de cluster	Public, privé ou les deux	Privés uniquement
Authentification du serveur d'API Kubernetes	Gestion des identités et des accès AWS (IAM) et OIDC	IAM et certificats `x.509`
Types de nœud	Autogérés uniquement	Autogérés uniquement
Types de nœuds de calcul	Amazon EC2 à la demande	Amazon EC2 à la demande

Fonctionnalité	Cluster étendu	Cluster local
Types de stockage de nœuds	SSD gp2 Amazon EBS et NVMe local	SSD gp2 Amazon EBS et NVMe local
AMI optimisées pour Amazon EKS	Amazon Linux, Windows et Bottlerocket	Amazon Linux uniquement
Versions d'adresses IP	`IPv4` uniquement	`IPv4` uniquement
Modules complémentaires	Modules complémentaires ou autogérés Amazon EKS	Module complémentaire autogéré uniquement
Interface réseau de conteneurs par défaut	Plugin CNI Amazon VPC pour Kubernetes	Plugin CNI Amazon VPC pour Kubernetes
Journaux du plan de contrôle Kubernetes	Amazon CloudWatch Logs	Amazon CloudWatch Logs
Équilibrage de charge	Utilisez l'AWS Load Balancer Controller pour allouer des Application Load Balancers uniquement (pas de Network Load Balancers)	Utilisez l'AWS Load Balancer Controller pour allouer des Application Load Balancers uniquement (pas de Network Load Balancers)
Chiffrement d'enveloppe des secrets	Consultez the section called "Activer le chiffrement secret"	Non pris en charge
Rôles IAM pour les comptes de service	Consultez the section called "Informations d'identification auprès de l'IRSA"	Non pris en charge
Dépannage	Consultez Dépannage	Consultez the section called "Résolution des problèmes liés à un cluster"

Créez des clusters Amazon EKS locaux sur AWS Outposts pour une haute disponibilité

Vous pouvez utiliser des clusters locaux pour exécuter l'intégralité de votre cluster Amazon EKS localement sur AWS Outposts. Cela permet de réduire le risque d'interruption des applications qui pourrait résulter de déconnexions temporaires du réseau au cloud. Ces déconnexions peuvent être causées par des coupures de fibre ou des événements météorologiques. Étant donné que l'ensemble du cluster Kubernetes s'exécute localement sur les Outposts, les applications restent disponibles. Vous pouvez effectuer des opérations de cluster en cas de déconnexion du réseau au cloud. Pour de plus amples informations, consultez the section called "Préparer les déconnexions". Le schéma suivant illustre un déploiement de cluster local.



Les clusters locaux sont généralement disponibles pour une utilisation avec les racks Outposts.

Régions AWS prises en charge

Vous pouvez créer des clusters locaux dans les régions AWS suivantes : USA Est (Ohio), USA Est (Virginie du Nord), USA Ouest (Californie du Nord), USA Ouest (Oregon), Asie-Pacifique (Séoul), Asie-Pacifique (Singapour), Asie-Pacifique (Sydney), Asie-Pacifique (Tokyo), Canada (Centre), Europe (Francfort), Europe (Irlande), Europe (Londres), Moyen-Orient (Bahreïn) et Amérique du Sud (São Paulo). Pour des informations détaillées sur les fonctions prises en charge, consultez the section called "Comparaison des options de déploiement".

Déployez un cluster Amazon EKS sur AWS Outposts

Cette rubrique fournit une vue d'ensemble des éléments à prendre en compte lors de l'exécution d'un cluster local sur un Outpost. Elle donne également des instructions pour déployer un cluster local sur un Outpost.

> **Important**
>
> - Ces considérations ne figurent pas dans la documentation Amazon EKS associée. Si d'autres rubriques de la documentation Amazon EKS entrent en conflit avec les considérations présentées ici, suivez ces dernières.
> - Ces considérations sont sujettes à modification et peuvent changer fréquemment. Nous vous recommandons donc de consulter régulièrement cette rubrique.
> - De nombreuses considérations sont différentes de celles relatives à la création d'un cluster sur le AWS cloud.

- Les clusters locaux prennent uniquement en charge les racks Outpost. Un seul cluster local peut s'exécuter sur plusieurs racks Outpost physiques qui constituent un seul Outpost logique. Un seul cluster local ne peut pas s'exécuter sur plusieurs Outposts logiques. Chaque Outpost logique possède un seul ARN d'Outpost.
- Les clusters locaux exécutent et gèrent le plan de contrôle Kubernetes sur votre compte sur l'Outpost. Vous ne pouvez pas exécuter de charges de travail sur les instances du plan de contrôle Kubernetes ni modifier les composants du plan de contrôle Kubernetes. Ces nœuds sont gérés par le service Amazon EKS. Les modifications apportées au plan de contrôle Kubernetes ne sont pas conservées lors des actions de gestion automatique d'Amazon EKS, telles que l'application de correctifs.
- Les clusters locaux prennent en charge les modules complémentaires autogérés et les groupes de nœuds Amazon Linux autogérés. Le plug-in Amazon VPC CNI pour Kubernetes, kube-proxy et les modules complémentaires CoreDNS sont automatiquement installés sur les clusters locaux.
- Les clusters locaux nécessitent l'utilisation d'Amazon EBS sur les Outposts. Amazon EBS doit être disponible dans votre Outpost pour le stockage du plan de contrôle Kubernetes. Les Outposts prennent en charge les volumes `gp2` Amazon EBS uniquement.
- Les Kubernetes `PersistentVolumes` basés sur Amazon EBS sont pris en charge à l'aide du pilote CSI Amazon EBS.
- Les instances du plan de contrôle des clusters locaux sont configurées selon une topologie empilée à haute disponibilité. Deux des trois instances du plan de contrôle doivent être saines à tout moment pour maintenir le quorum. Si le quorum est perdu, contactez le AWS support, car certaines actions côté service seront nécessaires pour activer les nouvelles instances gérées.

Conditions préalables

- Connaissance des options de déploiement des Outposts, de la sélection des types d'instances et des groupes de placement pour les clusters Amazon EKS sur les AWS Outposts en fonction de considérations relatives à la capacité, ainsi que des exigences et considérations relatives aux VPC.
- Un Outpost existant. Pour plus d'informations, consultez What is AWS Outposts.
- L'outil de ligne de commande `kubectl` est installé sur votre ordinateur ou AWS CloudShell. La version peut correspondre à celle utilisée par votre cluster Kubernetes, ou différer d'au plus une version mineure, qu'elle soit antérieure ou plus récente. Par exemple, si la version de votre cluster est `1.29`, vous pouvez utiliser la version `kubectl` `1.28`, `1.29` ou `1.30`. Pour installer ou mettre à niveau `kubectl`, veuillez consulter the section called "Configurer kubectl et eksctl".
- Version `2.12.3` ou version ultérieure `1.27.160` ou version ultérieure de l'interface de ligne de AWS commande (AWS CLI) installée et configurée sur votre appareil ou AWS CloudShell. Pour vérifier votre version actuelle, utilisez `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Les gestionnaires de packages tels que `yum` Homebrew pour macOS ont souvent plusieurs versions de retard sur la dernière version de la AWS CLI. `apt-get` Pour installer la dernière version, consultez la section Installation et configuration rapide avec aws configure dans le Guide de l'utilisateur de l'interface de ligne de AWS commande. La version de la AWS CLI installée AWS CloudShell peut également avoir plusieurs versions de retard par rapport à la dernière version. Pour le mettre à jour, consultez la section Installation de la AWS CLI dans votre répertoire de base dans le guide de AWS CloudShell l'utilisateur.
- Un principal IAM avec des autorisations pour `create` et `describe` un cluster Amazon EKS. Pour plus d'informations, consultez the section called "Créer un cluster Kubernetes local sur un Outpost" et the section called "Affichage de la liste ou description de tous les clusters".

Lorsqu'un cluster Amazon EKS local est créé, le principal IAM qui crée le cluster est ajouté de manière permanente. Le principal est spécifiquement ajouté à la table d'autorisation RBAC Kubernetes en tant qu'administrateur. Cette entité possède des autorisations `system:masters`. L'identité de cette entité n'est pas visible dans la configuration de votre cluster. Il est donc important de noter l'entité qui a créé le cluster et de veiller à ne jamais la supprimer. Initialement, seul le principal qui a créé le serveur peut effectuer des appels vers le serveur d'API Kubernetes à l'aide de `kubectl`. Si vous utilisez la console pour créer le cluster, assurez-vous que les mêmes informations d'identification IAM figurent dans la chaîne d'identification du AWS SDK lorsque vous exécutez des `kubectl` commandes sur votre cluster. Une fois votre cluster créé, vous pouvez accorder à d'autres principaux IAM l'accès à votre cluster.

Créer un cluster local Amazon EKS

Vous pouvez créer un cluster local à l'aide des outils suivants décrits dans cette page :

- the section called "`eksctl`"
- the section called "AWS Management Console"

Vous pouvez également utiliser la AWS CLI, l'API Amazon EKS AWS SDKs, AWS CloudFormationou Terraform pour créer des clusters sur Outposts.

`eksctl`

Pour créer un cluster avec **`eksctl`**

1. Installez la version `0.215.0` ou une version ultérieure de l'outil de ligne de `eksctl` commande sur votre appareil ou AWS CloudShell. Pour installer ou mettre à jour `eksctl`, veuillez consulter Installation dans la documentation de `eksctl`.
2. Copiez les contenus suivants sur votre appareil. Remplacez les valeurs suivantes, puis exécutez la commande modifiée pour créer le fichier `outpost-control-plane.yaml` :
   - *`region-code`*Remplacez-le par la AWS région prise en charge dans laquelle vous souhaitez créer votre cluster.
   - Remplacez *`my-cluster`* par un nom pour votre cluster. Un nom ne peut contenir que des caractères alphanumériques (sensibles à la casse) et des traits d'union. Il doit commencer par un caractère alphanumérique et ne peut pas dépasser 100 caractères. Le nom doit être unique dans la AWS région et le AWS compte dans lesquels vous créez le cluster. Le nom doit être unique dans la AWS région et le AWS compte dans lesquels vous créez le cluster.
   - Remplacez *`vpc-ExampleID1`* et *`subnet-ExampleID1`* par le IDs VPC et le sous-réseau existants. Le VPC et le sous-réseau doivent répondre aux exigences de la section Créer un VPC et des sous-réseaux pour les clusters Amazon EKS sur les Outposts. AWS
   - Remplacez *`uniqueid`* par l'identifiant de votre avant-poste.
   - Remplacez *`m5.large`* par un type d'instance disponible sur votre Outpost. Avant de choisir un type d'instance, consultez the section called "Considérations relatives à la capacité". Trois instances du plan de contrôle sont déployées. Vous ne pouvez pas modifier ce nombre.

```
cat >outpost-control-plane.yaml <<EOF
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
```

```
metadata:
  name: my-cluster
  region: region-code
  version: "1.33"

vpc:
  clusterEndpoints:
    privateAccess: true
  id: "vpc-vpc-ExampleID1"
  subnets:
    private:
      outpost-subnet-1:
        id: "subnet-subnet-ExampleID1"

outpost:
  controlPlaneOutpostARN: arn:aws: outposts:region-code:111122223333:outpost/op-
uniqueid
  controlPlaneInstanceType: m5.large
EOF
```

Pour voir la liste complète de toutes les options disponibles et des valeurs par défaut, consultez AWS Outposts Support et Config file schema (langue française non garantie) dans la documentation `eksctl`.

3. Créez le cluster à l'aide du fichier de configuration créé à l'étape précédente. `eksctl` crée un VPC et un sous-réseau sur votre Outpost pour y déployer le cluster.

```
eksctl create cluster -f outpost-control-plane.yaml
```

L'approvisionnement de cluster dure plusieurs minutes. Pendant la création du cluster, plusieurs lignes de sortie apparaissent. La dernière ligne de sortie est similaire à celle de l'exemple suivant.

```
[#]  EKS cluster "my-cluster" in "region-code" region is ready
```

> **Tip**
>
> Pour afficher la plupart des options qui peuvent être spécifiées lors de la création d'un cluster avec `eksctl`, utilisez la commande `eksctl create cluster --help`. Pour consulter toutes les options disponibles, vous pouvez utiliser un fichier `config`. Pour plus d'informations, consultez Utilisation des fichiers de configuration et du schéma du fichier

de configuration dans la documentation `eksctl`. Vous pouvez trouver des exemples de fichiers de configuration sur GitHub.

La commande `eksctl` a automatiquement créé une entrée d'accès pour le principal IAM (utilisateur ou rôle) qui a créé le cluster et a accordé au principal IAM des autorisations d'administrateur sur les objets Kubernetes du cluster. Si vous ne souhaitez pas que le créateur du cluster dispose d'un accès administrateur aux objets Kubernetes sur le cluster, ajoutez le texte suivant au fichier de configuration précédent : `bootstrapClusterCreatorAdminPermissions: false` (au même niveau que `metadata`, `vpc`, et `outpost`). Si vous avez ajouté cette option, après la création du cluster, vous devez créer une entrée d'accès pour au moins un principal IAM, sinon aucun principal IAM n'aura accès aux objets Kubernetes du cluster.

AWS Management Console

Pour créer votre cluster à l'aide de AWS Management Console

1. Vous devez disposer d'un VPC et d'un sous-réseau existants qui répondent aux exigences d'Amazon EKS. Pour de plus amples informations, veuillez consulter the section called "Créer un VPC et des sous-réseaux".
2. Si vous avez déjà un rôle IAM de cluster local, ou si vous allez créer votre cluster avec `eksctl`, vous pouvez ignorer cette étape. Par défaut, `eksctl` crée un rôle pour vous.
   a. Exécutez la commande suivante pour créer un fichier JSON de politique d'approbation IAM.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

b. Créez le rôle IAM de cluster Amazon EKS. Pour créer un rôle IAM, le principal IAM qui crée le rôle doit se voir attribuer l'action (autorisation) IAM `iam:CreateRole`.

```
aws iam create-role --role-name myAmazonEKSLocalClusterRole --assume-role-policy-
document file://"eks-local-cluster-role-trust-policy.json"
```

c. Associez la politique gérée Amazon EKS nommée Amazon EKSLocal OutpostClusterPolicy au rôle. Pour attacher une politique IAM à un principal IAM, le principal qui attache la politique doit se voir attribuer l'une des actions (autorisations) IAM `iam:AttachUserPolicy` ou `iam:AttachRolePolicy`.

```
aws iam attach-role-policy --policy-arn arn:aws: iam::aws:policy/
AmazonEKSLocalOutpostClusterPolicy --role-name myAmazonEKSLocalClusterRole
```

3. Ouvrez la console Amazon EKS.
4. En haut de l'écran de la console, assurez-vous que vous avez sélectionné une AWS région prise en charge.
5. Choisissez Add cluster (Ajouter un cluster), puis choisissez Create (Créer).
6. Sur la page Configurer le cluster, saisissez ou sélectionnez les valeurs des champs suivants :
   - Emplacement du plan de contrôle Kubernetes — Choisissez Outposts. AWS
   - ID Outpost (ID d'Outpost) – Choisissez l'ID de l'Outpost sur lequel vous souhaitez créer votre plan de contrôle.
   - Instance type (Type d'instance) – Sélectionnez un type d'instance. Seuls les types d'instances disponibles dans votre Outpost sont affichés. Dans la liste déroulante, chaque type d'instance indique le nombre de nœuds pour lesquels le type d'instance est recommandé. Avant de choisir un type d'instance, consultez the section called “Considérations relatives à la capacité”. Tous les réplicas sont déployés à l'aide du même type d'instance. Vous ne pouvez pas modifier le type d'instance une fois le cluster créé. Trois instances du plan de contrôle sont déployées. Vous ne pouvez pas modifier ce nombre.
   - Nom : nom de votre cluster. Il doit être unique dans votre AWS compte. Un nom ne peut contenir que des caractères alphanumériques (sensibles à la casse) et des traits d'union. Il doit commencer par un caractère alphanumérique et ne peut pas dépasser 100 caractères. Le nom doit être unique dans la AWS région et le AWS compte dans lesquels vous créez le cluster. Le nom doit être unique dans la AWS région et le AWS compte dans lesquels vous créez le cluster.

- Version Kubernetes : choisissez la version Kubernetes que vous souhaitez utiliser pour votre cluster. Nous vous recommandons de sélectionner la dernière version, sauf si vous devez utiliser une version antérieure.
- Rôle de service de cluster : choisissez le rôle IAM du cluster Amazon EKS que vous avez créé lors d'une étape précédente pour permettre au plan de contrôle Kubernetes de gérer les ressources. AWS
- Accès administrateur du cluster Kubernetes : si vous souhaitez que le principal IAM (rôle ou utilisateur) qui crée le cluster dispose d'un accès administrateur aux objets Kubernetes du cluster, acceptez la valeur par défaut (autoriser). Amazon EKS crée une entrée d'accès pour le principal IAM et accorde des autorisations d'administrateur du cluster à cette entrée d'accès. Pour plus d'informations sur les entrées d'accès, consultez the section called "Entrées d'accès".

  Si vous souhaitez qu'un principal IAM différent du principal qui crée le cluster dispose d'un accès administrateur aux objets du cluster Kubernetes, choisissez l'option Interdire. Après la création du cluster, tout principal IAM disposant des autorisations IAM pour créer des entrées d'accès peut ajouter des entrées d'accès pour tous les principaux IAM ayant besoin d'accéder aux objets du cluster Kubernetes. Pour plus d'informations sur les autorisations IAM requises, consultez la rubrique Actions définies par Amazon Elastic Kubernetes Service dans la Référence des autorisations de service. Si vous choisissez l'option d'interdiction et que vous ne créez aucune entrée d'accès, aucun principal IAM n'aura accès aux objets Kubernetes du cluster.
- Identifications◦: (facultatif) ajoutez des identifications à votre cluster. Pour de plus amples informations, veuillez consulter the section called "Balisage de vos ressources ". Lorsque vous avez terminé d'utiliser cette page, choisissez Suivant.

7. Sur la page Spécifier les réseaux sélectionnez des valeurs pour les champs suivants :
   - VPC – Choisissez un VPC existant. Le VPC doit disposer d'un nombre suffisant d'adresses IP disponibles pour le cluster, pour tous les nœuds et pour les autres ressources Kubernetes que vous souhaitez créer. Votre VPC doit répondre aux exigences indiquées dans Exigences relatives aux VPC.
   - Sous-réseaux : par défaut, tous les sous-réseaux disponibles dans le VPC spécifié dans le champ précédent sont présélectionnés. Les sous-réseaux que vous choisissez doivent répondre aux exigences énoncées dans la section Exigences et considérations relatives aux sous-réseaux.
   - Groupes de sécurité : (facultatif) spécifiez un ou plusieurs groupes de sécurité créant des interfaces réseau auxquelles vous souhaitez associer Amazon EKS. Amazon EKS crée automatiquement un groupe de sécurité qui permet la communication entre votre cluster et votre

VPC. Amazon EKS associe ce groupe de sécurité, et tout ce que vous choisissez, aux interfaces réseau qu'il crée. Pour plus d'informations sur le groupe de sécurité de cluster créé par Amazon EKS, consultez the section called “Exigences de groupe de sécurité”. Vous pouvez modifier les règles dans le groupe de sécurité de cluster créé par Amazon EKS. Si vous choisissez d'ajouter vos propres groupes de sécurité, vous ne pouvez pas modifier ceux que vous choisissez après la création du cluster. Pour que les hôtes sur site puissent communiquer avec le point de terminaison du cluster, vous devez autoriser le trafic entrant provenant du groupe de sécurité du cluster. Pour les clusters qui n'ont pas de connexion Internet entrante et sortante (également connus sous le nom de clusters privés), vous devez effectuer l'une des opérations suivantes :

- Ajoutez le groupe de sécurité associé aux points de terminaison d'un VPC requis. Pour plus d'informations sur les points de terminaison requis, voir the section called “Utilisation des points de terminaison de VPC d'interface” la section Accès aux AWS services par sous-réseau.
- Modifiez le groupe de sécurité qu'Amazon EKS a créé pour autoriser le trafic provenant du groupe de sécurité associé aux points de terminaison d'un VPC. Lorsque vous avez terminé d'utiliser cette page, choisissez Suivant.

8. Sur la page Configurer l'observabilité, vous pouvez éventuellement choisir les options de Métriques et de Journalisation du plan de contrôle que vous voulez activer. Par défaut, chaque type de journal est désactivé.
   - Pour plus d'informations sur l'option de métriques Prometheus, consultez the section called “Étape 1 : activer les métriques Prometheus”.
   - Pour plus d'informations sur les options de Journalisation du plan de contrôle, consultez the section called “journaux de plan de contrôle ”. Lorsque vous avez terminé d'utiliser cette page, choisissez Suivant.
9. Sur la page Vérifier et créer, passez en revue les informations que vous avez saisies ou sélectionnées sur les pages précédentes. Si vous devez apporter des modifications, choisissez Modifier. Quand vous êtes satisfait, choisissez Créer. Le champ État affiche EN COURS DE CRÉATION pendant que le cluster est provisionné.

   L'approvisionnement de cluster dure plusieurs minutes.

Afficher votre cluster local Amazon EKS

1. Une fois votre cluster créé, vous pouvez consulter les instances du plan EC2 de contrôle Amazon qui ont été créées.

```
aws ec2 describe-instances --query 'Reservations[*].Instances[*].{Name:Tags[?
Key==`Name`]|[0].Value}' | grep my-cluster-control-plane
```

L'exemple qui suit illustre un résultat.

```
"Name": "my-cluster-control-plane-id1"
"Name": "my-cluster-control-plane-id2"
"Name": "my-cluster-control-plane-id3"
```

Chaque instance est rejetée avec `node-role.eks-local.amazonaws.com/control-plane` pour qu'aucune charge de travail ne soit planifiée sur les instances du plan de contrôle. Pour plus d'informations sur les rejets, consultez la section Rejets et tolérances dans la documentation Kubernetes. Amazon EKS surveille en permanence l'état des clusters locaux. Nous effectuons des actions de gestion automatiques, telles que des correctifs de sécurité et la réparation des instances défectueuses. Lorsque des clusters locaux sont déconnectés du cloud, nous prenons des mesures pour nous assurer que le cluster retrouve un état sain lors de la reconnexion.

2. Si vous avez créé votre cluster à l'aide de `eksctl`, vous pouvez sauter cette étape. `eksctl` complète cette étape pour vous. Activez `kubectl` pour communiquer avec votre cluster en ajoutant un nouveau contexte au fichier `kubectl config`. Pour savoir comment créer et mettre à jour le fichier, consultez the section called "Accès au cluster avec kubectl".

   ```
   aws eks update-kubeconfig --region region-code --name my-cluster
   ```

   L'exemple qui suit illustre un résultat.

   ```
   Added new context arn:aws: eks:region-code:111122223333:cluster/my-cluster to /home/
   username/.kube/config
   ```

3. Pour vous connecter au serveur d'API Kubernetes de votre cluster local, vous devez avoir accès à la passerelle locale pour le sous-réseau ou vous connecter depuis le VPC. Pour plus d'informations sur la connexion d'un rack Outpost à votre réseau local, consultez Comment fonctionnent les passerelles locales pour les racks dans le Guide de l'utilisateur d' AWS Outposts. Si vous utilisez le routage VPC direct et que le sous-réseau Outpost possède un acheminement vers votre passerelle locale, les adresses IP privées des instances du plan de contrôle Kubernetes sont automatiquement diffusées sur votre réseau local. Le point de terminaison du serveur d'API Kubernetes du cluster local est hébergé dans Amazon Route 53 (Route 53). Le point de

terminaison du service d'API peut être résolu par des serveurs DNS publics vers les adresses IP privées des serveurs d'API Kubernetes.

Les instances du plan de contrôle Kubernetes des clusters locaux sont configurées avec des interfaces réseau Elastic statiques et des adresses IP privées fixes qui ne changent pas tout au long du cycle de vie du cluster. Les appareils qui interagissent avec le serveur d'API Kubernetes peuvent ne pas avoir de connectivité à Route 53 pendant les déconnexions du réseau. Si tel est le cas, nous recommandons de configurer `/etc/hosts` avec les adresses IP privées statiques pour la poursuite des opérations. Nous vous recommandons également de configurer des serveurs DNS locaux et de les connecter à votre Outpost. Pour plus d'informations, consultez la documentation AWS Outposts. Exécutez la commande suivante pour confirmer que la communication est établie avec votre cluster.

```
kubectl get svc
```

L'exemple qui suit illustre un résultat.

```
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.100.0.1   <none>        443/TCP   28h
```

4. (Facultatif) Testez l'authentification auprès de votre cluster local lorsqu'il est déconnecté du AWS cloud. Pour obtenir des instructions, veuillez consulter the section called "Préparer les déconnexions".

Ressources internes

Amazon EKS crée les ressources suivantes sur votre cluster. Les ressources sont destinées à un usage interne d'Amazon EKS. Pour le bon fonctionnement de votre cluster, ne modifiez pas ces ressources.

- Les mirror pods suivants :
  - `aws-iam-authenticator-`*`node-hostname`*
  - `eks-certificates-controller-`*`node-hostname`*
  - `etcd-`*`node-hostname`*
  - `kube-apiserver-`*`node-hostname`*
  - `kube-controller-manager-`*`node-hostname`*
  - `kube-scheduler-`*`node-hostname`*

- Les modules complémentaires autogérés suivants :
  - `kube-system/coredns`
  - `kube-system/kube-proxy` (non créé tant que vous n'avez pas ajouté votre premier nœud)
  - `kube-system/aws-node` (non créé tant que vous n'avez pas ajouté votre premier nœud). Les clusters locaux utilisent le plug-in CNI Amazon VPC pour Kubernetes pour la mise en réseau des clusters. Ne modifiez pas la configuration des instances du plan de contrôle (pods nommés `aws-node-controlplane-*`). Il existe des variables de configuration que vous pouvez utiliser pour modifier la valeur par défaut lorsque le plugin crée de nouvelles interfaces réseau. Pour plus d'informations, consultez la documentation sur GitHub.
- Les services suivants :
  - `default/kubernetes`
  - `kube-system/kube-dns`
- Une politique `PodSecurityPolicy` nommée `eks.system`
- Un rôle `ClusterRole` nommé `eks:system:podsecuritypolicy`
- Un rôle `ClusterRoleBinding` nommé `eks:system`
- Outre le groupe de sécurité du cluster, Amazon EKS crée un groupe de sécurité nommé dans votre AWS compte`eks-local-internal-do-not-use-or-edit-`*`cluster-name`*`-`*`uniqueid`* . Ce groupe de sécurité permet au trafic de circuler librement entre les composants Kubernetes exécutés sur les instances du plan de contrôle.

Étapes suivantes recommandées :

- Accordez au principal IAM qui a créé le cluster les autorisations requises pour consulter les ressources Kubernetes dans AWS Management Console
- Accordez aux entités IAM l'accès à votre cluster. Si vous souhaitez que les entités puissent afficher les ressources Kubernetes dans la console Amazon EKS, accordez-leur les autorisations requises.
- Configurer la journalisation de votre cluster
- Familiarisez-vous avec ce qui se passe en cas de déconnexion du réseau.
- Ajouter des nœuds à votre cluster
- Envisagez de mettre en place un plan de sauvegarde pour votre `etcd`. Amazon EKS ne prend pas en charge la sauvegarde et la restauration automatisées d'`etcd` pour les clusters locaux. Pour plus d'informations, consultez la section Sauvegarde d'un cluster etcd dans la documentation

Kubernetes. Les deux options principales utilisent `etcdctl` pour automatiser la prise d'instantanés ou l'utilisation de la sauvegarde des volumes de stockage Amazon EBS.

Découvrez les versions des plateformes Kubernetes et Amazon EKS pour Outposts AWS

Les versions de la plateforme de cluster locale représentent les capacités du cluster Amazon EKS sur AWS Outposts. Les versions comprennent les composants qui s'exécutent sur le plan de contrôle Kubernetes, pour lesquels les drapeaux du serveur d'API Kubernetes sont activés. Elles comprennent également la version actuelle du correctif Kubernetes. Une ou plusieurs versions de plateforme sont associées à chaque version mineure de Kubernetes. Les versions de plateforme pour les différentes versions mineures de Kubernetes sont indépendantes. Les versions de plateforme pour les clusters locaux et les clusters Amazon EKS dans le cloud sont indépendantes.

Lorsqu'une nouvelle version mineure de Kubernetes est disponible pour les clusters locaux, telle que `1.31`, la version initiale de la plateforme pour cette version mineure de Kubernetes commence à `eks-local-outposts.1`. Cependant, Amazon EKS publie régulièrement de nouvelles versions de plateforme pour activer de nouveaux paramètres de plan de contrôle Kubernetes et fournir des correctifs de sécurité.

Lorsque de nouvelles versions de plateforme de clusters locaux deviennent disponibles pour une version mineure :

- Le numéro de version de plateforme est incrémenté (`eks-local-outposts.n+1`).
- Amazon EKS met automatiquement à jour tous les clusters locaux existants vers la dernière version de plateforme pour les versions mineures Kubernetes correspondantes. Les mises à jour automatiques de versions de plateforme existantes sont déployées de façon incrémentielle. Le processus de déploiement consiste à remplacer les instances du plan de contrôle Kubernetes gérées qui s'exécutent sur Outpost, une par une, jusqu'à ce que les trois instances soient remplacées par de nouvelles.
- Le processus de remplacement de l'instance du plan de contrôle Kubernetes s'arrêtera s'il existe un risque d'interruption du service. Amazon EKS ne tentera de remplacer une instance que si les deux autres instances du plan de contrôle Kubernetes sont en bon état et remplissent toutes les conditions de disponibilité en tant que nœud de cluster.
- Le déploiement d'une version de plateforme prend généralement moins de 30 minutes. Si un cluster reste `UPDATING` actif pendant une période prolongée, consultez le Support the section

called “Résolution des problèmes liés à un cluster” et demandez de l' AWS aide. Ne mettez jamais fin manuellement aux instances du plan de contrôle Kubernetes, sauf indication contraire du Support. AWS

- Amazon EKS peut publier une nouvelle AMI de nœud avec une version de correctif correspondante. Toutes les versions de correctif sont compatibles entre le plan de contrôle et le nœud Kubernetes AMIs pour une seule version mineure de Kubernetes.

Les nouvelles versions de plateforme n'introduisent pas des modifications importantes ou ne provoquent pas des interruptions de service.

Les clusters locaux sont toujours créés avec la dernière version disponible de la plateforme (`eks-local-outposts.n`) pour la version Kubernetes spécifiée.

Les versions actuelles et récentes de plateforme sont décrites dans les tableaux ci-dessous.

Pour recevoir des notifications concernant toutes les modifications apportées aux fichiers sources de cette page de documentation spécifique, vous pouvez vous abonner à l'URL suivante à l'aide d'un lecteur RSS :

```
https://github.com/awsdocs/amazon-eks-user-guide/commits/mainline/latest/ug/outposts/eks-outposts-platform-versions.adoc.atom
```

Version de Kubernetes **`1.31`**

Les contrôleurs d'admission suivants sont activés pour toutes les versions de plateforme `1.31` : `CertificateApproval`, `CertificateSigning`, `CertificateSubjectRestriction`, `ClusterTrustBundleAttest`, `DefaultIngressClass`, `DefaultStorageClass`, `DefaultTolerationSeconds`, `ExtendedResourceToleration`, `LimitRanger`, `MutatingAdmissionWebhook`, `NamespaceLifecycle`, `NodeRestriction`, `PersistentVolumeClaimResize`, `PodSecurity`, `Priority`, `ResourceQuota`, `RuntimeClass`, `ServiceAccount`, `StorageObjectInUseProtection`, `TaintNodesByCondition`, `ValidatingAdmissionPolicy` et `ValidatingAdmissionWebhook`.

Version de Kubernetes	Version de la plateforme Amazon EKS	Notes de mise à jour	Date de publication
1.31.14	eks-local-outposts.8	Nouvelle version de la plateforme avec correctifs et améliorations de sécurité. Kube-proxy mis à jour vers. v1.31.14 AWS Authentificateur IAM mis à jour vers. v0.7.8 Mise à jour du plugin CNI Amazon VPC pour Kubernetes vers v1.20.4. Version de Bottlerocket mise à jour vers v1.52.0.	23 décembre 2025
1.31.12	eks-local-outposts.5	Nouvelle version de la plateforme avec correctifs et améliorations de sécurité. Kube-proxy mis à jour vers. v1.31.10 AWS Authentificateur IAM mis à jour vers. v0.7.4 Mise à jour du plugin CNI Amazon VPC pour Kubernetes vers v1.20.2. Version de Bottlerocket mise à jour vers v1.47.0.	3 octobre 2025

Version de Kubernetes	Version de la plateforme Amazon EKS	Notes de mise à jour	Date de publication
1.31.9	eks-local-outposts.4	Nouvelle version de la plateforme avec correctifs et améliorations de sécurité. Kube-proxy mis à jour vers. v1.31.9 AWS Authentificateur IAM mis à jour vers. v0.7.2 Le plugin Amazon VPC CNI pour Kubernetes a été mis à jour vers la version de Bottlerocket mise à jour vers. v1.20.0 v1.43.0	9 août 2025
1.31.7	eks-local-outposts.3	Nouvelle version de la plateforme avec correctifs et améliorations de sécurité. Kube-proxy mis à jour vers. v1.31.9 AWS Authentificateur IAM mis à jour vers. v0.7.1 Mise à jour du plugin CNI Amazon VPC pour Kubernetes vers v1.19.5. Version de Bottlerocket mise à jour vers v1.40.0.	19 juin 2025

Version de Kubernetes	Version de la plateforme Amazon EKS	Notes de mise à jour	Date de publication
1.31.6	eks-local-outposts.2	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations. Version de Bottlerocket mise à jour vers v1.36.0.	24 avril 2025
1.31.6	eks-local-outposts.1	Lancement initial de la version v1.31 de Kubernetes pour les clusters locaux Amazon EKS sur des Outposts.	9 avril 2025

Version de Kubernetes **1.30**

Les contrôleurs d'admission suivants sont activés pour toutes les versions de plateforme 1.30 : CertificateApproval, CertificateSigning, CertificateSubjectRestriction, ClusterTrustBundleAttest, DefaultIngressClass, DefaultStorageClass, DefaultTolerationSeconds, ExtendedResourceToleration, LimitRanger, MutatingAdmissionWebhook, NamespaceLifecycle, NodeRestriction, PersistentVolumeClaimResize, PodSecurity, Priority, ResourceQuota, RuntimeClass, ServiceAccount, StorageObjectInUseProtection, TaintNodesByCondition, ValidatingAdmissionPolicy et ValidatingAdmissionWebhook.

Version de Kubernetes	Version de la plateforme Amazon EKS	Notes de mise à jour	Date de publication
1.30.14	eks-local-outposts.10	Nouvelle version de la plateforme avec correctifs et améliorations de sécurité. AWS Authentificateur IAM mis à jour vers. v0.7.8 Mise à jour du plugin CNI Amazon VPC pour Kubernetes vers v1.20.4. Version de Bottlerocket mise à jour vers v1.52.0.	23 décembre 2025
1.30.14	eks-local-outposts.7	Nouvelle version de la plateforme avec correctifs et améliorations de sécurité. Kube-proxy mis à jour vers. v1.30.14 AWS Authentificateur IAM mis à jour vers. v0.7.4 Mise à jour du plugin CNI Amazon VPC pour Kubernetes vers v1.20.2. Version de Bottlerocket mise à jour vers v1.47.0.	3 octobre 2025
1.30.13	eks-local-outposts.6	Nouvelle version de la plateforme avec correctifs et améliorat	9 août 2025

Version de Kubernetes	Version de la plateforme Amazon EKS	Notes de mise à jour	Date de publication
		ions de sécurité. Kube-proxy mis à jour vers. `v1.30.13` AWS Authentificateur IAM mis à jour vers. `v0.7.2` Mise à jour du plugin CNI Amazon VPC pour Kubernetes vers `v1.20.0`. Version de Bottlerocket mise à jour vers `v1.43.0`.	
`1.30.11`	`eks-local-outposts.5`	Nouvelle version de la plateforme avec correctifs et améliorations de sécurité. Kube-proxy mis à jour vers. `v1.30.11` AWS Authentificateur IAM mis à jour vers. `v0.7.1` Le plugin Amazon VPC CNI pour Kubernetes a été mis à jour vers la version de Bottlerocket mise à jour vers. `v1.19.5` `v1.40.0`	19 juin 2025

Version de Kubernetes	Version de la plateforme Amazon EKS	Notes de mise à jour	Date de publication
1.30.10	eks-local-outposts.4	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations. Version de Bottlerocket mise à jour vers v1.36.0.	24 avril 2025
1.30.10	eks-local-outposts.3	Nouvelle version de la plateforme avec correctifs et améliorations de sécurité. Kube-proxy mis à jour vers. v1.30.10 AWS Authentificateur IAM mis à jour vers. v0.6.29 Mise à jour du plugin CNI Amazon VPC pour Kubernetes vers v1.19.2. CoreDNS mis à jour vers. v1.11.4 AWS Cloud Controller Manager a été mis à jour versv1.30.8. Version de Bottlerocket mise à jour vers v1.34.0.	27 mars 2025

Version de Kubernetes	Version de la plateforme Amazon EKS	Notes de mise à jour	Date de publication
1.30.7	eks-local-outposts.2	Nouvelle version de la plateforme avec correctifs et améliorations de sécurité. Kube-proxy mis à jour vers. v1.30.7 AWS Authentificateur IAM mis à jour vers. v0.6.28 Mise à jour du plugin CNI Amazon VPC pour Kubernetes vers v1.19.0. Version de Bottlerocket mise à jour vers. v1.29.0	10 janvier 2025
1.30.5	eks-local-outposts.1	Lancement initial de la version v1.30 de Kubernetes pour les clusters locaux Amazon EKS sur des Outposts.	13 novembre 2024

Version de Kubernetes **1.29**

Les contrôleurs d'admission suivants sont activés pour toutes les versions de plateforme 1.29 : CertificateApproval, CertificateSigning, CertificateSubjectRestriction, ClusterTrustBundleAttest, DefaultIngressClass, DefaultStorageClass, DefaultTolerationSeconds, ExtendedResourceToleration, LimitRanger, MutatingAdmissionWebhook, NamespaceLifecycle, NodeRestriction, PersistentVolumeClaimResize, PodSecurity, Priority, ResourceQuota, RuntimeClass, ServiceAccount, StorageObjectInUseProtection,

TaintNodesByCondition, ValidatingAdmissionPolicy et ValidatingAdmissionWebhook.

Version de Kubernetes	Version de la plateforme Amazon EKS	Notes de mise à jour	Date de publication
1.29.15	eks-local-outposts.13	Nouvelle version de la plateforme avec correctifs et améliorations de sécurité. AWS Authentificateur IAM mis à jour vers. v0.7.8 Mise à jour du plugin CNI Amazon VPC pour Kubernetes vers v1.20.4. Version de Bottlerocket mise à jour vers v1.52.0.	23 décembre 2025
v1.29.15	eks-local-outposts.10	Nouvelle version de la plateforme avec correctifs et améliorations de sécurité. AWS Authentificateur IAM mis à jour vers. v0.7.4 Mise à jour du plugin CNI Amazon VPC pour Kubernetes vers v1.20.2. Version de Bottlerocket mise à jour vers v1.47.0.	3 octobre 2025
v1.29.15	eks-local-outposts.9	Nouvelle version de la plateforme avec	9 août 2025

Version de Kubernetes	Version de la plateforme Amazon EKS	Notes de mise à jour	Date de publication
		correctifs et améliorations de sécurité. AWS Authentificateur IAM mis à jour vers. `v0.7.2` Mise à jour du plugin CNI Amazon VPC pour Kubernetes vers `v1.20.0`. Version de Bottlerocket mise à jour vers `v1.43.0`.	
`v1.29.15`	`eks-local-outposts.8`	Nouvelle version de la plateforme avec correctifs et améliorations de sécurité. Kube-proxy mis à jour vers. `v1.29.15` AWS Authentificateur IAM mis à jour vers. `v0.7.1` Mise à jour du plugin CNI Amazon VPC pour Kubernetes vers `v1.19.5`. Version de Bottlerocket mise à jour vers `v1.40.0`.	19 juin 2025

Version de Kubernetes	Version de la plateforme Amazon EKS	Notes de mise à jour	Date de publication
v1.29.14	eks-local-outposts.7	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations. Version de Bottlerocket mise à jour vers v1.36.0.	24 mars 2025
v1.29.14	eks-local-outposts.6	Nouvelle version de plateforme avec des corrections de sécurité et des améliorations. kube-proxy mis à jour vers v1.29.14. Mise à jour du plugin CNI Amazon VPC pour Kubernetes vers v1.19.2. CoreDNS mis à jour vers. v1.11.4 AWS Cloud Controller Manager a été mis à jour versv1.29.8. Version de Bottlerocket mise à jour vers v1.34.0.	27 mars 2025

Version de Kubernetes	Version de la plateforme Amazon EKS	Notes de mise à jour	Date de publication
v1.29.11	eks-local-outposts.5	Nouvelle version de plateforme avec des corrections de sécurité et des améliorations. kube-proxy mis à jour vers v1.29.11. Mise à jour du plugin CNI Amazon VPC pour Kubernetes vers v1.19.0. Mise à jour de l'image de CoreDNS vers v1.11.3. Version de Bottlerocket mise à jour vers. v1.29.0	10 janvier 2025
1.29.9	eks-local-outposts.4	Nouvelle version de la plateforme avec correctifs et améliorations de sécurité. Kube-proxy mis à jour vers. v1.29.9 AWS Authentificateur IAM mis à jour vers. v0.6.26 Version de Bottlerocket mise à jour vers. v1.26.0	8 novembre 2024

Version de Kubernetes	Version de la plateforme Amazon EKS	Notes de mise à jour	Date de publication
1.29.6	eks-local-outposts.3	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations. Version de Bottlerocket mise à jour vers. v1.22.0	22 octobre 2024
1.29.6	eks-local-outposts.2	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations. Version de Bottlerocket mise à jour vers. v1.21.0	27 août 2024
1.29.6	eks-local-outposts.1	Lancement initial de la version v1.29 de Kubernetes pour les clusters locaux Amazon EKS sur des Outposts.	20 août 2024

Créer un VPC et des sous-réseaux pour les clusters Amazon EKS sur AWS Outposts

Lorsque vous créez un cluster local, vous spécifiez un VPC et au moins un sous-réseau privé qui fonctionne sur les Outposts. Cette rubrique fournit une vue d'ensemble des exigences et considérations spécifiques au VPC et aux sous-réseaux pour votre cluster local.

Exigences et considérations requises pour le VPC

Lorsque vous créez un cluster local, le VPC que vous spécifiez doit répondre aux exigences et aux considérations suivantes :

- Assurez-vous que le VPC dispose d'un nombre suffisant d'adresses IP pour le cluster local, tous les nœuds et les autres ressources Kubernetes que vous souhaitez créer. Si le VPC que vous souhaitez utiliser ne dispose pas de suffisamment d'adresses IP, augmentez le nombre d'adresses IP disponibles. Vous pouvez effectuer cette opération en associant des blocs d'adresse CIDR (Routage inter-domaines sans classe) supplémentaires avec votre VPC. Vous pouvez associer des blocs d'adresses CIDR privés (RFC 1918) et publics (non-RFC 1918) à votre VPC avant ou après la création de votre cluster. Un cluster peut prendre jusqu'à 5 heures avant qu'un bloc d'adresse CIDR associé à un VPC ne soit reconnu.
- Le VPC ne peut pas avoir de préfixes IP ou de blocs d'adresses CIDR IPv6 attribués. En raison de ces contraintes, les informations fournies dans Attribuer davantage d'adresses IP aux nœuds Amazon EKS avec des préfixes et the section called "IPv6" ne s'appliquent pas à votre VPC.
- Le VPC possède un nom d'hôte DNS et une résolution DNS activés. Sans ces fonctions, la création du cluster local échoue, et vous devez activer les fonctions et recréer votre cluster. Pour plus d'informations, consultez Attributs DNS pour votre VPC dans le guide de l'utilisateur d'Amazon VPC.
- Pour accéder à votre cluster local via votre réseau local, le VPC doit être associé à la table de routage de la passerelle locale de votre Outpost. Pour plus d'informations, consultez Associations de VPC dans le Guide de l'utilisateur AWS Outposts.

Exigences et considérations requises pour les sous-réseaux

Lorsque vous créez le cluster, spécifiez au moins un sous-réseau privé. Si vous spécifiez plusieurs sous-réseaux, les instances du plan de contrôle Kubernetes sont réparties uniformément à travers les sous-réseaux. Si plus d'un sous-réseau est spécifié, les sous-réseaux doivent exister sur le même Outpost. En outre, les sous-réseaux doivent également disposer d'un acheminement et d'autorisations de groupe de sécurité appropriés pour communiquer entre eux. Lorsque vous créez un cluster local, les sous-réseaux que vous spécifiez doivent répondre aux exigences suivantes :

- Les sous-réseaux doivent se trouver sur le même Outpost logique.
- Les sous-réseaux doivent disposer ensemble d'au moins trois adresses IP disponibles pour les instances du plan de contrôle Kubernetes. Si trois sous-réseaux sont spécifiés, chaque sous-réseau doit avoir au moins une adresse IP disponible. Si deux sous-réseaux sont spécifiés, chaque

sous-réseau doit avoir au moins deux adresses IP disponibles. Si un sous-réseau est spécifié, le sous-réseau doit avoir au moins trois adresses IP disponibles.

- Les sous-réseaux disposent d'une route vers la passerelle locale du rack Outpost pour accéder au serveur API Kubernetes via votre réseau local. Si les sous-réseaux ne disposent pas d'une route vers la passerelle locale du rack Outpost, vous devez communiquer avec votre serveur API Kubernetes depuis l'intérieur du VPC.
- Les sous-réseaux doivent utiliser une dénomination basée sur une adresse IP. La dénomination basée sur la ressource Amazon EC2 n'est pas prise en charge par Amazon EKS.

Accès au sous-réseau pour les services AWS

Les sous-réseaux privés du cluster local sur Outposts doivent pouvoir communiquer avec les services AWS régionaux. Vous pouvez y parvenir en utilisant une passerelle NAT pour l'accès Internet sortant ou, si vous voulez préserver la confidentialité de tout le trafic au sein de votre VPC, en utilisant des points de terminaison d'un VPC d'interface.

Utiliser une passerelle NAT

Les sous-réseaux privés du cluster local sur Outposts doivent inclure une table de routage associée contenant un acheminement vers une passerelle NAT dans un sous-réseau public qui est dans la zone de disponibilité parente de l'Outpost. Le sous-réseau public doit disposer d'un acheminement vers une passerelle Internet. La passerelle NAT permet l'accès à Internet sortant et empêche les connexions entrantes non sollicitées depuis Internet vers les instances de l'Outpost.

Utilisation des points de terminaison de VPC d'interface

Si les sous-réseaux privés du cluster local sur Outposts n'ont pas d'accès Internet sortant ou si vous voulez préserver la confidentialité de tout le trafic au sein de votre VPC, vous devez créer les points de terminaison d'un VPC d'interface et le point de terminaison de passerelle suivants dans un sous-réseau régional avant de créer votre cluster.

Point de terminaison	Type de point de terminaison
`com.amazonaws.` *`region-code`* `.ssm`	utilisateur
`com.amazonaws.` *`region-code`* `.ssmmessages`	utilisateur

Point de terminaison	Type de point de terminaison
com.amazonaws. *region-code* .ec2messages	utilisateur
com.amazonaws. *region-code* .ec2	utilisateur
com.amazonaws. *region-code* .secretsmanager	utilisateur
com.amazonaws. *region-code* .logs	utilisateur
com.amazonaws. *region-code* .sts	utilisateur
com.amazonaws. *region-code* .ecr.api	utilisateur
com.amazonaws. *region-code* .ecr.dkr	utilisateur
com.amazonaws. *region-code* .s3	Passerelle

Les points de terminaison doivent répondre aux critères suivants :

- Créé dans un sous-réseau privé situé dans la zone de disponibilité parent de votre Outpost
- Noms DNS privés activés
- Avoir un groupe de sécurité attaché qui autorise le trafic HTTPS entrant à partir de la plage CIDR du sous-réseau privé d'Outpost.

La création de points de terminaison entraîne des frais. Pour plus d'informations, veuillez consulter Tarification AWS PrivateLink. Si vos Pods ont besoin d'accéder à d'autres services AWS, vous devez créer des points de terminaison supplémentaires. Pour obtenir la liste complète des points de terminaison, consultez les services AWS qui s'intègrent à AWS PrivateLink.

Création d'un VPC

Vous pouvez créer un VPC qui répond aux exigences précédentes à l'aide de l'un des modèles AWS CloudFormation suivants :

- Modèle 1 : le modèle crée un VPC avec un sous-réseau privé sur l'Outpost et un sous-réseau public dans la région AWS. Le sous-réseau privé dispose d'une route vers Internet via une

passerelle NAT qui réside dans le sous-réseau public de la région AWS. Ce modèle peut être utilisé pour créer un cluster local dans un sous-réseau avec un accès Internet de sortie.

- Modèle 2 : ce modèle crée un VPC avec un sous-réseau privé sur l'Outpost et l'ensemble minimal de points de terminaison d'un VPC requis pour créer un cluster local dans un sous-réseau qui ne dispose pas d'accès Internet entrant ou sortant (également appelé sous-réseau privé).

Préparer des clusters Amazon EKS locaux sur AWS Outposts pour les déconnexions du réseau

Si votre réseau local a perdu la connectivité avec AWS Cloud, vous pouvez continuer à utiliser votre cluster Amazon EKS local sur un Outpost. Cette rubrique explique comment préparer votre cluster local pour les déconnexions du réseau et les considérations connexes.

- Les clusters locaux garantissent la stabilité et la continuité des opérations en cas de déconnexions réseau temporaires et imprévues. AWS Outposts reste une offre entièrement connectée qui agit comme extension de AWS Cloud dans votre centre de données. En cas de déconnexion du réseau entre votre Outpost et AWS Cloud, nous vous recommandons de tenter de rétablir votre connexion. Pour obtenir des instructions, consultez la liste de contrôle pour le dépannage du réseau de racks AWS Outposts dans le Guide de l'utilisateur AWS Outposts. Pour plus d'informations sur la résolution des problèmes liés aux clusters locaux, consultez the section called "Résolution des problèmes liés à un cluster".
- Les Outposts génèrent une métrique `ConnectedStatus` qui permet de surveiller l'état de connectivité de votre Outpost. Pour plus d'informations, consultez la section Outposts Metrics dans le Guide de l'utilisateur AWS Outposts.
- Les clusters locaux utilisent IAM comme mécanisme d'authentification par défaut à l'aide de l'authentificateur de la gestion des identités et des accès AWS pour Kubernetes. IAM n'est pas disponible lors des déconnexions du réseau. Les clusters locaux prennent en charge un mécanisme d'authentification à l'aide des certificats `x.509` que vous pouvez utiliser pour vous connecter à votre cluster lors des déconnexions du réseau. Pour plus d'informations sur l'obtention et l'utilisation d'un certificat `x.509` pour votre cluster, consultez the section called "Authentification auprès de votre cluster local lors d'une déconnexion du réseau".
- Si vous ne pouvez pas accéder à Route 53 lors des déconnexions du réseau, pensez à utiliser des serveurs DNS locaux dans votre environnement sur site. Les instances du plan de contrôle Kubernetes utilisent des adresses IP statiques. Vous pouvez configurer les hôtes que vous utilisez pour vous connecter à votre cluster avec le nom d'hôte et les adresses IP du point de terminaison

comme alternative à l'utilisation de serveurs DNS locaux. Pour plus d'informations, consultez la section DNS dans le Guide de l'utilisateur AWS Outposts.

- Si vous prévoyez une augmentation du trafic d'applications lorsque le réseau se déconnecte, vous pouvez allouer de la capacité de calcul de réserve dans votre cluster lorsque vous êtes connecté au cloud. Les instances Amazon EC2 sont incluses dans le prix de AWS Outposts. Ainsi, l'exécution d'instances de rechange n'a pas d'impact sur votre coût d'utilisation d'AWS.
- Lors des déconnexions du réseau pour permettre les opérations de création, de mise à jour et de mise à l'échelle des charges de travail, les images de conteneur de votre application doivent être accessibles sur le réseau local et votre cluster doit disposer d'une capacité suffisante. Les clusters locaux n'hébergent pas de registre de conteneurs pour vous. Si les pods ont déjà été exécutés sur ces nœuds, les images de conteneur sont mises en cache sur les nœuds. Si vous téléchargez généralement les images de conteneurs de votre application depuis Amazon ECR dans le cloud, envisagez d'exécuter un cache ou un registre local. Un cache ou un registre local est utile si vous avez besoin d'opérations de création, de mise à jour et de mise à l'échelle pour les ressources de la charge de travail pendant les déconnexions du réseau.
- Les clusters locaux utilisent Amazon EBS comme classe de stockage par défaut pour les volumes persistants et le pilote CSI Amazon EBS pour gérer le cycle de vie des volumes persistants Amazon EBS. Pendant les déconnexions du réseau, les pods qui sont sauvegardés par Amazon EBS ne peuvent pas être créés, mis à jour ou mis à l'échelle. En effet, ces opérations nécessitent des appels à l'API Amazon EBS dans le cloud. Si vous déployez des charges de travail avec état sur des clusters locaux et que vous avez besoin d'opérations de création, de mise à jour ou de mise à l'échelle lors des déconnexions du réseau, envisagez d'utiliser un autre mécanisme de stockage.
- Il est impossible de créer ou de supprimer des instantanés Amazon EBS si AWS Outposts n'a pas accès aux API pertinentes de la région AWS (telles que les API pour Amazon EBS ou Amazon S3).
- Lors de l'intégration d'ALB (Ingress) avec AWS Certificate Manager (ACM), les certificats sont transférés et stockés dans la mémoire de l'instance AWS Outposts ALB Compute. La terminaison TLS actuelle continuera de fonctionner en cas de déconnexion de la région AWS. Dans ce contexte, les opérations de mutation (telles que les nouvelles définitions d'entrée, les nouvelles opérations d'API de certificats basés sur ACM, le dimensionnement du calcul ALB ou la rotation des certificats) échoueront. Pour plus d'informations, consultez la section Dépannage du renouvellement des certificats gérés dans le Guide de l'utilisateur du gestionnaire de certificats AWS.
- Les journaux du plan de contrôle Amazon EKS sont mis en cache localement sur les instances du plan de contrôle Kubernetes lors des déconnexions du réseau. Lors de la reconnexion, les

journaux sont envoyés à CloudWatch Logs dans la région AWS parent. Vous pouvez utiliser Prometheus, Grafana ou les solutions partenaires Amazon EKS pour surveiller le cluster localement à l'aide du point de terminaison des métriques du serveur API Kubernetes ou à l'aide de Fluent Bit pour les journaux.

- Si vous utilisez l'AWS Load Balancer Controller sur Outposts pour le trafic des applications, les pods existants gérés par l'AWS Load Balancer Controller continuent de recevoir du trafic pendant les déconnexions réseau. Les nouveaux pods créés lors de déconnexions du réseau ne reçoivent pas de trafic tant que l'Outpost n'est pas reconnecté au AWS Cloud. Envisagez de définir le nombre de réplicas pour vos applications lorsque vous êtes connecté au AWS Cloud afin de répondre à vos besoins de mise à l'échelle lors des déconnexions du réseau.
- Le plug-in CNI Amazon VPC pour Kubernetes utilise par défaut le mode IP secondaire. Il est configuré avec `WARM_ENI_TARGET=1`, qui permet au plug-in de conserver « une interface réseau entièrement Elastic » contenant les adresses IP disponibles. Pensez à modifier les valeurs `WARM_ENI_TARGET`, `WARM_IP_TARGET` et `MINIMUM_IP_TARGET` en fonction de vos besoins de mise à l'échelle lors d'un état déconnecté. Pour plus d'informations, consultez le fichier readme pour le plug-in sur GitHub. Pour obtenir la liste du nombre maximal de pods pris en charge par chaque type d'instance, consultez le fichier eni-max-pods.txt sur GitHub.

Authentification auprès de votre cluster local lors d'une déconnexion du réseau

La gestion des identités et des accès AWS (IAM) n'est pas disponible en cas de déconnexions réseau. Vous ne pouvez pas vous authentifier auprès de votre cluster local à l'aide des informations d'identification IAM lorsque vous êtes déconnecté. Cependant, vous pouvez vous connecter à votre cluster sur votre réseau local à l'aide de certificats `x509` en cas de déconnexion. Vous devez télécharger et stocker un certificat `X509` client à utiliser lors des déconnexions. Dans cette rubrique, vous allez voir comment créer et utiliser le certificat pour vous authentifier auprès de votre cluster lorsqu'il est dans un état déconnecté.

1. Créer une demande de signature de certificat.
   a. Générez une demande de signature de certificat.

```
openssl req -new -newkey rsa:4096 -nodes -days 365 \
    -keyout admin.key -out admin.csr -subj "/CN=admin"
```

   b. Créez une demande de signature de certificat dans Kubernetes.

```
BASE64_CSR=$(cat admin.csr | base64 -w 0)
```

```
cat << EOF > admin-csr.yaml
apiVersion: certificates.k8s.io/v1
kind: CertificateSigningRequest
metadata:
  name: admin-csr
spec:
  signerName: kubernetes.io/kube-apiserver-client
  request: ${BASE64_CSR}
  usages:
  - client auth
EOF
```

2. Créez une demande de signature de certificat à l'aide de `kubectl`.

```
kubectl create -f admin-csr.yaml
```

3. Vérifiez le statut de la demande de signature de certificat.

```
kubectl get csr admin-csr
```

L'exemple qui suit illustre un résultat.

```
NAME       AGE   REQUESTOR                      CONDITION
admin-csr  11m   kubernetes-admin               Pending
```

Kubernetes a créé la demande de signature de certificat.

4. Approuvez la demande de signature de certificat.

```
kubectl certificate approve admin-csr
```

5. Vérifiez à nouveau l'état de la demande de signature de certificat pour approbation.

```
kubectl get csr admin-csr
```

L'exemple qui suit illustre un résultat.

```
NAME       AGE   REQUESTOR                    CONDITION
admin-csr  11m   kubernetes-admin             Approved
```

6. Récupérez et vérifiez le certificat.

a. Récupérez le certificat.

```
kubectl get csr admin-csr -o jsonpath='{.status.certificate}' | base64 --decode >
 admin.crt
```

b. Vérifiez le certificat.

```
cat admin.crt
```

7. Créez une liaison de rôle de cluster pour un utilisateur `admin`.

```
kubectl create clusterrolebinding admin --clusterrole=cluster-admin \
    --user=admin --group=system:masters
```

8. Générez un kubeconfig spécifique à l'utilisateur pour un état déconnecté.

   Vous pouvez générer un fichier `kubeconfig` à l'aide des certificats `admin` téléchargés. Remplacez *`my-cluster`* et *`apiserver-endpoint`* dans les commandes suivantes.

```
aws eks describe-cluster --name my-cluster \
    --query "cluster.certificateAuthority" \
    --output text | base64 --decode > ca.crt
```

```
kubectl config --kubeconfig admin.kubeconfig set-cluster my-cluster \
    --certificate-authority=ca.crt --server apiserver-endpoint --embed-certs
```

```
kubectl config --kubeconfig admin.kubeconfig set-credentials admin \
    --client-certificate=admin.crt --client-key=admin.key --embed-certs
```

```
kubectl config --kubeconfig admin.kubeconfig set-context admin@my-cluster \
    --cluster my-cluster --user admin
```

```
kubectl config --kubeconfig admin.kubeconfig use-context admin@my-cluster
```

9. Affichez votre fichier `kubeconfig`.

```
kubectl get nodes --kubeconfig admin.kubeconfig
```

10. Si des services sont déjà en production sur votre Outpost, ignorez cette étape. Si Amazon EKS est le seul service qui s'exécute sur votre Outpost et que ce dernier n'est pas en production, vous pouvez simuler une déconnexion du réseau. Avant de passer en production avec votre cluster local, simulez une déconnexion pour vous assurer que vous pouvez accéder à votre cluster lorsqu'il est déconnecté.

    a. Appliquez les règles de pare-feu sur les appareils réseaux qui connectent votre Outpost à la région AWS. Cela déconnecte le lien de service de l'Outpost. Vous ne pouvez pas créer de nouvelles instances. Les instances en cours d'exécution perdent leur connectivité à la région AWS et à Internet.

    b. Vous pouvez tester la connection à votre cluster local à l'aide du certificat `x509` en cas de déconnexion. Assurez-vous de remplacer votre `kubeconfig` par le `admin.kubeconfig` que vous avez créé à une étape précédente. Remplacez *`my-cluster`* par le nom de votre cluster local.

    ```
    kubectl config use-context admin@my-cluster --kubeconfig admin.kubeconfig
    ```

    Si vous remarquez des problèmes avec vos clusters locaux alors qu'ils sont déconnectés, nous vous recommandons d'ouvrir un ticket de support.

Sélection des types d'instances et des groupes de placement pour les clusters Amazon EKS sur AWS Outposts en fonction de considérations de capacité

Cette rubrique fournit des conseils sur la sélection du type d'instance de plan de contrôle Kubernetes et sur l'utilisation (facultative) de groupes de placement pour répondre aux exigences de haute disponibilité de votre cluster Amazon EKS local sur un Outpost.

Avant de sélectionner un type d'instance (telle que `m5`, `c5` ou `r5`) à utiliser pour le plan de contrôle Kubernetes de votre cluster local sur des Outposts, confirmez les types d'instances disponibles dans votre configuration Outpost. Après avoir identifié les types d'instance disponibles, sélectionnez la taille de l'instance (telle que `large`, `xlarge` ou `2xlarge`) en fonction du nombre de nœuds requis par vos charges de travail. Le tableau suivant fournit des recommandations pour choisir une taille d'instance.

> **Note**
>
> Les tailles d'instance doivent avoir été définies sur vos Outposts. Assurez-vous que vous avez suffisamment de capacité pour trois instances de la taille disponible sur vos Outposts pour la durée de vie de votre cluster local. Pour obtenir la liste des types d'instances Amazon EC2 disponibles, consultez les sections Calcul et stockage dans les fonctionnalités des racks AWS Outposts.

Nombre de modèles	Taille des instances du plan de contrôle Kubernetes
1-20	`large`
21–100	`xlarge`
101–250	`2xlarge`
251–500	`4xlarge`

Le stockage pour le plan de contrôle Kubernetes nécessite 246 Go de stockage Amazon EBS pour chaque cluster local afin de répondre aux exigences IOPS requises pour `etcd`. Lorsque le cluster local est créé, les volumes Amazon EBS sont alloués automatiquement pour vous.

Placement du plan de contrôle

Lorsque vous ne spécifiez aucun groupe de placement avec la propriété `OutpostConfig.ControlPlanePlacement.GroupName`, les instances Amazon EC2 provisionnées pour votre plan de contrôle Kubernetes ne font l'objet d'aucune contrainte de placement matériel spécifique sur la capacité sous-jacente disponible sur votre Outpost.

Vous pouvez utiliser des groupes de placement pour répondre aux exigences de haute disponibilité de votre cluster Amazon EKS local sur un Outpost. En spécifiant un groupe de placement lors de la création du cluster, vous influencez le placement des instances de plan de contrôle Kubernetes. Les instances sont réparties sur du matériel sous-jacent indépendant (racks ou hôtes), minimisant ainsi l'impact des instances corrélées en cas de panne matérielle.

Le type de répartition que vous pouvez configurer dépend du nombre de racks Outpost dont vous disposez dans votre déploiement.

- Déploiements avec un ou deux racks physiques au sein d'un seul Outpost logique : au moins trois hôtes doivent être configurés avec le type d'instance que vous choisissez pour vos instances de plan de contrôle Kubernetes. Avec un groupe de placement par répartition utilisant la répartition au niveau des hôtes, toutes les instances de plan de contrôle Kubernetes s'exécutent sur des hôtes distincts au sein des racks sous-jacents disponibles dans votre déploiement Outpost.
- Déploiements avec au moins trois racks physiques au sein d'un seul Outpost logique : au moins trois hôtes doivent être configurés avec le type d'instance que vous choisissez pour vos instances de plan de contrôle Kubernetes. Avec un groupe de placement par répartition utilisant la répartition au niveau des racks, toutes les instances de plan de contrôle Kubernetes s'exécutent sur des racks distincts dans votre déploiement Outpost. Vous pouvez également utiliser le groupe de placement par répartition au niveau des hôtes comme décrit dans l'option précédente.

La création du groupe de placement souhaité vous incombe. Vous spécifiez le groupe de placement lorsque vous appelez l'API `CreateCluster`. Pour plus d'informations sur les groupes de placement et leur création, consultez la section Groupes de placement dans le Guide de l'utilisateur Amazon EC2.

- Lorsqu'un groupe de placement est spécifié, de la capacité doit être disponible sur votre Outpost pour créer un cluster Amazon EKS local. La capacité varie selon que vous utilisez le type de répartition au niveau des hôtes ou des racks. Si la capacité est insuffisante, le cluster reste à l'état `Creating`. Vous pouvez voir le message `Insufficient Capacity Error` dans le champ de santé de la réponse de l'API DescribeCluster. Vous devez libérer de la capacité pour que le processus de création puisse progresser.
- Lors des mises à jour de la plateforme et de la version du cluster local Amazon EKS, les instances de plan de contrôle Kubernetes de votre cluster sont remplacées par de nouvelles instances selon une stratégie de mise à jour continue. Au cours de ce processus de remplacement, chaque instance de plan de contrôle est arrêtée et son emplacement est libéré. Une nouvelle instance mise à jour est provisionnée à sa place. L'instance mise à jour peut être placée dans l'emplacement qui a été libéré. Si l'emplacement est occupé par une autre instance indépendante et que la capacité disponible est insuffisante pour répondre aux exigences topologiques de répartition, le cluster reste à l'état `Updating`. Vous pouvez voir le message `Insufficient Capacity Error` correspondant dans le champ de santé de la réponse de l'API DescribeCluster. Vous devez libérer

de la capacité pour que le processus de mise à jour puisse progresser et rétablir les niveaux de haute disponibilité antérieurs.

- Vous pouvez créer un maximum de 500 groupes de placement par compte dans chaque région AWS. Pour plus d'informations, consultez la section Règles générales et limitations dans le Guide de l'utilisateur Amazon EC2.

Résoudre les problèmes liés aux clusters Amazon EKS locaux sur Outposts AWS

Cette rubrique traite de certaines erreurs courantes que vous pourriez rencontrer lorsque vous utilisez des clusters locaux, ainsi que des solutions. Les clusters locaux sont similaires aux clusters Amazon EKS dans le cloud, mais il existe quelques différences dans la façon dont ils sont gérés par Amazon EKS.

> **Important**
>
> Ne mettez jamais fin à une instance de `Kubernetes` plan de contrôle du cluster local EKS gérée exécutée sur Outpost, sauf instruction explicite du Support. AWS La fermeture de ces instances présente un risque pour la disponibilité du service de cluster local, notamment la perte du cluster local en cas de fermeture simultanée de plusieurs instances. Les instances du `Kubernetes` plan de contrôle du cluster local EKS sont identifiées par la balise `eks-local:controlplane-name` sur la console d' EC2 instance.

Comportement de l'API

Les clusters locaux sont créés via l'API Amazon EKS, mais sont exécutés de manière asynchrone. Cela signifie que les demandes adressées à l'API Amazon EKS aboutissent immédiatement pour les clusters locaux. Toutefois, ces demandes peuvent aboutir, effectuer une interruption immédiate en raison d'erreurs de validation d'entrée, ou échouer et comporter des erreurs de validation descriptives. Ce comportement est similaire à celui de l'API Kubernetes.

Les clusters locaux ne passent pas à l'état `FAILED`. Amazon EKS tente de rapprocher l'état du cluster de l'état souhaité demandé par l'utilisateur de manière continue. Par conséquent, un cluster local peut rester dans l'état `CREATING` pendant une longue période, jusqu'à ce que le problème sous-jacent soit résolu.

Décrire le champ de santé du cluster

Les problèmes de cluster locaux peuvent être découverts à l'aide de la commande describe-cluster Amazon EKS AWS CLI. Les problèmes liés aux clusters locaux sont signalés par le champ `cluster.health` de la réponse de la commande `describe-cluster`. Le message contenu dans ce champ inclut un code d'erreur, un message descriptif et une ressource associée IDs. Ces informations sont uniquement disponibles via l'API et la AWS CLI Amazon EKS. Dans l'exemple suivant, remplacez *my-cluster* par le nom de votre cluster local.

```
aws eks describe-cluster --name my-cluster --query 'cluster.health'
```

L'exemple qui suit illustre un résultat.

```
{
    "issues": [
        {
            "code": "ConfigurationConflict",
            "message": "The instance type 'm5.large' is not supported in Outpost 'my-
outpost-arn'.",
            "resourceIds": [
                "my-cluster-arn"
            ]
        }
    ]
}
```

Si le problème est irréparable, vous devrez peut-être supprimer le cluster local et en créer un nouveau. Par exemple, essayez d'allouer un cluster avec un type d'instance qui n'est pas disponible sur votre Outpost. Le tableau suivant répertorie les erreurs courantes liées à l'état.

Scénario d'erreur	Code	Message	ResourceIds
Les sous-réseaux fournis sont introuvables.	`ResourceNotFound`	`The subnet ID` *subnet-id* `does not exist`	Tous les sous-réseaux fournis IDs
Les sous-réseaux fournis n'appartiennent pas au même VPC.	`ConfigurationConflict`	`Subnets specified must`	Tous les sous-réseaux fournis IDs

Scénario d'erreur	Code	Message	ResourceIds
		belong to the same VPC	
Certains sous-réseaux fournis n'appartiennent pas à l'Outpost spécifié.	ConfigurationConflict	Subnet *subnet-id* expected to be in *outpost-arn* , but is in *other-outpost-arn*	ID de sous-réseau problématique
Certains sous-réseaux fournis n'appartiennent à aucun Outpost.	ConfigurationConflict	Subnet *subnet-id* is not part of any Outpost	ID de sous-réseau problématique
Certains sous-réseaux fournis ne disposent pas de suffisamment d'adresses libres pour créer des interfaces réseau Elastic pour les instances du plan de contrôle.	ResourceLimitExceeded	The specified subnet does not have enough free addresses to satisfy the request.	ID de sous-réseau problématique
Le type d'instance du plan de contrôle spécifié n'est pas pris en charge sur votre Outpost.	ConfigurationConflict	The instance type *type* is not supported in Outpost *outpost-arn*	ARN de cluster

Scénario d'erreur	Code	Message	ResourceIds
Vous avez mis fin à une EC2 instance Amazon du plan de contrôle ou vous avez `run-instance` réussi, mais l'état observé change de`Terminated`. Cela peut se produire pendant un certain temps après que votre Outpost se soit reconnecté et que des erreurs internes d'Amazon EBS aient entraîné l'échec d'un flux de travail EC2 interne d'Amazon.	`InternalFailure`	`EC2 instance state "Terminated" is unexpected`	ARN de cluster
Vous avez une capacité insuffisante sur votre Outpost. Cela peut également se produire lors de la création d'un cluster si un avant-poste est déconnecté de la AWS région.	`ResourceLimitExceeded`	`There is not enough capacity on the Outpost to launch or start the instance.`	ARN de cluster
Votre compte a dépassé votre quota de groupes de sécurité.	`ResourceLimitExceeded`	Message d'erreur renvoyé par l' EC2 API Amazon	ID du VPC cible

Scénario d'erreur	Code	Message	ResourceIds
Votre compte a dépassé votre quota d'interface réseau Elastic.	`ResourceLimitExceeded`	Message d'erreur renvoyé par l' EC2 API Amazon	l'ID du sous-réseau cible
Les instances du plan de contrôle n'étaient pas accessibles via AWS Systems Manager. Pour la résolution, consultez the section called "Les instances du plan de contrôle ne sont pas accessibles via AWS Systems Manager".	`ClusterUnreachable`	Les instances du plan de contrôle Amazon EKS ne sont pas accessibles via SSM. Vérifiez votre SSM et votre configuration réseau, et consultez la documentation de résolution des problèmes EKS sur les Outposts.	EC2 Instance Amazon IDs
Une erreur s'est produite lors de l'obtention de détails pour un groupe de sécurité géré ou une interface réseau Elastic.	Basé sur le code d'erreur du EC2 client Amazon.	Message d'erreur renvoyé par l' EC2 API Amazon	Tous les groupes de sécurité sont gérés IDs
Une erreur s'est produite lors de l'autorisation ou de la révocation des règles d'entrée du groupe de sécurité. Cela s'applique à la fois aux groupes de sécurité du cluster et du plan de contrôle.	Basé sur le code d'erreur du EC2 client Amazon.	Message d'erreur renvoyé par l' EC2 API Amazon	ID de groupe de sécurité problémat ique

Scénario d'erreur	Code	Message	ResourceIds
Une erreur s'est produite lors de la suppression d'une interface réseau Elastic d'une instance du plan de contrôle.	Basé sur le code d'erreur du EC2 client Amazon.	Message d'erreur renvoyé par l' EC2 API Amazon	ID d'interface réseau Elastoc problématique

Le tableau suivant répertorie les erreurs provenant d'autres AWS services présentées dans le champ de santé de la `describe-cluster` réponse.

Code EC2 d'erreur Amazon	Code de problème de santé du cluster	Description
`AuthFailure`	`AccessDenied`	Cette erreur peut se produire pour différentes raisons. La raison la plus courante est que vous avez accidentellement supprimé une balise que le service utilise pour réduire la portée de la politique de rôle lié à un service du plan de contrôle. Dans ce cas, Amazon EKS ne pourra plus gérer ni surveiller ces AWS ressources.
`UnauthorizedOperation`	`AccessDenied`	Cette erreur peut se produire pour différentes raisons. La raison la plus courante est que vous avez accidentellement supprimé une balise que le service utilise pour réduire la portée de la politique de rôle lié à un service du plan

Code EC2 d'erreur Amazon	Code de problème de santé du cluster	Description
		de contrôle. Dans ce cas, Amazon EKS ne pourra plus gérer ni surveiller ces AWS ressources.
`InvalidSubnetID.NotFound`	`ResourceNotFound`	Cette erreur se produit lorsque l'ID de sous-réseau pour les règles d'entrée d'un groupe de sécurité est introuvable.
`InvalidPermission.NotFound`	`ResourceNotFound`	Cette erreur se produit lorsque les autorisations pour les règles d'entrée d'un groupe de sécurité ne sont pas correctes.
`InvalidGroup.NotFound`	`ResourceNotFound`	Cette erreur se produit lorsque le groupe des règles d'entrée d'un groupe de sécurité est introuvable.
`InvalidNetworkInterfaceID.NotFound`	`ResourceNotFound`	Cette erreur se produit lorsque l'ID de l'interface réseau pour les règles d'entrée d'un groupe de sécurité est introuvable.
`InsufficientFreeAddressesInSubnet`	`ResourceLimitExceeded`	Cette erreur se produit lorsque le quota de ressources du sous-réseau est dépassé.
`InsufficientCapacityOnOutpost`	`ResourceLimitExceeded`	Cette erreur se produit lorsque le quota de capacité de l'avant-poste est dépassé.

Code EC2 d'erreur Amazon	Code de problème de santé du cluster	Description
`NetworkInterfaceLimitExceeded`	`ResourceLimitExceeded`	Cette erreur se produit lorsque le quota de l'interface réseau Elastic est dépassé.
`SecurityGroupLimitExceeded`	`ResourceLimitExceeded`	Cette erreur se produit lorsque le quota du groupe de sécurité est dépassé.
`VcpuLimitExceeded`	`ResourceLimitExceeded`	Cela est observé lors de la création d'une EC2 instance Amazon dans un nouveau compte. L'erreur peut être similaire à ce qui suit : "`You have requested more vCPU capacity than your current vCPU limit of 32 allows for the instance bucket that the specified instance type belongs to. Please visit http://aws.amazon.com/contact-us/ec2-request to request an adjustment to this limit.`"
`InvalidParameterValue`	`ConfigurationConflict`	Amazon EC2 renvoie ce code d'erreur si le type d'instance spécifié n'est pas pris en charge sur l'Outpost.
Toutes les autres erreurs	`InternalFailure`	Aucune

Impossible de créer ou de modifier des clusters

Les clusters locaux nécessitent des autorisations et des politiques différentes de celles des clusters Amazon EKS qui sont hébergés dans le cloud. Lorsqu'un cluster ne parvient pas à se créer et génère une `InvalidPermissions` erreur, vérifiez que le rôle de cluster que vous utilisez est associé à la politique EKSLocal OutpostClusterPolicy gérée par Amazon. Tous les autres appels d'API nécessitent le même ensemble d'autorisations que les clusters Amazon EKS dans le cloud.

Le cluster est bloqué à l'état **`CREATING`**

Le temps nécessaire à la création d'un cluster local varie en fonction de plusieurs facteurs. Ces facteurs comprennent la configuration de votre réseau, la configuration d'Outpost et la configuration du cluster. En général, un cluster local est créé et passe à l'état `ACTIVE` dans les 15 à 20 minutes. Si un cluster local reste dans l'état `CREATING`, vous pouvez appeler `describe-cluster` pour obtenir des informations sur la cause dans le champ de sortie `cluster.health`.

Les problèmes les plus courants sont les suivants :

- Votre cluster ne peut pas se connecter à l'instance du plan de contrôle depuis la AWS région dans laquelle se trouve Systems Manager. Vous pouvez le vérifier en appelant `aws ssm start-session --target` *`instance-id`* depuis un hôte bastion de la région. Si cette commande ne fonctionne pas, vérifiez si Systems Manager est exécuté sur l'instance du plan de contrôle. Une autre solution consiste à supprimer le cluster, puis à le recréer.
- Les instances du plan de contrôle ne peuvent pas être créées en raison des autorisations de clé KMS pour les volumes EBS. Avec les clés KMS gérées par l'utilisateur pour les volumes EBS chiffrés, les instances du plan de contrôle seront interrompues si la clé n'est pas accessible. Si les instances sont mises hors service, passez à une clé KMS AWS gérée ou assurez-vous que votre politique de clé gérée par l'utilisateur accorde les autorisations nécessaires au rôle de cluster.
- Il se peut que les instances du plan de contrôle de Systems Manager n'aient pas accès à Internet. Vérifiez si le sous-réseau que vous avez fourni lors de la création du cluster possède une passerelle NAT et un VPC avec une passerelle Internet. Utilisez l'analyseur d'accessibilité VPC pour vérifier si l'instance du plan de contrôle peut atteindre la passerelle Internet. Pour plus d'informations, consultez Getting started with VPC Reachability Analyzer (langue française non garantie).
- Le rôle ARN que vous avez fourni ne contient pas de politiques. Vérifiez si la politique AWS gérée : Amazon EKSLocal OutpostClusterPolicy a été supprimée du rôle. Cela peut également se produire si une AWS CloudFormation pile est mal configurée.

- Tous les sous-réseaux fournis doivent être associés au même Outpost et pouvoir communiquer entre eux. Lorsque plusieurs sous-réseaux sont spécifiés lors de la création d'un cluster, Amazon EKS tente de répartir les instances du plan de contrôle sur plusieurs sous-réseaux.
- Les groupes de sécurité gérés par Amazon EKS sont appliqués sur l'interface réseau Elastic. Cependant, d'autres éléments de configuration tels que les règles de pare-feu NACL peuvent entrer en conflit avec les règles de l'interface réseau Elastic.

La configuration du VPC et du DNS du sous-réseau est mal configurée ou manquante

Consultez l'article Créer un VPC et des sous-réseaux pour les clusters Amazon EKS sur Outposts. AWS

Le cluster est bloqué à l'état **`UPDATING`**

Amazon EKS met automatiquement à jour tous les clusters locaux existants vers les dernières versions de plateforme correspondant à leur version mineure Kubernetes. Pour plus d'informations sur les versions de plateforme, veuillez consulter the section called "Versions de la plateforme EKS".

Lors d'un déploiement automatique de la version de plateforme, l'état du cluster passe à `UPDATING`. Le processus de mise à jour consiste à remplacer toutes les instances du plan de contrôle Kubernetes par de nouvelles instances contenant les derniers correctifs de sécurité et les dernières corrections de bogues publiés pour la version mineure respective de Kubernetes. En général, le processus de mise à jour d'une plateforme de cluster local s'effectue en moins de 30 minutes, après quoi le cluster revient à son état `ACTIVE`. Si un cluster local reste dans l'état `UPDATING` pendant une période prolongée, vous pouvez appeler `describe-cluster` pour obtenir des informations sur la cause dans le champ de sortie `cluster.health`.

Amazon EKS garantit qu'au moins deux des trois instances du plan de contrôle Kubernetes sont des nœuds de cluster sains et opérationnels afin de maintenir la disponibilité du cluster local et d'éviter toute interruption de service. Si un cluster local est bloqué dans un état `UPDATING`, c'est généralement parce qu'un problème d'infrastructure ou de configuration empêche de garantir la disponibilité minimale de deux instances dans le cas où le processus se poursuit. Le processus de mise à jour s'arrête donc afin de protéger le service de cluster local contre toute interruption.

Il est important de dépanner un cluster local bloqué dans son état `UPDATING` et de traiter la cause profonde afin que le processus de mise à jour puisse s'achever et restaurer le cluster local sur `ACTIVE` avec la haute disponibilité de 3 instances du plan de contrôle Kubernetes.

Ne mettez fin à aucune `Kubernetes` instance de cluster local EKS gérée sur Outposts, sauf indication explicite du Support AWS . Ceci est particulièrement important pour les clusters locaux bloqués dans un état `UPDATING`, car il y a une forte probabilité qu'un autre nœud du plan de contrôle ne soit pas entièrement opérationnel et que la fermeture de la mauvaise instance puisse entraîner une interruption du service et un risque de perte de données du cluster local.

Les problèmes les plus courants sont les suivants :

- Une ou plusieurs instances du plan de contrôle ne parviennent pas à se connecter à System Manager en raison d'une modification de la configuration réseau depuis la création initiale du cluster local. Vous pouvez le vérifier en appelant `aws ssm start-session --target` *`instance-id`* depuis un hôte bastion de la région. Si cette commande ne fonctionne pas, vérifiez si Systems Manager est exécuté sur l'instance du plan de contrôle.
- Les nouvelles instances du plan de contrôle ne peuvent pas être créées en raison des autorisations de clé KMS pour les volumes EBS. Avec les clés KMS gérées par l'utilisateur pour les volumes EBS chiffrés, les instances du plan de contrôle seront interrompues si la clé n'est pas accessible. Si les instances sont mises hors service, passez à une clé KMS AWS gérée ou assurez-vous que votre politique de clé gérée par l'utilisateur accorde les autorisations nécessaires au rôle de cluster.
- Les instances du plan de contrôle Systems Manager ont peut-être perdu leur accès à Internet. Vérifiez si le sous-réseau que vous avez fourni lors de la création du cluster possède une passerelle NAT et un VPC avec une passerelle Internet. Utilisez l'analyseur d'accessibilité VPC pour vérifier si l'instance du plan de contrôle peut atteindre la passerelle Internet. Pour plus d'informations, consultez Getting started with VPC Reachability Analyzer (langue française non garantie). Si vos réseaux privés ne disposent pas d'une connexion Internet sortante, assurez-vous que tous les points de terminaison d'un VPC et le point de terminaison de passerelle requis sont toujours présents dans le sous-réseau régional de votre cluster (voir the section called "Accès au sous-réseau pour les services AWS").
- Le rôle ARN que vous avez fourni ne contient pas de politiques. Vérifiez si la politique AWS gérée : Amazon n'EKSLocalOutpostClusterPolicya pas été supprimée du rôle.
- L'une des nouvelles instances du plan de contrôle Kubernetes a peut-être subi une défaillance inattendue lors du démarrage. Veuillez envoyer une demande au centre d'assistance AWS pour obtenir des conseils supplémentaires sur le dépannage et la collecte des journaux dans ce cas exceptionnel.

Impossible de joindre des nœuds à un cluster

- Problèmes AMI :
  - Vous utilisez une AMI incompatible. Seul Amazon Linux 2 optimisé pour Amazon EKS AMIs est pris en charge (`amazon-linux-2amazon-linux-2-gpu,,amazon-linux-2-arm64`). Si vous essayez de joindre AL2 023 nœuds à EKS LocalClusters on AWS Outposts, les nœuds ne parviennent pas à rejoindre le cluster. Pour plus d'informations, consultez Créer des nœuds Amazon Linux sur AWS Outposts.
  - Si vous avez utilisé un AWS CloudFormation modèle pour créer vos nœuds, assurez-vous qu'il n'utilise pas une AMI non prise en charge.
- L'authentificateur AWS IAM est manquant `ConfigMap` : s'il est absent, vous devez le créer. Pour plus d'informations, consultez the section called "Appliquer la `ConfigMap aws-auth` à votre cluster".
- Le mauvais groupe de sécurité est utilisé – Assurez-vous d'utiliser `eks-cluster-sg-`*`cluster-name`*`-`*`uniqueid`* pour le groupe de sécurité de vos composants master. Le groupe de sécurité sélectionné est modifié AWS CloudFormation pour autoriser un nouveau groupe de sécurité chaque fois que la pile est utilisée.
- Suite à des étapes inattendues liées à un VPC de lien privé – Des données CA incorrectes (`--b64-cluster-ca`) ou le mauvais point de terminaison de l'API (`--apiserver-endpoint`) sont transmis.

Collecte des journaux

Lorsqu'un avant-poste est déconnecté de la AWS région à laquelle il est associé, le cluster Kubernetes continuera probablement à fonctionner normalement. Toutefois, si le cluster ne fonctionne pas correctement, suivez les étapes de résolution des problèmes décrites dans Préparer les clusters Amazon EKS locaux sur AWS Outposts pour les déconnexions réseau. Si vous rencontrez d'autres problèmes, contactez le AWS Support. AWS Support peut vous aider à télécharger et à exécuter un outil de collecte de journaux. Ainsi, vous pouvez collecter des journaux à partir de vos instances de plan de contrôle du cluster Kubernetes et les envoyer au AWS support technique pour une enquête plus approfondie.

Les instances du plan de contrôle ne sont pas accessibles via AWS Systems Manager

Lorsque les instances du plan de contrôle Amazon EKS ne sont pas accessibles via AWS Systems Manager (Systems Manager), Amazon EKS affiche le message d'erreur suivant pour votre cluster.

```
Amazon EKS control plane instances are not reachable through SSM. Please verify your
 SSM and network configuration, and reference the EKS on Outposts troubleshooting
 documentation.
```

Pour résoudre ce problème, assurez-vous que votre VPC et vos sous-réseaux répondent aux exigences de la section Créer un VPC et des sous-réseaux pour les clusters Amazon EKS sur AWS Outposts et que vous avez suivi les étapes décrites dans Configuration du gestionnaire de session dans le guide de l'utilisateur de Systems Manager. AWS

Créez des nœuds Amazon Linux sur AWS Outposts

> **Important**
>
> Amazon EKS Local Clusters on Outposts prend uniquement en charge les nœuds créés à partir du système Amazon Linux 2023 optimisé pour Amazon EKS suivant : AMIs
>
> - Amazon Linux 2023 standard (`amazon-linux-2023/x86_64/standard`)
> - Nvidia Amazon Linux 2023 accéléré (`amazon-linux-2023/x86_64/nvidia`)
> - Neuron accéléré Amazon Linux 2023 () `amazon-linux-2023/x86_64/neuron`
>
> AWS fin du support pour EKS, AL2 optimisé et AL2 accéléré AMIs, à compter du 26 novembre 2025. Bien que vous puissiez continuer à utiliser EKS AL2 AMIs après la date end-of-support (EOS) (26 novembre 2025), EKS ne publiera plus de nouvelles versions ou mises à jour de Kubernetes AL2 AMIs, y compris les versions mineures, les correctifs et les corrections de bogues après cette date. Consultez ceci pour plus d'informations sur la AL2 dépréciation.

Cette rubrique décrit comment lancer des groupes Auto Scaling de nœuds Amazon Linux sur un Outpost qui s'enregistrent auprès de votre cluster Amazon EKS. Le cluster peut se trouver dans le AWS cloud ou dans un avant-poste.

- Un Outpost existant. Pour plus d'informations, consultez What is AWS Outposts.
- Un cluster Amazon EKS existant. Pour déployer un cluster sur le AWS cloud, consultezthe section called “Créer un cluster”. Pour déployer un cluster sur un Outpost, consultez the section called “Exécuter des clusters locaux”.

- Supposons que vous créiez vos nœuds dans un cluster sur le AWS Cloud et que vous disposiez de sous-réseaux dans la AWS région où AWS Outposts, AWS Wavelength ou Local AWS Zones sont activés. Alors, ces sous-réseaux ne doivent pas avoir été transmis lors de la création de votre cluster. Si vous créez vos nœuds dans un cluster sur un Outpost, vous devez avoir transmis un sous-réseau Outpost lors de la création de votre cluster.
- (Recommandé pour les clusters sur le AWS cloud) Le module complémentaire Amazon VPC CNI pour Kubernetes est configuré avec son propre rôle IAM auquel est attachée la politique IAM nécessaire. Pour de plus amples informations, veuillez consulter the section called “Configuration pour IRSA”. Les clusters locaux ne prennent pas en charge les rôles IAM pour les comptes de service.

Vous pouvez créer un groupe de nœuds Amazon Linux autogéré avec `eksctl` ou AWS Management Console (avec un AWS CloudFormation modèle). Vous pouvez également utiliser Terraform.

Vous pouvez créer un groupe de nœuds autogéré pour un cluster local à l'aide des outils décrits dans cette page :

- the section called “`eksctl`”
- the section called “AWS Management Console”

⚠ Important

- Le groupe de nœuds autogéré inclut EC2 les instances Amazon dans votre compte. Ces instances ne sont pas mises à niveau automatiquement lorsque la version du plan de contrôle est mise à jour par vous ou par Amazon EKS en votre nom. Un groupe de nœuds autogérés n'a aucune indication dans la console qu'il doit être mis à jour. Vous pouvez afficher la version `kubelet` installée sur un nœud en sélectionnant le nœud dans la boîte de dialogue Nœuds dans la Présentation de votre cluster pour déterminer quels nœuds doivent être mis à jour. Vous devez mettre à jour les nœuds manuellement. Pour de plus amples informations, veuillez consulter the section called “Méthodes de mise à jour”.
- Les certificats utilisés par kubelet sur vos nœuds autogérés sont émis avec une durée de validité d'un an. Par défaut, la rotation des certificats n'est pas activée (voir : https:// kubernetes). io/docs/reference/config-api/kubelet-config.v1beta1/# kubelet-config-k 8 s-io-v 1beta1-KubeletConfiguration), cela signifie que si vous avez un nœud autogéré qui fonctionne depuis plus d'un an, il ne pourra plus s'authentifier auprès de l'API Kubernetes.

- À titre de bonne pratique, nous recommandons aux clients de mettre régulièrement à jour leurs groupes de nœuds autogérés afin de recevoir CVEs les correctifs de sécurité de la dernière AMI optimisée pour Amazon EKS. La mise à jour de l'AMI utilisée dans les groupes de nœuds autogérés déclenche également la recréation des nœuds et garantit qu'ils ne rencontrent pas de problèmes liés à l'expiration des certificats kubelet.
- Vous pouvez également activer la rotation des certificats clients (voir : https://kubernetes).io/docs/tasks/tls/certificate-rotation/) lors de la création des groupes de nœuds autogérés pour s'assurer que les certificats kubelet sont renouvelés lorsque le certificat actuel approche de son expiration.

`eksctl`

Pour lancer des nœuds Linux autogérés à l'aide de **`eksctl`**

1. Installez la version `0.215.0` ou une version ultérieure de l'outil de ligne de `eksctl` commande installé sur votre appareil ou AWS CloudShell. Pour installer ou mettre à jour `eksctl`, veuillez consulter Installation dans la documentation de `eksctl`.
2. Si votre cluster se trouve AWS dans le cloud et que la politique IAM gérée par Amazoneks_CNI_Policy est attachée au rôle IAM de votre nœud Amazon EKS, nous vous recommandons de l'attribuer à un rôle IAM que vous associez plutôt au compte de service Kubernetes. `aws-node` Pour de plus amples informations, veuillez consulter the section called "Configuration pour IRSA". Si votre cluster se trouve sur votre Outpost, la politique doit être associée à votre rôle de nœud.
3. La commande suivante crée un groupe de nœuds dans un cluster existant. Le cluster doit avoir été créé à l'aide d'`eksctl`. Remplacer *`al-nodes`* avec un nom pour votre groupe de nœuds. Le nom du groupe de nœuds ne peut pas dépasser 63 caractères. Il doit commencer par une lettre ou un chiffre, mais peut également inclure des tirets et des traits de soulignement pour les autres caractères. Remplacez *`my-cluster`* par le nom de votre cluster. Un nom ne peut contenir que des caractères alphanumériques (sensibles à la casse) et des traits d'union. Il doit commencer par un caractère alphanumérique et ne peut pas dépasser 100 caractères. Le nom doit être unique dans la AWS région et le AWS compte dans lesquels vous créez le cluster. Si votre cluster existe sur un Outpost, remplacez *`id`* par l'ID d'un sous-réseau Outpost. Si votre cluster existe dans le AWS cloud, remplacez-le *`id`* par l'ID d'un sous-réseau que vous n'avez pas spécifié lors de la création de votre cluster. Remplacez la valeur d'exemple restante par vos propres valeurs. Par défaut, les nœuds sont créés avec la même version de Kubernetes que le plan de contrôle.

Remplacez *instance-type* par un type d'instance disponible sur votre Outpost.

*my-key*Remplacez-le par le nom de votre paire de EC2 clés Amazon ou de votre clé publique. Cette clé est utilisée pour SSH dans vos nœuds après leur lancement. Si vous ne possédez pas encore de paire de EC2 clés Amazon, vous pouvez en créer une dans le AWS Management Console. Pour plus d'informations, consultez les paires de EC2 clés Amazon dans le guide de EC2 l'utilisateur Amazon.

Créez votre groupe de nœuds avec la commande suivante.

```
eksctl create nodegroup --cluster my-cluster --name al-nodes --node-type instance-
type \
    --nodes 3 --nodes-min 1 --nodes-max 4 --managed=false \
    --node-volume-type gp2 --subnet-ids subnet-id \
    --node-ami-family AmazonLinux2023
```

Si votre cluster est déployé sur le AWS cloud :

- Le groupe de nœuds que vous déployez peut attribuer des adresses `IPv4` aux pods à partir d'un bloc CIDR différent de celui de l'instance. Pour de plus amples informations, veuillez consulter the section called "Mise en réseau personnalisée".
- Le groupe de nœuds que vous déployez ne nécessite pas d'accès Internet sortant. Pour de plus amples informations, veuillez consulter the section called "Clusters privés".

Pour obtenir la liste complète de toutes les options disponibles et des paramètres par défaut, consultez la section Assistance AWS Outposts dans la documentation `eksctl`.

- Si les nœuds ne parviennent pas à rejoindre le cluster, consultez the section called "Les nœuds ne parviennent pas à joindre le cluster" les sections Résolution des problèmes liés aux clusters et nœuds Amazon EKS et Résoudre les problèmes liés the section called "Impossible de joindre des nœuds à un cluster" aux clusters Amazon EKS locaux sur AWS Outposts.
- L'exemple qui suit illustre un résultat. Plusieurs lignes sont affichées pendant la création des nœuds. L'une des dernières lignes de sortie est similaire à la ligne d'exemple suivante.

  ```
  [#]  created 1 nodegroup(s) in cluster "my-cluster"
  ```

4. (Facultatif) Déployez un exemple d'application pour tester votre cluster et les nœuds Linux.

AWS Management Console

Étape 1 : lancer les nœuds Linux autogérés à l'aide de la AWS Management Console

1. Téléchargez la dernière version du AWS CloudFormation modèle.

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2025-11-24/
amazon-eks-outpost-nodegroup.yaml
```

2. Ouvrez la AWS CloudFormation console.
3. Choisissez Create stack (Créer une pile), puis sélectionnez Avec de nouvelles ressources (standard).
4. Pour Spécifier un modèle, sélectionnez Upload a template file (Télécharger un fichier de modèle), puis sélectionnez Choose file (Choisir un fichier). Sélectionnez le fichier `amazon-eks-nodegroup.yaml` que vous avez téléchargé à une étape précédente, puis sélectionnez Next (Suivant).
5. Dans la page Specify stack details (Spécifier les détails de la pile), saisissez les paramètres suivants, puis choisissez Next (Suivant) :
   - Nom de pile : Choisissez un nom de pile pour votre AWS CloudFormation pile. Par exemple, vous pouvez l'appeler *`al-nodes`*. Un nom ne peut contenir que des caractères alphanumériques (sensibles à la casse) et des traits d'union. Il doit commencer par un caractère alphanumérique et ne peut pas dépasser 100 caractères. Le nom doit être unique dans la AWS région et le AWS compte dans lesquels vous créez le cluster.
   - ApiServerEndpoint: Entrez le point de terminaison du serveur d'API Kubernetes, visible dans la console EKS ou via l'API. DescribeCluster
   - ClusterName: Entrez le nom de votre cluster. Si ce nom ne correspond pas à celui de votre cluster, vos nœuds ne peuvent pas rejoindre le cluster.
   - ClusterId: Entrez l'identifiant attribué au cluster par le service EKS. Visible via DescribeCluster l'API. Si cet identifiant ne correspond pas à votre identifiant de cluster, vos nœuds ne peuvent pas rejoindre le cluster.
   - CertificateAuthority: Entrez la chaîne codée en base64 de l'autorité de certification Kubernetes. Visible dans la console EKS ou via DescribeCluster l'API.
   - ServiceCidr: Entrez le CIDR des services Kubernetes. Visible dans la console EKS ou via DescribeCluster l'API.
   - ClusterControlPlaneSecurityGroup: Choisissez la SecurityGroupsvaleur à partir de la AWS CloudFormation sortie que vous avez générée lors de la création de votre VPC.

Les étapes suivantes montrent une opération permettant de récupérer le groupe applicable.

i. Ouvrez la console Amazon EKS.

ii. Choisissez le nom du cluster.

iii. Choisissez l'onglet Networking (Mise en réseau).

iv. Utilisez la valeur Groupes de sécurité supplémentaires comme référence lorsque vous effectuez une sélection ClusterControlPlaneSecurityGroupdans la liste déroulante.

- NodeGroupName: Entrez le nom de votre groupe de nœuds. Ce nom peut être utilisé ultérieurement pour identifier le groupe de nœuds autoscaling qui est créé pour vos nœuds.
- NodeAutoScalingGroupMinSize: Entrez le nombre minimum de nœuds que votre groupe Auto Scaling de nœuds peut atteindre.
- NodeAutoScalingGroupDesiredCapacity: Entrez le nombre de nœuds que vous souhaitez atteindre lors de la création de votre pile.
- NodeAutoScalingGroupMaxSize: Entrez le nombre maximum de nœuds que votre groupe Auto Scaling de nœuds peut atteindre.
- NodeInstanceType: Choisissez un type d'instance pour vos nœuds. Si votre cluster s'exécute sur le AWS Cloud, pour plus d'informations, consultezthe section called "Types d'instances Amazon EC2". Si votre cluster s'exécute sur un Outpost, vous ne pouvez sélectionner qu'un type d'instance disponible sur votre Outpost.
- NodeImageIdSSMParam: prérempli avec le paramètre Amazon EC2 Systems Manager d'une AMI récemment optimisée pour Amazon EKS pour une version variable de Kubernetes. Pour utiliser une autre version mineure de Kubernetes prise en charge avec Amazon EKS, remplacez *1.XX* par une autre version prise en charge. Nous vous recommandons de spécifier la même version de Kubernetes que celle de votre cluster.

  Pour utiliser une AMI accélérée optimisée pour Amazon EKS, mettez à jour *NodeImageIdSSMParam* la valeur selon le paramètre SSM souhaité. Découvrez comment récupérer l'AMI EKS IDs à partir de SSM ici.

> **Note**
>
> Le nœud Amazon EKS AMIs est basé sur Amazon Linux. Vous pouvez suivre les événements liés à la sécurité ou à la confidentialité pourAmazon Linux dans le centre de sécurité Amazon Linux en sélectionnant l'onglet correspondant à la version souhaitée. Vous pouvez également vous abonner au flux RSS applicable. Les événements de

sécurité et de confidentialité incluent une présentation du problème, les packages concernés et la manière de mettre à jour vos instances pour résoudre le problème.

- NodeImageId: (Facultatif) Si vous utilisez votre propre AMI personnalisée (au lieu d'une AMI optimisée pour Amazon EKS), entrez un ID d'AMI de nœud pour votre AWS région. Si vous spécifiez une valeur ici, elle remplace toutes les valeurs du NodeImageIdSSMParamchamp.
- NodeVolumeSize: Spécifiez une taille de volume racine pour vos nœuds, en GiB.
- NodeVolumeType: Spécifiez un type de volume racine pour vos nœuds.
- KeyName: Entrez le nom d'une paire de clés Amazon EC2 SSH que vous pourrez utiliser pour vous connecter via SSH à vos nœuds après leur lancement. Si vous ne possédez pas encore de paire de EC2 clés Amazon, vous pouvez en créer une dans le AWS Management Console. Pour plus d'informations, consultez les paires de EC2 clés Amazon dans le guide de EC2 l'utilisateur Amazon.

> **Note**
>
> Si vous ne fournissez pas de paire de clés ici, la création de la AWS CloudFormation pile échoue.

- Désactiver IMDSv1 : par défaut, chaque nœud prend en charge les versions 1 du service de métadonnées d'instance (IMDSv1) et IMDSv2. Vous pouvez le désactiver IMDSv1. Pour empêcher l'utilisation des futurs nœuds et pods du groupe de nœuds IMDSv1, définissez Disable IMDSv1 sur true. Pour de plus amples informations au sujet d'IMDS, consultez Configuration du service des métadonnées d'instance. Pour plus d'informations sur la façon d'en restreindre l'accès sur vos nœuds, consultez Restreindre l'accès au profil d'instance affecté au composant master.
- VpcId: Entrez l'ID du VPC que vous avez créé. Avant de choisir un VPC, examinez les exigences et les considérations relatives au VPC.
- Subnets : si votre cluster se trouve sur un Outpost, choisissez au moins un sous-réseau privé dans votre VPC. Avant de choisir les sous-réseaux, consultez Exigences et considérations requises pour les sous-réseaux. Vous pouvez consulter les sous-réseaux privés en ouvrant le lien de chaque sous-réseau depuis l'onglet Networking (Mise en réseau) de votre cluster.

6. Sélectionnez les choix que vous souhaitez sur la page Configure stack options (Configurer les options de la pile), puis choisissez Next (Suivant).
7. Cochez la case à gauche de J'accuse réception AWS CloudFormation susceptible de créer des ressources IAM. , puis choisissez Create stack.

8. Lorsque la création de votre pile est terminée, sélectionnez la pile dans la console et choisissez Outputs (Sorties).
9. Enregistrez le NodeInstanceRolepour le groupe de nœuds créé. Vous en aurez besoin lors de la configuration de vos nœuds pour Amazon EKS.

Étape 2 : autoriser les nœuds à rejoindre votre cluster

1. Vérifiez si vous avez déjà appliqué le `ConfigMap` `aws-auth`.

```
kubectl describe configmap -n kube-system aws-auth
```

2. Si vous voyez un `ConfigMap` `aws-auth`, mettez-le à jour si nécessaire.
   a. Ouvrez le `ConfigMap` pour le modifier.

```
kubectl edit -n kube-system configmap/aws-auth
```

   b. Ajoutez une nouvelle entrée `mapRoles` si nécessaire. Définissez la `rolearn` valeur sur la NodeInstanceRolevaleur que vous avez enregistrée lors de la procédure précédente.

```
[...]
data:
  mapRoles: |
    - rolearn: <ARN of instance role (not instance profile)>
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
[...]
```

   c. Enregistrez le fichier et quittez votre éditeur de texte.
3. Si vous avez reçu un message d'erreur indiquant « `Error from server (NotFound): configmaps "aws-auth" not found` », appliquez le stock `ConfigMap`.
   a. Téléchargez la mappe de configuration.

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/
aws-auth-cm.yaml
```

b. Dans le `aws-auth-cm.yaml` fichier, définissez la `rolearn` NodeInstanceRolevaleur que vous avez enregistrée lors de la procédure précédente. Pour ce faire, utilisez un éditeur de texte ou remplacez *`my-node-instance-role`* et exécutez la commande suivante :

```
sed -i.bak -e 's|<ARN of instance role (not instance profile)>|my-node-instance-
role|' aws-auth-cm.yaml
```

c. Appliquez la configuration. L'exécution de cette commande peut prendre quelques minutes.

```
kubectl apply -f aws-auth-cm.yaml
```

4. Observez le statut de vos nœuds et attendez qu'ils obtiennent le statut `Ready`.

```
kubectl get nodes --watch
```

Saisissez `Ctrl+C` pour revenir à une invite de shell.

> **Note**
>
> Si vous recevez d'autres erreurs concernant les types d'autorisations ou de ressources, consultez the section called “Accès non autorisé ou refusé (`kubectl`)” dans la rubrique relative à la résolution des problèmes.

Si les nœuds ne parviennent pas à rejoindre le cluster, consultez the section called “Les nœuds ne parviennent pas à joindre le cluster” les sections Résolution des problèmes liés aux clusters et nœuds Amazon EKS et Résoudre les problèmes liés the section called “Impossible de joindre des nœuds à un cluster” aux clusters Amazon EKS locaux sur AWS Outposts.

5. Installez le pilote CSI Amazon EBS. Pour plus d'informations, consultez la section Installation sur GitHub. Dans la section Set up driver permission (Configurer les autorisations du pilote), assurez-vous de suivre les instructions de l'option Using IAM instance profile (Utilisation du profil d'instance IAM). Vous devez utiliser la classe de stockage `gp2`. La classe de stockage `gp3` n'est pas prise en charge.

Pour créer une classe de stockage `gp2` sur votre cluster, suivez les étapes suivantes.

a. Exécutez la commande ci-dessous pour créer un fichier`gp2-storage-class.yaml`.

```
cat >gp2-storage-class.yaml <<EOF
```

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  annotations:
    storageclass.kubernetes.io/is-default-class: "true"
  name: ebs-sc
provisioner: ebs.csi.aws.com
volumeBindingMode: WaitForFirstConsumer
parameters:
  type: gp2
  encrypted: "true"
allowVolumeExpansion: true
EOF
```

b. Appliquez le manifeste à votre cluster.

```
kubectl apply -f gp2-storage-class.yaml
```

6. (Nœuds GPU uniquement) Si vous avez choisi un type d'instance GPU et une AMI accélérée optimisée pour Amazon EKS, vous devez appliquer le plug-in d'appareil NVIDIA pour Kubernetes DaemonSet sur votre cluster. *vX.X.X*Remplacez-le par la s-device-plugin version Nvidia/K8 de votre choix avant d'exécuter la commande suivante.

```
kubectl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/vX.X.X/
deployments/static/nvidia-device-plugin.yml
```

Étape 3 : actions supplémentaires

1. (Facultatif) Déployez un exemple d'application pour tester votre cluster et les nœuds Linux.
2. Si votre cluster est déployé sur un Outpost, ignorez cette étape. Si votre cluster est déployé sur le AWS Cloud, les informations suivantes sont facultatives. Si la politique IAM gérée AmazonEKS_CNI_Policy est associée à votre rôle IAM de nœud Amazon EKS, nous vous recommandons de l'attribuer à un rôle IAM que vous associez au compte de service `aws-node` Kubernetes à la place. Pour de plus amples informations, veuillez consulter the section called "Configuration pour IRSA".

Présentation de l’intelligence artificielle (IA) et du Machine Learning (ML) sur Amazon EKS

Amazon Elastic Kubernetes Service (EKS) est une plateforme Kubernetes gérée qui permet aux entreprises de déployer, gérer et adapter les charges de travail liées à l’intelligence artificielle et au machine learning (ML) avec une flexibilité et un contrôle inégalés. Construit sur l'écosystème open source Kubernetes, EKS vous permet de tirer parti de votre expertise Kubernetes existante, tout en s'intégrant parfaitement aux outils et services open source. AWS

Que vous entraîniez des modèles à grande échelle, que vous exécutiez des inférences en ligne en temps réel ou que vous déployiez des applications d'IA génératives, EKS fournit les performances, l'évolutivité et la rentabilité qu'exigent vos AI/ML projets.

Pourquoi choisir EKS pour l’IA/ML ?

EKS est une plateforme Kubernetes gérée qui vous aide à déployer et à gérer des charges de travail complexes. AI/ML Construit sur l'écosystème open source Kubernetes, il s'intègre AWS aux services, fournissant le contrôle et l'évolutivité nécessaires aux projets avancés. Pour les équipes qui découvrent AI/ML les déploiements, les compétences Kubernetes existantes sont transférées directement, ce qui permet une orchestration efficace de plusieurs charges de travail.

EKS prend en charge tout, depuis la personnalisation des systèmes d’exploitation jusqu’à la mise à l’échelle des ressources informatiques, et sa base open source favorise la flexibilité technologique, préservant ainsi le choix pour les décisions futures en matière d’infrastructure. La plate-forme fournit les performances et les options de réglage requises par les AI/ML charges de travail, en prenant en charge des fonctionnalités telles que :

- Contrôle complet du cluster pour affiner les coûts et les configurations sans abstractions cachées
- Latence inférieure à la seconde pour les charges de travail d’inférence en temps réel en production
- Personnalisations avancées telles que le multi-instance GPUs, les stratégies multi-cloud et le réglage au niveau du système d'exploitation
- Possibilité de centraliser les charges de travail à l'aide d'EKS en tant qu'orchestrateur unifié entre les pipelines AI/ML

Principaux cas d'utilisation

Amazon EKS fournit une plate-forme robuste pour un large éventail de AI/ML charges de travail, prenant en charge diverses technologies et modèles de déploiement :

- Inférence en temps réel (en ligne) : EKS fournit des prévisions immédiates sur les données entrantes, telles que la détection des fraudes, avec une latence inférieure à une seconde à l'aide d'outils tels que TorchServeTriton Inference Server et sur les instances KServeAmazon EC2 Inf1 et Inf2. Ces charges de travail bénéficient d'une mise à l'échelle dynamique avec Karpenter et KEDA, tout en tirant parti d'Amazon EFS pour le découpage des modèles entre les pods. Amazon ECR Pull Through Cache (PTC) accélère les mises à jour des modèles, et les volumes de données Bottlerocket avec les volumes optimisés Amazon EBS garantissent un accès rapide aux données.
- Formation générale sur les modèles : Les organisations utilisent EKS pour former des modèles complexes sur de grands ensembles de données sur de longues périodes à l'aide de Kubeflow Training Operator, Ray Serve et Torch Distributed Elastic sur des instances Amazon EC2 P4d et Amazon EC2 Trn1. Ces charges de travail sont prises en charge par la planification par lots avec des outils tels que Volcano, Yunikorn et Kueue. Amazon EFS permet de partager les points de contrôle des modèles, tandis qu'Amazon S3 gère les politiques relatives au cycle de vie import/export des modèles pour la gestion des versions.
- Fénération à enrichissement contextuel (RAG) : EKS gère les chatbots de support client et les applications similaires en intégrant des processus de récupération et de génération. Ces charges de travail utilisent souvent des outils tels qu'Argo Workflows et Kubeflow pour l'orchestration, des bases de données vectorielles comme Pinecone, Weaviate ou Amazon OpenSearch, et exposent les applications aux utilisateurs via l'Application Load Balancer Controller (LBC). NVIDIA NIM optimise l'utilisation du GPU, tandis que Prometheus et Grafana surveillent l'utilisation des ressources.
- Déploiement de modèles d'IA générative : les entreprises déploient des services de création de contenu en temps réel sur EKS, tels que la génération de texte ou d'images, à l'aide de Ray Serve, vLLM et Triton Inference Server sur Amazon EC2 G5 et les accélérateurs Inferentia. Ces déploiements optimisent les performances et l'utilisation de la mémoire pour les modèles à grande échelle. JupyterHubpermet le développement itératif, Gradius fournit des interfaces Web simples et le pilote S3 Mountpoint CSI permet de monter des compartiments S3 en tant que systèmes de fichiers pour accéder à des fichiers de modèles volumineux.
- Inférence par lots (hors ligne) : les organisations traitent efficacement de grands jeux de données grâce à des tâches planifiées avec AWS Batch ou Volcano. Ces charges de travail utilisent souvent des instances Inf1 et Inf2 pour les AWS puces Inferentia, des EC2 instances Amazon EC2 G4dn

pour NVIDIA T4 ou des instances de processeur c5 et c6i GPUs, maximisant ainsi l'utilisation des ressources pendant les heures creuses pour les tâches d'analyse. Le SDK AWS Neuron et les pilotes GPU NVIDIA optimisent les performances, tout en MIG/TS permettant le partage du GPU. Les solutions de stockage incluent Amazon S3, Amazon EFS et FSx pour Lustre, avec des pilotes CSI pour différentes classes de stockage. La gestion des modèles s'appuie sur des outils tels que Kubeflow Pipelines, Argo Workflows et Ray Cluster, tandis que la surveillance est assurée par Prometheus, Grafana et des outils de surveillance des modèles personnalisés.

Études de cas

Les clients choisissent Amazon EKS pour diverses raisons, telles que l'optimisation de l'utilisation des GPU ou l'exécution de charges de travail d'inférence en temps réel avec une latence inférieure à la seconde, comme le montrent les études de cas suivantes. Pour obtenir la liste de toutes les études de cas relatives à Amazon EKS, consultez la rubrique Témoignages clients AWS.

- Unitary traite quotidiennement 26 millions de vidéos à l'aide de l'IA pour la modération de contenu, ce qui nécessite une inférence à haut débit et à faible latence. L'entreprise a réduit de 80 % le temps de démarrage des conteneurs, garantissant ainsi une réponse rapide aux événements de mise à l'échelle lorsque le trafic fluctue.
- Miro, la plateforme de collaboration visuelle qui compte 70 millions d'utilisateurs dans le monde, a annoncé une réduction de 80 % de ses coûts informatiques par rapport à ses anciens clusters Kubernetes autogérés.
- Synthesia, qui propose un service de création vidéo par IA générative permettant aux clients de créer des vidéos réalistes à partir de suggestions textuelles, a multiplié par 30 le débit de l'entraînement de ses modèles de machine learning.
- Harri, fournisseur de technologies RH pour le secteur de l'hôtellerie, a accéléré de 90 % sa mise à l'échelle en réponse aux pics de demande et a réduit ses coûts de calcul de 30 % en migrant vers les processeurs AWS Graviton.
- Ada Support, une entreprise spécialisée dans l'automatisation du service client grâce à l'intelligence artificielle, a réduit ses coûts informatiques de 15 % tout en augmentant son efficacité informatique de 30 %.
- Snorkel AI, qui permet aux entreprises de créer et d'adapter des modèles de base et des modèles linguistiques à grande échelle, a réalisé plus de 40 % d'économies en mettant en œuvre des mécanismes de mise à l'échelle intelligents pour ses ressources GPU.

Commencez à utiliser Machine learning sur EKS

Pour commencer à planifier et à utiliser les plateformes et les charges de travail de Machine Learning sur EKS AWS dans le cloud, passez à la the section called "Ressources" section.

Exécution de charges de travail d'inférence en ligne en temps réel sur Amazon EKS

> **Tip**
>
> Inscrivez-vous aux prochains AI/ML ateliers Amazon EKS.

Cette section est conçue pour vous aider à déployer et à exploiter des charges de travail d'inférence en ligne en temps réel sur Amazon Elastic Kubernetes Service (EKS). Vous y trouverez des conseils sur la création de clusters optimisés avec des nœuds accélérés par GPU, l'intégration de AWS services de stockage et de mise à l'échelle automatique, le déploiement d'exemples de modèles à des fins de validation, ainsi que des considérations architecturales clés telles que le découplage des tâches du processeur et du processeur graphique, la sélection des types d'instance appropriés AMIs et la garantie d'une exposition à faible latence des points de terminaison d'inférence.

Rubriques

- Guide de configuration de clusters basé sur les meilleures pratiques pour l'inférence en temps réel sur Amazon EKS
- Démarrage rapide : inférence LLM à haut débit avec vLLM sur Amazon EKS

Guide de configuration de clusters basé sur les meilleures pratiques pour l'inférence en temps réel sur Amazon EKS

> **Tip**
>
> Inscrivez-vous aux prochains AI/ML ateliers Amazon EKS.

Introduction

Ce guide propose une procédure pratique pour configurer un cluster Amazon Elastic Kubernetes Service (EKS) optimisé pour les charges de travail d'inférence en ligne en temps réel, en intégrant les meilleures pratiques élaborées par des experts. AWS Il utilise une architecture EKS Quickstart bien ancrée, c'est-à-dire un ensemble de pilotes, de types d'instances et de configurations sélectionnés conformément aux AWS meilleures pratiques en matière de modèles, d'accélérateurs et de scalabilité. Cette approche vous évite d'avoir à sélectionner les paramètres du cluster, ce qui vous permet de disposer rapidement d'un cluster fonctionnel et préconfiguré. En cours de route, nous déploierons des exemples de charges de travail pour valider votre configuration, expliquer les concepts architecturaux clés (tels que le découplage des tâches liées au processeur des calculs gourmands en GPU), répondre aux questions courantes (par exemple, pourquoi choisir Bottlerocket AMI ?) AL2023 , et décrivez les prochaines étapes pour étendre les capacités de votre cluster.

Conçu spécifiquement pour les ingénieurs du Machine Learning (ML) et de l'intelligence artificielle (IA), les administrateurs de plateformes, les opérateurs et les data/AI spécialistes qui découvrent l'écosystème EKS pour la AWS première fois, ce guide suppose une connaissance de Kubernetes mais aucune expérience préalable d'EKS. Il est conçu pour vous aider à comprendre les étapes et les processus nécessaires à la mise en place et au fonctionnement des charges de travail d'inférence en ligne en temps réel. Ce guide explique les éléments essentiels de la création d'un cluster d'inférence à nœud unique, notamment le provisionnement des ressources GPU, l'intégration du stockage pour les artefacts du modèle, la sécurisation de l'accès aux AWS services et la mise en évidence des points de terminaison d'inférence. Tout au long du document, l'accent est mis sur une conception résiliente et à faible latence pour les applications destinées aux utilisateurs, telles que la détection des fraudes, les chatbots en temps réel et l'analyse des sentiments dans les systèmes de commentaires des clients.

Dans ce guide, nous nous concentrons exclusivement sur la mise en place d'un point de départ fondamental et normatif à l'aide d'instances G5 EC2. Si vous recherchez des configurations de clusters ou des end-to-end flux de travail AWS spécifiques à Inferentia, consultez nos the section called "Configurer des clusters d'inférence avec Inferentia" ateliers dans. the section called "Ressources"

Avant de commencer

Avant de commencer, assurez-vous d'avoir effectué les tâches suivantes :

- Configuration de votre environnement pour Amazon EKS

- Installation de la dernière version d'eksctl
- Installer Helm
- (Facultatif) Installer Docker
- (Facultatif) Installer l'interface CLI NGC

Architecture

L'inférence en ligne en temps réel désigne le processus consistant à utiliser un modèle de machine learning entraîné pour générer des prédictions ou des résultats sur des flux de données entrants avec une latence minimale. Par exemple, il permet la détection de fraudes en temps réel, la classification d'images ou la génération de graphes de connaissances en réponse aux entrées des utilisateurs. L'architecture d'un système d'inférence en ligne en temps réel fournit des prédictions de machine learning à faible latence dans les applications destinées aux utilisateurs en dissociant le traitement du trafic Web lié au processeur des calculs d'IA gourmands en ressources GPU. Ce processus s'inscrit généralement dans un écosystème applicatif plus large et comprend souvent des services dorsaux, frontaux, vectoriels et de modélisation, en mettant l'accent sur des composants spécialisés afin de permettre une évolutivité indépendante, un développement parallèle et une résilience face aux défaillances. L'isolation des tâches d'inférence sur du matériel GPU dédié et l'exploitation d'interfaces telles que APIs et WebSockets garantissent une simultanéité élevée, un traitement rapide de modèles tels que les transformateurs et des interactions avec les utilisateurs via le frontend. Notez que, bien que les bases de données vectorielles et les pipelines de génération à enrichissement contextuel (RAG) jouent souvent un rôle important dans les systèmes d'inférence en temps réel, ces composants ne sont pas abordés dans ce guide. Au minimum, une architecture typique comprend souvent :

- Service frontal : Sert d'interface utilisateur, gère la logique côté client, affiche le contenu dynamique et facilite les interactions en temps réel. Il communique avec le service principal pour lancer des demandes d'inférence et afficher les résultats, en adressant souvent des demandes au service principal qui l'utilise WebSockets pour les mises à jour en continu ou pour l'échange de données structurées. APIs Ce service ne nécessite généralement pas d'équilibreur de charge dédié, car il peut être hébergé sur des réseaux de diffusion de contenu (CDNs), comme AWS CloudFront pour les actifs statiques, ou servi directement à partir de serveurs Web, le dimensionnement étant géré via des groupes d'auto-dimensionnement si nécessaire pour le contenu dynamique.
- Service principal : agit en tant qu'orchestrateur de l'application, gérant la logique métier telle que l'authentification des utilisateurs, la validation des données et la coordination des services (par exemple, via APIs pour les RESTful points de terminaison ou WebSockets pour les connexions

persistantes). Il communique avec le service d'inférence, s'adapte indépendamment sur le multicœur CPUs et la RAM pour gérer le trafic Web élevé sans compter sur celui-ci GPUs, et nécessite souvent un équilibreur de charge (tel que Application Load AWS Balancer ou Network Load Balancer) pour répartir les demandes entrantes sur plusieurs instances, en particulier dans les scénarios de concurrence élevée. Un contrôleur d'accès peut également gérer les règles d'accès externe et de routage afin d'améliorer la sécurité et la gestion du trafic.

- Service d'inférence : sert de base aux calculs d'intelligence artificielle, fonctionnant GPUs avec suffisamment de VRAM (par exemple, 8 à 12 Go pour des modèles tels que Distilbert) pour effectuer des intégrations vectorielles, l'extraction de connaissances et l'inférence de modèles (par exemple, exposée pour les demandes par lots ou APIs pour le streaming en temps réel) à l'aide de modèles personnalisés ou open source WebSockets . Cet isolement empêche les conflits de dépendance, permet la mise à jour des modèles sans durée d'indisponibilité du service et facilite la mise à l'échelle horizontale avec équilibrage de charge pour plusieurs requêtes simultanées. Pour exposer efficacement le modèle de service, celui-ci est généralement installé derrière un équilibreur de charge chargé de répartir les charges de travail liées au GPU entre les instances répliquées, tandis qu'une ressource ou un contrôleur d'entrée (tel qu'un contrôleur d'entrée ALB intégré AWS) gère le routage externe, la terminaison SSL et le transfert basé sur le chemin afin de garantir un accès sécurisé et efficace sans surcharger le personnel. GPUs

Présentation de la solution

Les systèmes d'inférence en ligne en temps réel nécessitent une architecture résiliente et hautement performante, capable d'offrir une latence ultra-faible tout en gérant des pics de trafic imprévisibles et volumineux. Cette présentation de la solution explique comment les AWS composants suivants fonctionnent ensemble dans le cluster Amazon EKS que nous allons créer pour garantir que notre cluster est capable d'héberger et de gérer des modèles d'apprentissage automatique qui fournissent des prévisions immédiates sur les données en temps réel avec un délai minimal pour les utilisateurs finaux.

- Instances Amazon G5 EC2 — Pour les tâches d'inférence gourmandes en ressources GPU, nous utilisons les types d'instances g5.xlarge et g5.2xlarge G5 EC2, qui disposent d'un (1) GPU NVIDIA A10G avec 24 Go de mémoire (par exemple, 8 milliards de paramètres en FP16). Basés sur l'architecture NVIDIA Ampere, ces GPU sont équipés de GPU NVIDIA A10G Tensor Core et de processeurs AMD EPYC de 2e génération. Ils prennent en charge 4 à 8 vCPU, une bande passante réseau pouvant atteindre 10 Gbit/s et 250 à 450 Go de stockage SSD NVMe local, garantissant ainsi un transfert rapide des données et une puissance de calcul suffisante pour les modèles complexes. Ils sont donc parfaits pour les tâches d'inférence à faible latence et haut débit.

Le choix d'un type d'instance EC2 dépend de l'application, de votre modèle (par exemple, modèle d'image, de vidéo ou de texte) et de vos exigences en matière de latence et de débit. Par exemple, si vous utilisez un modèle d'image et/ou de vidéo, vous pouvez utiliser des instances P5 EC2 pour obtenir une latence optimale en temps réel. Nous vous recommandons de commencer avec des instances G5 EC2, car elles constituent un bon point de départ pour être rapidement opérationnel, puis d'évaluer si elles sont adaptées à vos charges de travail à l'aide de tests de performance. Pour les cas plus avancés, pensez aux instances G6 EC2.

- Instances Amazon EC2 M7g — Pour les tâches gourmandes en ressources CPU telles que le prétraitement des données, le traitement des requêtes API, l'hébergement du contrôleur Karpenter, les modules complémentaires et d'autres composants système, nous utilisons le type d'instance EC2 m5.xlarge M7g. Les instances m7G sont des instances ARM dotées de 4 VCPUs, de 16 Go de mémoire, d'une bande passante réseau allant jusqu'à 12,5 Gbit/s et alimentées par des processeurs Graviton3. AWS Le choix d'un type d'instance EC2 dépend de l'application et des exigences de votre charge de travail en matière de calcul, de mémoire et de capacité de mise à l'échelle. Pour les charges de travail optimisées pour le calcul, vous pouvez envisager les instances EC2 C7g, qui utilisent également des processeurs Graviton3, mais qui sont optimisées pour offrir des performances de calcul supérieures à celles des instances M7g dans certains cas d'utilisation. Par ailleurs, les nouvelles instances EC2 C8g (lorsqu'elles sont disponibles) offrent des performances de calcul jusqu'à 30 % supérieures à celles des instances C7g. Nous vous recommandons de commencer par les instances M7g EC2 pour leur rentabilité et leur compatibilité avec un large éventail de charges de travail (par exemple, serveurs d'applications, microservices, serveurs de jeux, magasins de données de taille moyenne), puis d'évaluer si elles conviennent à vos charges de travail à l'aide de tests de performance.
- Pilote CSI Amazon S3 Mountpoint : Pour les charges de travail sur des instances à GPU unique où plusieurs pods partagent un GPU (par exemple, plusieurs pods planifiés sur le même nœud pour utiliser ses ressources GPU), nous utilisons le pilote CSI Mountpoint S3 afin d'optimiser l'utilisation de la mémoire, ce qui est essentiel pour des tâches telles que l'inférence de modèles volumineux dans des configurations peu complexes et sensibles aux coûts. Il expose les compartiments Amazon S3 comme un système de fichiers de type POSIX accessible au cluster Kubernetes, ce qui permet aux pods d'inférence de lire les artefacts du modèle (par exemple, les poids du modèle) directement en mémoire sans avoir à les télécharger au préalable, et d'entrer des jeux de données à l'aide d'opérations de fichiers standard. De plus, S3 dispose d'une capacité de stockage pratiquement illimitée et accélère les charges de travail d'inférence gourmandes en données. Le choix d'un pilote CSI de stockage dépend de l'application et des exigences de votre charge de travail en matière de débit et de latence. Bien que le pilote CSI FSx pour OpenZFS offre une latence inférieure à la milliseconde pour les volumes persistants partagés aléatoires I/O

ou entièrement compatibles POSIX entre les nœuds, nous vous recommandons de commencer par le pilote CSI Mountpoint S3 en raison de son évolutivité, de ses coûts réduits pour les grands ensembles de données et de son intégration au stockage d'objets géré par S3 pour les modèles d'inférence lourds (par exemple, les entrées du modèle de streaming), puis d'évaluer s'il convient à vos charges de travail grâce à des tests de performance.

- Agent d'identité EKS Pod — Pour permettre l'accès aux AWS services, nous utilisons l'agent EKS Pod Identity, qui utilise un seul principal de service et facilite les associations de rôles IAM au niveau du pod au sein du cluster Amazon EKS. EKS Pod Identity offre une alternative simplifiée à l'approche traditionnelle IAM Roles for Service Accounts (IRSA) en utilisant un seul principal de service (pods.eks.amazonaws.com) au lieu de s'appuyer sur des fournisseurs OIDC individuels pour chaque cluster, ce qui facilite l'attribution des autorisations. De plus, il permet de réutiliser les rôles dans plusieurs clusters et prend en charge des fonctionnalités avancées telles que les balises de session de rôle IAM et les rôles IAM cibles.
- Agent de surveillance des nœuds EKS : afin de garantir la disponibilité et la fiabilité continues des services d'inférence, nous utilisons l'agent de surveillance des nœuds EKS avec réparation automatique, qui détecte et remplace automatiquement les nœuds défectueux, minimisant ainsi la durée d'indisponibilité. Il surveille en permanence les nœuds pour détecter les problèmes liés au matériel, au noyau, au réseau et au stockage à l'aide de contrôles de santé améliorés (par exemple KernelReady,, NetworkingReady). Pour les nœuds GPU, il détecte les défaillances spécifiques à l'accélérateur, lance une correction en douceur en isolant les nœuds défaillants, attend 10 minutes que les problèmes GPU transitoires se résolvent et remplace les nœuds après 30 minutes en cas de défaillances persistantes.
- AMI Bottlerocket : Afin de fournir une base sécurisée à notre cluster EKS, nous utilisons l'AMI Bottlerocket, qui ne comprend que les composants essentiels nécessaires au fonctionnement des conteneurs et offre des temps de démarrage minimaux pour une mise à l'échelle rapide. Le choix d'une AMI de nœud dépend de l'application et des exigences de personnalisation, de sécurité et de capacité de mise à l'échelle de votre charge de travail. Bien que l'AL2023 AMI offre une plus grande flexibilité pour les installations et les personnalisations au niveau de l'hôte (par exemple, en spécifiant un répertoire de cache dédié dans une configuration PV/PVC sans nœuds supplémentaires), nous vous recommandons de commencer par l'AMI Bottlerocket pour son encombrement réduit et son optimisation intégrée pour les charges de travail conteneurisées (par exemple, microservices, serveurs d'inférence, évolutifs APIs), puis d'évaluer si elle convient à vos charges de travail grâce à des tests de performance.
- AWS Load Balancer Controller (LBC) — Pour exposer les points de terminaison d'inférence en temps réel, nous utilisons le Load Balancer Controller, qui approvisionne et gère automatiquement les équilibreurs de AWS charge d'application () pour le trafic et les équilibreurs de charge réseau

(ALBs) pour le HTTP/HTTPS trafic en fonction des ressources Kubernetes Ingress et Service, permettant ainsi l'intégration de modèles d'inférence avec des clients externes. NLBs TCP/UDP En outre, il prend en charge des fonctionnalités telles que le routage basé sur les chemins pour distribuer les demandes d'inférence sur plusieurs pods ou nœuds, garantissant ainsi l'évolutivité lors des pics de trafic et minimisant la latence grâce à des optimisations AWS natives telles que le multiplexage des connexions et les bilans de santé.

1. Créer votre cluster EKS

Dans cette étape, nous créons un cluster avec des nœuds CPU et un groupe de nœuds gérés à l'aide d'un modèle eksctl AWS CloudFormation alimenté par un modèle eksctl ClusterConfig. L'initialisation du cluster avec uniquement des nœuds CPU nous permet d'utiliser Karpenter exclusivement pour gérer les nœuds gourmands en CPU et en GPU afin d'optimiser l'allocation des ressources à l'aide de Karpenter NodePools , que nous créerons ultérieurement. Pour prendre en charge nos charges de travail d'inférence en temps réel, nous fournissons au cluster l'AMI EKS Bottlerocket, l'agent de surveillance des nœuds EKS, l'agent EKS Pod Identity, le pilote Mountpoint S3 CSI, le contrôleur Load AWS Balancer (LBC) et les pilotes kube-proxy, vpc-cni et coredns. Les instances m7g.xlarge seront utilisées pour les tâches système du processeur, notamment l'hébergement du contrôleur Karpenter, des modules complémentaires et d'autres composants système.

Par défaut, `eksctl` créera un VPC dédié pour le cluster avec un bloc CIDR de `192.168.0.0/16`. Le VPC comprend trois sous-réseaux publics et trois sous-réseaux privés, chacun étant réparti sur trois zones de disponibilité différentes (ou deux AZs dans la `us-east-1` région), ce qui constitue la méthode idéale pour déployer les charges de travail Kubernetes. Le modèle déploie également une passerelle Internet, fournissant un accès Internet aux sous-réseaux publics via des routes par défaut dans leurs tables de routage et une passerelle NAT unique dans l'un des sous-réseaux publics, avec des routes par défaut dans les tables de routage des sous-réseaux privés dirigeant le trafic sortant via la passerelle NAT pour l'accès à Internet. Pour en savoir plus sur cette configuration, consultez Déployer des nœuds dans des sous-réseaux privés.

Vérifiez vos informations d'identification

Vérifiez si vos informations d'identification AWS CLI sont valides et si vous pouvez vous authentifier auprès des AWS services :

```
aws sts get-caller-identity
```

En cas de succès, la CLI renverra des informations sur votre AWS identité (UserId, votre compte et votre ARN).

Vérifier la disponibilité des instances

Les types d'instances G5 ne sont pas disponibles dans toutes les régions. Vérifiez votre région la plus proche. Par exemple :

```
aws ec2 describe-instance-types --instance-types g5.xlarge g5.2xlarge --region us-
east-1
```

Si l'opération réussit, le type d'instance G5 est disponible dans la région que vous avez spécifiée.

L'AMI Bottlerocket n'est pas disponible dans toutes les régions. Vérifiez en récupérant un identifiant AMI Bottlerocket pour votre région la plus proche. Par exemple :

```
aws ssm get-parameter --name /aws/service/bottlerocket/aws-k8s-1.33/arm64/latest/
image_id \
    --region us-east-1 --query "Parameter.Value" --output text
```

Si l'opération réussit, l'AMI Bottlerocket est disponible dans la région que vous avez spécifiée.

Préparez votre environnement

Commencez par définir les variables d'environnement suivantes dans une nouvelle fenêtre de terminal. Remarque : Veillez à remplacer les espaces réservés par vos propres valeurs, notamment le nom du cluster, la région souhaitée, la version de Karpenter et la version de Kubernetes.

> **Tip**
>
> Certaines variables (telles que `${AWS_REGION}` et `${K8S_VERSION}`) sont définies au début du bloc, puis référencées dans les commandes suivantes pour des raisons de cohérence et afin d'éviter les répétitions. Veillez à exécuter les commandes dans l'ordre afin que ces valeurs soient correctement exportées et disponibles pour être utilisées dans les définitions suivantes.

```
export TEMPOUT="$(mktemp)"
export K8S_VERSION=1.33
export KARPENTER_VERSION="1.5.0"
export AWS_REGION="us-east-1"
```

```
export EKS_CLUSTER_NAME="eks-rt-inference-${AWS_REGION}"
export S3_BUCKET_NAME="eks-rt-inference-models-${AWS_REGION}-$(date +%s)"
export NVIDIA_BOTTLEROCKET_AMI="$(aws ssm get-parameter --name /aws/service/
bottlerocket/aws-k8s-${K8S_VERSION}-nvidia/x86_64/latest/image_id --query
 Parameter.Value --output text)"
export STANDARD_BOTTLEROCKET_AMI="$(aws ssm get-parameter --name /aws/service/
bottlerocket/aws-k8s-${K8S_VERSION}/arm64/latest/image_id --query Parameter.Value --
output text)"
export AWS_ACCOUNT_ID="$(aws sts get-caller-identity --query Account --output text)"
export ALIAS_VERSION="$(aws ssm get-parameter --name "/aws/service/eks/optimized-
ami/${K8S_VERSION}/amazon-linux-2023/x86_64/standard/recommended/image_id" --query
 Parameter.Value | xargs aws ec2 describe-images --query 'Images[0].Name' --image-ids |
 sed -r 's/^.*(v[[:digit:]]+).*$/\1/')"
```

Créer les rôles et les politiques requis

Karpenter a besoin de rôles et de politiques IAM spécifiques (par exemple, rôle IAM du contrôleur Karpenter, profil d'instance et politiques) pour gérer les instances EC2 en tant que composants master Kubernetes. Il utilise ces rôles pour effectuer des actions telles que le lancement et la résiliation d'instances EC2, le balisage des ressources et l'interaction avec d'autres services. AWS Créez les rôles et politiques Karpenter à l'aide du fichier cloudformation.yaml de Karpenter :

```
curl -fsSL https://raw.githubusercontent.com/aws/karpenter-provider-aws/v
${KARPENTER_VERSION}/website/content/en/preview/getting-started/getting-started-with-
karpenter/cloudformation.yaml > "${TEMPOUT}" \
&& aws cloudformation deploy \
  --stack-name "Karpenter-${EKS_CLUSTER_NAME}" \
  --template-file "${TEMPOUT}" \
  --capabilities CAPABILITY_NAMED_IAM \
  --parameter-overrides "ClusterName=${EKS_CLUSTER_NAME}"
```

Le AWS LBC a besoin d'une autorisation pour approvisionner et gérer des équilibreurs de AWS charge, par exemple ALBs pour créer des ressources Ingress ou NLBs pour des services de type similaire. `LoadBalancer` Nous définirons cette politique d'autorisation lors de la création du cluster. Lors de la création du cluster, nous créerons le compte de service avec eksctl dans le. ClusterConfig Créer la politique IAM LBC :

```
aws iam create-policy \
  --policy-name AWSLoadBalancerControllerIAMPolicy \
  --policy-document "$(curl -fsSL https://raw.githubusercontent.com/kubernetes-sigs/
aws-load-balancer-controller/v2.14.1/docs/install/iam_policy.json)"
```

Lorsque le pilote CSI Mountpoint S3 est installé, ses DaemonSet modules sont configurés pour utiliser un compte de service pour l'exécution. Le pilote Mountpoint pour Mountpoint S3 CSI nécessite une autorisation pour interagir avec le compartiment Amazon S3 que vous créerez plus loin dans ce guide. Nous définirons cette politique d'autorisation lors de la création du cluster. Lors de la création du cluster, nous créerons le compte de service avec eksctl dans le. ClusterConfig Créez la politique IAM S3 :

```
aws iam create-policy \
    --policy-name S3CSIDriverPolicy \
    --policy-document "{\"Version\": \"2012-10-17\", \"Statement\": [{\"Effect\":
 \"Allow\", \"Action\": [\"s3:GetObject\", \"s3:PutObject\", \"s3:AbortMultipartUpload
\", \"s3:DeleteObject\", \"s3:ListBucket\"], \"Resource\": [\"arn:aws:s3:::
${S3_BUCKET_NAME}\", \"arn:aws:s3:::${S3_BUCKET_NAME}/*\"]}]}"
```

Remarque : Si un rôle portant ce nom existe déjà, donnez-lui un autre nom. Le rôle que nous créons à cette étape est spécifique à votre cluster et à votre compartiment S3.

Créer le cluster

Dans ce modèle, eksctl crée automatiquement un compte de service Kubernetes pour EKS Pod Identity, Node Monitoring Agent, CoreDNS, Kubeproxy et le plug-in VPC CNI. À ce jour, le pilote Mountpoint S3 CSI n'est pas disponible pour EKS Pod Identity. Nous créons donc un rôle IAM pour compte de service (IRSA) et un point de terminaison OIDC. En outre, nous créons un compte de service pour le AWS Load Balancer Controller (LBC). Pour accéder aux nœuds Bottlerocket, eksctl attache automatiquement SSMManaged InstanceCore Amazon pour Bottlerocket afin de permettre des sessions shell sécurisées via SSM.

Dans le même terminal où vous définissez vos variables d'environnement, exécutez le bloc de commande suivant pour créer le cluster :

```
eksctl create cluster -f - <<EOF
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
metadata:
  name: ${EKS_CLUSTER_NAME}
  region: ${AWS_REGION}
  version: "${K8S_VERSION}"
  tags:
    karpenter.sh/discovery: ${EKS_CLUSTER_NAME}
    # Add more tags if needed for billing
```

```
iam:
  # Creates an OIDC endpoint and IRSA service account for the Mountpoint S3 CSI Driver
  # Uses the S3 CSI Driver policy for permissions
  withOIDC: true
  podIdentityAssociations:
  # Creates the pod identity association and service account
  # Uses the Karpenter controller IAM policy for permissions
  - namespace: "kube-system"
    serviceAccountName: karpenter
    roleName: ${EKS_CLUSTER_NAME}-karpenter
    permissionPolicyARNs:
    - arn:aws:iam::${AWS_ACCOUNT_ID}:policy/KarpenterControllerPolicy-
${EKS_CLUSTER_NAME}
  # Creates the pod identity association and service account
  # Uses the {aws} LBC policy for permissions
  - namespace: kube-system
    serviceAccountName: aws-load-balancer-controller
    createServiceAccount: true
    roleName: AmazonEKSLoadBalancerControllerRole
    permissionPolicyARNs:
    - arn:aws:iam::${AWS_ACCOUNT_ID}:policy/AWSLoadBalancerControllerIAMPolicy
iamIdentityMappings:
- arn: "arn:aws:iam::${AWS_ACCOUNT_ID}:role/KarpenterNodeRole-${EKS_CLUSTER_NAME}"
  username: system:node:{{EC2PrivateDNSName}}
  groups:
  - system:bootstrappers
  - system:nodes
managedNodeGroups:
  # Creates 2 CPU nodes for lightweight system tasks
  - name: ${EKS_CLUSTER_NAME}-m7-cpu
    instanceType: m7g.xlarge
    amiFamily: Bottlerocket
    desiredCapacity: 2
    minSize: 1
    maxSize: 10
    labels:
      role: cpu-worker
# Enable automatic Pod Identity associations for VPC CNI Driver, coreDNS, kube-proxy
addonsConfig:
  autoApplyPodIdentityAssociations: true
addons:
  # Installs the S3 CSI Driver addon and creates IAM role
  # Uses the S3 CSI Driver policy for IRSA permissions
  - name: aws-mountpoint-s3-csi-driver
```

```
    attachPolicyARNs:
      - "arn:aws:iam::${AWS_ACCOUNT_ID}:policy/S3CSIDriverPolicy"
  - name: eks-pod-identity-agent
  - name: eks-node-monitoring-agent
  - name: coredns
  - name: kube-proxy
  - name: vpc-cni
EOF
```

Ce processus prend plusieurs minutes. Si vous souhaitez surveiller l'état, consultez la AWS CloudFormationconsole.

2. Vérifiez l'état du nœud et du pod du cluster

Effectuons quelques surveillance de l'état pour nous assurer que le cluster est prêt. Une fois la commande précédente terminée, affichez les types d'instance et vérifiez que vos nœuds système CPU ont atteint l'état `Ready` souhaité à l'aide de la commande suivante :

```
kubectl get nodes -L node.kubernetes.io/instance-type
```

Le résultat attendu devrait ressembler à ceci :

```
NAME                            STATUS   ROLES    AGE     VERSION
 INSTANCE-TYPE
ip-192-168-35-103.ec2.internal   Ready    <none>   12m     v1.33.0-eks-802817d
 m7g.xlarge
ip-192-168-7-15.ec2.internal     Ready    <none>   12m     v1.33.0-eks-802817d
 m7g.xlarge
```

Vérifiez toutes les associations d'identité de pod et la manière dont elles mappent un rôle à un compte de service dans un espace de noms du cluster à l'aide de la commande suivante :

```
eksctl get podidentityassociation --cluster ${EKS_CLUSTER_NAME} --region ${AWS_REGION}
```

La sortie doit indiquer les rôles IAM de Karpenter (« karpenter ») et du AWS LBC (» «). aws-load-balancer-controller

Vérifiez qu' DaemonSets ils sont disponibles :

```
kubectl get daemonsets -n kube-system
```

Le résultat attendu devrait ressembler à ceci :

```
NAME                          DESIRED CURRENT READY UP-TO-DATE AVAILABLE NODE SELECTOR
          AGE
aws-node                      3       3       3     3          3         <none>
          12m
dcgm-server                   0       0       0     0          0
 kubernetes.io/os=linux 12m
eks-node-monitoring-agent     3       3       3     3          3
 kubernetes.io/os=linux 12m
eks-pod-identity-agent        3       3       3     3          3         <none>
          12m
kube-proxy                    3       3       3     3          3         <none>
          12m
s3-csi-node                   2       2       2     2          2
 kubernetes.io/os=linux 12m
```

Vérifiez que tous les addons sont installés sur le cluster :

```
eksctl get addons --cluster ${EKS_CLUSTER_NAME} --region ${AWS_REGION}
```

Le résultat attendu devrait ressembler à ceci :

```
NAME                          VERSION              STATUS    ISSUES    IAMROLE
                                     UPDATE AVAILABLE    CONFIGURATION VALUES    POD
 IDENTITY ASSOCIATION ROLES
aws-mountpoint-s3-csi-driver   v1.15.0-eksbuild.1   ACTIVE    0
 arn:aws:iam::143095308808:role/eksctl-eks-rt-inference-us-east-1-addon-aws-m-Role1-
RAUjk4sJnc0L
coredns                       v1.12.1-eksbuild.2   ACTIVE    0
eks-node-monitoring-agent     v1.3.0-eksbuild.2    ACTIVE    0
eks-pod-identity-agent        v1.3.7-eksbuild.2    ACTIVE    0
kube-proxy                    v1.33.0-eksbuild.2   ACTIVE    0
metrics-server                v0.7.2-eksbuild.3    ACTIVE    0
vpc-cni                       v1.19.5-eksbuild.1   ACTIVE    0
```

3. Installez Karpenter

Installez le contrôleur Karpenter sur les composants master de votre processeur (`cpu-worker`) afin d'optimiser les coûts et de préserver les ressources du GPU. Nous allons l'installer dans l'espace de noms « kube-system » et spécifier le compte de service « karpenter » que nous avons défini lors

de la création du cluster. En outre, cette commande configure le nom du cluster et une file d'attente d'interruption d'instance Spot pour les nœuds du processeur. Karpenter utilisera l'IRSA pour assumer ce rôle IAM.

```
# Logout of helm registry before pulling from public ECR
helm registry logout public.ecr.aws

# Install Karpenter
helm upgrade --install karpenter oci://public.ecr.aws/karpenter/karpenter --version
 "${KARPENTER_VERSION}" --namespace "kube-system" --create-namespace  \
  --set "settings.clusterName=${EKS_CLUSTER_NAME}" \
  --set "settings.interruptionQueue=${EKS_CLUSTER_NAME}" \
  --set controller.resources.requests.cpu=1 \
  --set controller.resources.requests.memory=1Gi \
  --set controller.resources.limits.cpu=1 \
  --set controller.resources.limits.memory=1Gi \
  --set serviceAccount.annotations."eks\.amazonaws\.com/role-arn"="arn:aws:iam::
${AWS_ACCOUNT_ID}:role/${EKS_CLUSTER_NAME}-karpenter" \
  --wait
```

Le résultat attendu devrait ressembler à ceci :

```
Release "karpenter" does not exist. Installing it now.
Pulled: public.ecr.aws/karpenter/karpenter:1.5.0
Digest: sha256:9a155c7831fbff070669e58500f68d7ccdcf3f7c808dcb4c21d3885aa20c0a1c
NAME: karpenter
LAST DEPLOYED: Thu Jun 19 09:57:06 2025
NAMESPACE: kube-system
STATUS: deployed
REVISION: 1
TEST SUITE: None
```

Vérifiez que Karpenter est en cours d'exécution :

```
kubectl get pods -n kube-system -l app.kubernetes.io/name=karpenter
```

Le résultat attendu devrait ressembler à ceci :

```
NAME                       READY   STATUS    RESTARTS   AGE
karpenter-555895dc-865bc   1/1     Running   0          5m58s
karpenter-555895dc-j7tk9   1/1     Running   0          5m58s
```

4. Configurer Karpenter NodePools

Dans cette étape, nous configurons le processeur et le GPU Karpenter NodePools qui s'excluent mutuellement. Le `limits` champ de la NodePool spécification limite le maximum de ressources totales (par exemple, processeur, mémoire GPUs) que chacun NodePool peut consommer sur tous les nœuds provisionnés, empêchant ainsi le provisionnement de nœuds supplémentaires si ces limites sont dépassées. Tout en NodePools prenant en charge de larges catégories d'instances (par exemple`c`,`g`), la spécification de types d'instances spécifiques, de types de capacité et de limites de ressources vous permet d'estimer plus facilement les coûts de vos charges de travail à la demande. Dans ces derniers NodePools, nous utilisons un ensemble diversifié de types d'instances au sein de la famille d'instances G5. Cela permet à Karpenter de sélectionner automatiquement le type d'instance le plus approprié en fonction des demandes de ressources du pod, optimisant ainsi l'utilisation des ressources tout en respectant les limites totales NodePool du pod. Pour en savoir plus, consultez la section Création NodePools.

Configuration du GPU NodePool

Dans ce cadre NodePool, nous définissons des limites de ressources pour gérer le provisionnement des nœuds dotés de capacités GPU. Ces limites sont conçues pour plafonner les ressources totales sur tous les nœuds du pool, permettant jusqu'à 10 instances au total. Chaque instance peut être g5.xlarge (4 v, CPUs 16 GiB de mémoire, 1 GPU) ou g5.2xlarge (8 v, 32 GiB de mémoireCPUs, 1 GPU), à condition que le total de CPUs v ne dépasse pas 80, que la mémoire totale ne dépasse pas 320 Go et que le total ne dépasse pas 10. GPUs Par exemple, le pool peut provisionner 10 instances g5.2xlarge (80 v, CPUs 320 GiB, 10 GPUs) ou 10 instances g5.xlarge (40 v, CPUs 160 GiB, 10), ou une combinaison telle que 5 g5.xlarge et 5 g5.2xlarge (60 v, 240 GiB, 10 GPUs), garantissant ainsi une flexibilité basée sur les demandes de charge de travail tout en respectant les CPUs contraintes de ressources. GPUs

De plus, nous indiquons l'ID de la variante Nvidia de l'AMI Bottlerocket. Enfin, nous avons défini une politique de perturbation visant à supprimer les nœuds vides après 30 minutes (`consolidateAfter: 30m`) et fixé une durée de vie maximale de 30 jours (`expireAfter: 720h`) pour les nœuds afin d'optimiser les coûts et de maintenir l'état des nœuds pour les tâches gourmandes en ressources GPU. Pour en savoir plus, consultez Désactiver la consolidation Karpenter pour les charges de travail sensibles aux interruptions et Utiliser ttlSecondsAfter Finished pour nettoyer automatiquement les tâches Kubernetes.

```
cat <<EOF | envsubst | kubectl apply -f -
apiVersion: karpenter.sh/v1
kind: NodePool
```

```
metadata:
  name: gpu-a10g-inference-g5
spec:
  template:
    metadata:
      labels:
        role: gpu-worker
        gpu-type: nvidia-a10g
    spec:
      requirements:
        - key: node.kubernetes.io/instance-type
          operator: In
          values: ["g5.xlarge", "g5.2xlarge"]
        - key: "karpenter.sh/capacity-type"
          operator: In
          values: ["on-demand"]
      taints:
        - key: nvidia.com/gpu
          value: "true"
          effect: NoSchedule
      nodeClassRef:
        name: gpu-a10g-inference-ec2
        group: karpenter.k8s.aws
        kind: EC2NodeClass
      expireAfter: 720h
  limits:
    cpu: "80"
    memory: "320Gi"
    nvidia.com/gpu: "10"
  disruption:
    consolidationPolicy: WhenEmpty
    consolidateAfter: 30m
---
apiVersion: karpenter.k8s.aws/v1
kind: EC2NodeClass
metadata:
  name: gpu-a10g-inference-ec2
spec:
  amiFamily: Bottlerocket
  amiSelectorTerms:
    - id: ${NVIDIA_BOTTLEROCKET_AMI}
  role: "KarpenterNodeRole-${EKS_CLUSTER_NAME}"
  subnetSelectorTerms:
    - tags:
```

```
        karpenter.sh/discovery: "${EKS_CLUSTER_NAME}"
  securityGroupSelectorTerms:
    - tags:
        karpenter.sh/discovery: "${EKS_CLUSTER_NAME}"
  tags:
    nvidia.com/gpu: "true"
EOF
```

Le résultat attendu devrait ressembler à ceci :

```
nodepool.karpenter.sh/gpu-a10g-inference-g5 created
ec2nodeclass.karpenter.k8s.aws/gpu-a10g-inference-ec2 created
```

Vérifiez que le NodePool est créé et qu'il est sain :

```
kubectl get nodepool gpu-a10g-inference-g5 -o yaml
```

Cherchez `status.conditions` comme `ValidationSucceeded: TrueNodeClassReady: True`, et confirmez `Ready: True` qu' NodePool il est sain.

Configuration du processeur NodePool

Dans ce cadre NodePool, nous avons défini des limites pour prendre en charge environ 50 instances, en fonction d'une charge de travail modérée du processeur (par exemple, 100 à 200 pods) et de quotas de AWS vCPU classiques (par exemple, 128 à 1152). Les limites sont calculées en supposant que l'échelle NodePool devrait atteindre 50 instances m7.xlarge : processeur (4 v CPUs par instance × 50 instances = 200 vCPUs) et mémoire (16 GiB par instance × 50 instances = 800 GiB). Ces limites sont conçues pour plafonner les ressources totales sur tous les nœuds du pool, en autorisant jusqu'à 50 instances de 7 g x large (chacune avec 4 V et CPUs 16 Go de mémoire), à condition que le total de v ne dépasse pas 200 et que la mémoire totale CPUs ne dépasse pas 800 Go.

De plus, nous indiquons l'ID de la variante standard de l'AMI Bottlerocket. Enfin, nous avons défini une politique de perturbation visant à supprimer les nœuds vides après 60 minutes (`consolidateAfter: 60m`) et fixé une durée de vie maximale de 30 jours (`expireAfter: 720h`) pour les nœuds afin d'optimiser les coûts et de maintenir l'état des nœuds pour les tâches gourmandes en ressources GPU. Pour en savoir plus, consultez Désactiver la consolidation Karpenter pour les charges de travail sensibles aux interruptions et Utiliser ttlSecondsAfter Finished pour nettoyer automatiquement les tâches Kubernetes.

```
cat <<EOF | envsubst | kubectl apply -f -
```

```
apiVersion: karpenter.sh/v1
kind: NodePool
metadata:
  name: cpu-inference-m7gxlarge
spec:
  template:
    metadata:
      labels:
        role: cpu-worker
    spec:
      requirements:
        - key: node.kubernetes.io/instance-type
          operator: In
          values: ["m7g.xlarge"]
        - key: karpenter.sh/capacity-type
          operator: In
          values: ["on-demand"]
      taints:
        - key: role
          value: cpu-intensive
          effect: NoSchedule
      nodeClassRef:
        name: cpu-inference-m7gxlarge-ec2
        group: karpenter.k8s.aws
        kind: EC2NodeClass
      expireAfter: 720h
  limits:
    cpu: "200"
    memory: "800Gi"
  disruption:
    consolidationPolicy: WhenEmpty
    consolidateAfter: 60m
---
apiVersion: karpenter.k8s.aws/v1
kind: EC2NodeClass
metadata:
  name: cpu-inference-m7gxlarge-ec2
spec:
  amiFamily: Bottlerocket
  amiSelectorTerms:
    - id: ${STANDARD_BOTTLEROCKET_AMI}
  role: "KarpenterNodeRole-${EKS_CLUSTER_NAME}"
  subnetSelectorTerms:
    - tags:
```

```
        karpenter.sh/discovery: "${EKS_CLUSTER_NAME}"
  securityGroupSelectorTerms:
    - tags:
        karpenter.sh/discovery: "${EKS_CLUSTER_NAME}"
EOF
```

Le résultat attendu devrait ressembler à ceci :

```
nodepool.karpenter.sh/cpu-inference-m7gxlarge created
ec2nodeclass.karpenter.k8s.aws/cpu-inference-m7gxlarge-ec2 created
```

Vérifiez que le NodePool est créé et qu'il est sain :

```
kubectl get nodepool cpu-inference-m7gxlarge -o yaml
```

Cherchez `status.conditions` comme `ValidationSucceeded: TrueNodeClassReady: True`, et confirmez `Ready: True` qu' NodePool il est sain.

5. Déployer un module GPU pour exposer un GPU

Vous avez besoin du plug-in Nvidia Device pour permettre à Kubernetes d'exposer les périphériques GPU au cluster Kubernetes. En général, vous devez déployer le plug-in en tant que tel DaemonSet ; toutefois, l'AMI Bottlerocket préinstalle le plug-in dans le cadre de l'AMI. Cela signifie que lorsque vous utilisez Bottlerocket AMIs, il n'est pas nécessaire de déployer le plugin pour appareil Nvidia. DaemonSet Pour en savoir plus, consultez la section consacrée au plugin Kubernetes Device. GPUs

Déploiement d'un exemple de pod

Karpenter agit de manière dynamique : il provisionne des nœuds GPU lorsqu'une charge de travail (pod) demande des ressources GPU. Pour vérifier que les pods peuvent demander et utiliser GPUs, déployez un pod qui demande la `nvidia.com/gpu` ressource dans ses limites (par exemple,`nvidia.com/gpu: 1`). Pour en savoir plus sur ces étiquettes, consultez Planifier des charges de travail avec des exigences GPU à l'aide d'étiquettes bien connues.

```
cat <<EOF | envsubst | kubectl apply -f -
apiVersion: v1
kind: Pod
metadata:
  name: gpu-nvidia-smi
spec:
  restartPolicy: OnFailure
```

```
  tolerations:
  - key: "nvidia.com/gpu"
    operator: "Exists"
    effect: "NoSchedule"
  nodeSelector:
    role: gpu-worker  # Matches GPU NodePool's label
  containers:
  - name: cuda-container
    image: nvidia/cuda:12.9.1-base-ubuntu20.04
    command: ["nvidia-smi"]
    resources:
      limits:
        nvidia.com/gpu: 1
      requests:
        nvidia.com/gpu: 1
EOF
```

Le résultat attendu devrait ressembler à ceci :

```
pod/gpu-ndivia-smi created
```

Attendez une minute, puis vérifiez si le Pod a le statut « En attente »ContainerCreating, « En cours », puis « Terminé » :

```
kubectl get pod gpu-nvidia-smi -w
```

Vérifiez que le nœud du pod appartient au GPU NodePool :

```
kubectl get node $(kubectl get pod gpu-nvidia-smi -o jsonpath='{.spec.nodeName}') -o
 custom-columns="Name:.metadata.name,Nodepool:.metadata.labels.karpenter\.sh/nodepool"
```

Le résultat attendu devrait ressembler à ceci :

```
Name                              Nodepool
ip-192-168-83-245.ec2.internal   gpu-a10g-inference-g5
```

Consultez les journaux du pod :

```
kubectl logs gpu-nvidia-smi
```

Le résultat attendu devrait ressembler à ceci :

```
Thu Jul 17 04:31:33 2025
+-----------------------------------------------------------------------------------------
+
| NVIDIA-SMI 570.148.08                 Driver Version: 570.148.08         CUDA
 Version: 12.9 |
|-----------------------------------------+----------------------
+----------------------+
| GPU  Name                 Persistence-M | Bus-Id        Disp.A | Volatile Uncorr. ECC
 |
| Fan  Temp   Perf          Pwr:Usage/Cap |         Memory-Usage | GPU-Util  Compute M.
 |
|                                         |                      |               MIG M.
 |
|=========================================+======================
+======================|
|   0  NVIDIA A10G                    On  | 00000000:00:1E.0 Off |                    0
 |
|  0%   30C    P8               9W / 300W |      0MiB / 23028MiB |      0%      Default
 |
|                                         |                      |                  N/A
 |
+-----------------------------------------------------------------------------------------
+

+-----------------------------------------------------------------------------------------
+
| Processes:
 |
|  GPU        GI     CI        PID   Type   Process name                 GPU Memory
 |
|                     ID        ID                                        Usage
 |
|
=========================================================================================|
|  No running processes found
 |
+-----------------------------------------------------------------------------------------
+
```

6. (Facultatif) Préparer et télécharger les artefacts du modèle pour le déploiement

Au cours de cette étape, vous allez déployer un service de modèle pour la classification d'images en temps réel, en commençant par télécharger les poids du modèle vers un compartiment Amazon S3.

À des fins de démonstration, nous utilisons le modèle open source de vision GPUNet-0 de NVIDIA GPUNet, qui prend en charge l'inférence à faible latence sur les images à l'aide de NVIDIA et de TensorRT. GPUs Ce modèle est préentraîné ImageNet, nous permet de classer des objets sur des photos ou des flux vidéo à la volée et est considéré comme un petit modèle avec 11,9 millions de paramètres.

Configuration de votre environnement

Pour télécharger les poids de modèle GPUNet -0 Au cours de cette étape, vous devez accéder au catalogue NGC de NVIDIA et à Docker installé sur votre machine locale. Procédez comme suit pour créer un compte gratuit et configurer la CLI NGC :

- Inscrivez-vous pour obtenir un compte NGC gratuit et générez une clé API à partir du tableau de bord NGC (icône utilisateur > Configuration > Générer une clé API > Générer une clé personnelle > Catalogue NGC).
- Téléchargez et installez la CLI NGC (Linux/macOS/Windows) et configurez la CLI en utilisant :`ngc config set`. Saisissez votre clé API lorsque vous y êtes invité ; définissez org sur `nvidia` et appuyez sur Entrée pour accepter les valeurs par défaut pour les autres. Si l'opération aboutit, vous devriez voir quelque chose ressemblant à : `Successfully saved NGC configuration to /Users/your-username/.ngc/config`.

Vérifier les autorisations du compte de service

Avant de commencer, vérifiez les autorisations du compte de service Kubernetes :

```
kubectl get serviceaccount s3-csi-driver-sa -n kube-system -o yaml
```

Lors de la création du cluster, nous avons associé la CSIDriver politique S3 à un rôle IAM et annoté le compte de service (« s3- csi-driver-sa »). Les pods du pilote CSI Mountpoint S3 héritent des autorisations du rôle IAM lorsqu'ils interagissent avec S3. Le résultat attendu devrait ressembler à ceci :

```
apiVersion: v1
kind: ServiceAccount
metadata:
  annotations:
    eks.amazonaws.com/role-arn: arn:aws:iam::143095308808:role/eksctl-eks-rt-inference-
us-east-1-addon-aws-m-Role1-fpXXjRYdKN8r
  creationTimestamp: "2025-07-17T03:55:29Z"
```

```
  labels:
    app.kubernetes.io/component: csi-driver
    app.kubernetes.io/instance: aws-mountpoint-s3-csi-driver
    app.kubernetes.io/managed-by: EKS
    app.kubernetes.io/name: aws-mountpoint-s3-csi-driver
  name: s3-csi-driver-sa
  namespace: kube-system
  resourceVersion: "2278"
  uid: 50b36272-6716-4c68-bdc3-c4054df1177c
```

Ajouter une tolérance

Le pilote S3 CSI s'exécute en tant que tel DaemonSet sur tous les nœuds. Les pods utilisent le pilote CSI sur ces nœuds pour monter les volumes S3. Pour lui permettre de se programmer sur nos nœuds GPU présentant des anomalies, ajoutez une tolérance aux éléments suivants : DaemonSet

```
kubectl patch daemonset s3-csi-node -n kube-system --type='json' -p='[{"op": "add",
 "path": "/spec/template/spec/tolerations/-", "value": {"key": "nvidia.com/gpu",
 "operator": "Exists", "effect": "NoSchedule"}}]'
```

Le résultat attendu devrait ressembler à ceci :

```
daemonset.apps/s3-csi-node patched
```

Charger les poids du modèle dans S3

Au cours de cette étape, vous allez créer un compartiment Amazon S3, télécharger les poids de modèle GPUNet -0 depuis NVIDIA GPU Cloud (NGC) et les charger dans le compartiment. Ces poids seront accessibles par notre application au moment de l'exécution à des fins d'inférence.

Créez votre compartiment Amazon S3 :

```
aws s3 mb s3://${S3_BUCKET_NAME} --region ${AWS_REGION}
```

Activez la gestion des versions S3 pour le compartiment afin d'éviter que des suppressions et des écrasements accidentels n'entraînent une perte immédiate et permanente des données :

```
aws s3api put-bucket-versioning --bucket ${S3_BUCKET_NAME} --versioning-configuration
 Status=Enabled
```

Appliquez une règle de cycle de vie au compartiment pour supprimer les versions d'objets écrasées ou supprimées 14 jours après leur expiration, supprimer les marqueurs de suppression expirés et supprimer les téléchargements en plusieurs parties incomplets après 7 jours. Pour en savoir plus, consultez les exemples de configurations du cycle de vie S3.

```
aws s3api put-bucket-lifecycle-configuration --bucket $S3_BUCKET_NAME --lifecycle-
configuration '{"Rules":[{"ID":"LifecycleRule","Status":"Enabled","Filter":
{},"Expiration":{"ExpiredObjectDeleteMarker":true},"NoncurrentVersionExpiration":
{"NoncurrentDays":14},"AbortIncompleteMultipartUpload":{"DaysAfterInitiation":7}}]}'
```

Téléchargez les poids du modèle GPUNet -0 sur NGC. Par exemple, sur macOS :

```
ngc registry model download-version nvidia/dle/gpunet_0_pyt_ckpt:21.12.0_amp --dest ~/
downloads
```

> **Note**
>
> Vous devrez peut-être adapter cette commande de téléchargement à votre système d'exploitation. Pour que cette commande fonctionne sur un système Linux, vous devrez probablement créer le répertoire dans le cadre de la commande (par exemple, `mkdir ~/downloads`).

Le résultat attendu devrait ressembler à ceci :

```
{
  "download_end": "2025-07-18 08:22:39",
  "download_start": "2025-07-18 08:22:33",
  "download_time": "6s",
  "files_downloaded": 1,
  "local_path": "/Users/your-username/downloads/gpunet_0_pyt_ckpt_v21.12.0_amp",
  "size_downloaded": "181.85 MB",
  "status": "Completed",
  "transfer_id": "gpunet_0_pyt_ckpt[version=21.12.0_amp]"
}
```

Renommez le fichier de point de contrôle pour qu'il corresponde au nom attendu dans le code de notre application lors des étapes ultérieures (aucune extraction n'est nécessaire, car il s'agit d'un point de contrôle PyTorch *.pth.tar standard contenant le dictionnaire d'état du modèle) :

```
mv ~/downloads/gpunet_0_pyt_ckpt_v21.12.0_amp/0.65ms.pth.tar gpunet-0.pth
```

Activez le AWS Common Runtime dans la AWS CLI pour optimiser le débit S3 :

```
aws configure set s3.preferred_transfer_client crt
```

Téléchargez les poids du modèle dans votre compartiment S3 :

```
aws s3 cp gpunet-0.pth s3://${S3_BUCKET_NAME}/gpunet-0.pth
```

Le résultat attendu devrait ressembler à ceci :

```
upload: ./gpunet-0.pth to s3://eks-rt-inference-models-us-east-1-1752722786/
gpunet-0.pth
```

Créer le service de modèle

Au cours de cette étape, vous allez configurer une application Web FastAPI pour la classification d'images accélérée par GPU à l'aide du modèle de vision -0. GPUNet L'application télécharge les pondérations des modèles depuis Amazon S3 au moment de l'exécution, extrait l'architecture du modèle depuis le référentiel NVIDIA pour la mise en cache et télécharge les étiquettes de ImageNet classe via HTTP. L'application comprend des transformations de prétraitement d'images et expose deux points de terminaison : un GET racine pour la vérification de l'état et un point de terminaison `/predict` POST qui accepte une URL d'image.

Nous diffusons le modèle à l'aide de FastAPI PyTorch, en chargeant les poids depuis Amazon S3 lors de l'exécution dans une configuration conteneurisée pour un prototypage rapide et un déploiement Kubernetes. Pour d'autres méthodes telles que le traitement par lots optimisé ou les moteurs à haut débit, consultez la section Utilisation des modèles ML.

Pour créer l'application

Créez un répertoire pour vos fichiers d'application, tel que `model-testing`, puis accédez à ce répertoire et ajoutez le code suivant à un nouveau fichier nommé `app.py` :

```
import os
import torch
import json
import requests
from fastapi import FastAPI, HTTPException
```

```
from PIL import Image
from io import BytesIO, StringIO
import torchvision.transforms as transforms
from torch.nn.functional import softmax
import warnings
from contextlib import redirect_stdout, redirect_stderr
import argparse
import boto3
app = FastAPI()

# Suppress specific warnings from the model code (quantization is optional and unused
 here)
warnings.simplefilter("ignore", UserWarning)

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

# Load model code from cache (if present)
# Use backed cache directory
torch.hub.set_dir('/cache/torch/hub')

# Allowlist for secure deserialization (handles potential issues in older checkpoints)
torch.serialization.add_safe_globals([argparse.Namespace])
# Load the model architecture only on container startup (changed to pretrained=False)
# Precision (FP32 for full accuracy, could be 'fp16' for speed on Ampere+ GPUs)
with redirect_stdout(StringIO()), redirect_stderr(StringIO()):
    gpunet = torch.hub.load('NVIDIA/DeepLearningExamples:torchhub', 'nvidia_gpunet',
 pretrained=False, model_type='GPUNet-0', model_math='fp32')

# Download weights from S3 if not present, then load them
model_path = os.getenv('MODEL_PATH', '/cache/torch/hub/checkpoints/gpunet-0.pth')
os.makedirs(os.path.dirname(model_path), exist_ok=True)  # Ensure checkpoints dir
 exists
if not os.path.exists(model_path):
    s3 = boto3.client('s3')
    s3.download_file(os.getenv('S3_BUCKET_NAME'), 'gpunet-0.pth', model_path)
checkpoint = torch.load(model_path, map_location=device, weights_only=True)
gpunet.load_state_dict(checkpoint['state_dict'])
# Move to GPU/CPU
gpunet.to(device)
gpunet.eval()

# Preprocessing
preprocess = transforms.Compose([
    transforms.Resize(256),
```

```
    transforms.CenterCrop(224),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]),
])

# Load ImageNet labels
labels_url = "https://s3.amazonaws.com/deep-learning-models/image-models/
imagenet_class_index.json"
response = requests.get(labels_url)
json_data = json.loads(response.text)
labels = [json_data[str(i)][1].replace('_', ' ') for i in range(1000)]

# Required, FastAPI root
@app.get("/")
async def hello():
    return {"status": "hello"}

# Serve model requests
@app.post("/predict")
async def predict(image_url: str):
    try:
        response = requests.get(image_url)
        response.raise_for_status()
        img = Image.open(BytesIO(response.content)).convert("RGB")
        input_tensor = preprocess(img).unsqueeze(0).to(device)

        with torch.no_grad():
            output = gpunet(input_tensor)

        probs = softmax(output, dim=1)[0]
        top5_idx = probs.topk(5).indices.cpu().numpy()
        top5_probs = probs.topk(5).values.cpu().numpy()

        results = [{ "label": labels[idx], "probability": float(prob) } for idx, prob
 in zip(top5_idx, top5_probs)]

        return {"predictions": results}
    except Exception as e:
        raise HTTPException(status_code=400, detail=str(e))
```

Créer le fichier Dockerfile

Le Dockerfile suivant crée une image de conteneur pour notre application en utilisant le GPUNet modèle du référentiel NVIDIA Deep Learning Examples for Tensor Cores. GitHub

Nous réduisons la taille de l'image du conteneur en utilisant une PyTorch base réservée à l'exécution, en installant uniquement les packages essentiels avec nettoyage du cache, pré-mise en cache du code du modèle, et en évitant de « modifier » les poids dans l'image du conteneur afin de permettre des extractions et des mises à jour plus rapides. Pour en savoir plus, consultez la section Réduire la taille des images de conteneur.

Dans le même répertoire que `app.py`, créez le `Dockerfile` :

```
FROM pytorch/pytorch:2.4.0-cuda12.4-cudnn9-runtime

# Install required system packages required for git cloning
RUN apt-get update && apt-get install -y git && rm -rf /var/lib/apt/lists/*

# Install application dependencies
RUN pip install --no-cache-dir fastapi uvicorn requests pillow boto3 timm==0.5.4

# Pre-cache the GPUNet code from Torch Hub (without weights)
# Clone the repository containing the GPUNet code
RUN mkdir -p /cache/torch/hub && \
    cd /cache/torch/hub && \
    git clone --branch torchhub --depth 1 https://github.com/NVIDIA/
DeepLearningExamples NVIDIA_DeepLearningExamples_torchhub

COPY app.py /app/app.py

WORKDIR /app

CMD ["uvicorn", "app:app", "--host", "0.0.0.0", "--port", "80"]
```

Tester l'application

À partir du même répertoire que votre `app.py` et`Dockerfile`, créez l'image du conteneur pour l'application d'inférence, en ciblant AMD64 l'architecture :

```
docker build --platform linux/amd64 -t gpunet-inference-app .
```

Définissez des variables d'environnement pour vos AWS informations d'identification, et éventuellement un jeton de AWS session. Par exemple :

```
export AWS_REGION="us-east-1"
export AWS_ACCESS_KEY_ID=ABCEXAMPLESCUJFEIELSMUHHAZ
export AWS_SECRET_ACCESS_KEY=123EXAMPLEMZREoQXr8XkiicsOgWDQ5TpUsq0/Z
```

Exécutez le conteneur localement, en injectant des AWS informations d'identification en tant que variables d'environnement pour l'accès à S3. Par exemple :

```
docker run --platform linux/amd64 -p 8080:80 \
  -e S3_BUCKET_NAME=${S3_BUCKET_NAME} \
  -e AWS_ACCESS_KEY_ID=${AWS_ACCESS_KEY_ID} \
  -e AWS_SECRET_ACCESS_KEY=${AWS_SECRET_ACCESS_KEY} \
  -e AWS_DEFAULT_REGION=${AWS_REGION} \
  gpunet-inference-app
```

Le résultat attendu devrait ressembler à ceci :

```
INFO:     Started server process [1]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     Uvicorn running on http://0.0.0.0:80 (Press CTRL+C to quit)
```

Dans une nouvelle fenêtre de terminal, testez le point de terminaison d'inférence en envoyant une requête POST d'exemple avec une URL d'image publique comme paramètre de requête :

```
curl -X POST "http://localhost:8080/predict?image_url=http://images.cocodataset.org/
test-stuff2017/000000024309.jpg"
```

Le résultat attendu devrait être une réponse JSON avec les 5 meilleures prédictions, similaire à ceci (les étiquettes et probabilités réelles peuvent varier légèrement en fonction de l'image et de la précision du modèle) :

```
{"predictions":[{"label":"desk","probability":0.28885871171951294},
{"label":"laptop","probability":0.24679335951805115},
{"label":"notebook","probability":0.08539070934057236},
{"label":"library","probability":0.030645888298749924},
{"label":"monitor","probability":0.02989606373012066}]}
```

Quittez l'application en appuyant sur « Ctrl + C ».

Poussez le conteneur vers Amazon ECR

Au cours de cette étape, nous téléchargeons l'image du conteneur pour le service modèle GPUNet -0 sur Amazon Elastic Container Registry (ECR), afin de la rendre disponible pour le déploiement sur Amazon EKS. Ce processus consiste à créer un nouveau référentiel ECR pour stocker l'image, à s'authentifier auprès d'ECR, puis à baliser et à pousser l'image du conteneur vers notre registre.

Tout d'abord, retournez dans le répertoire où vous avez défini vos variables d'environnement au début de ce guide. Par exemple :

```
cd ..
```

Créer un référentiel dans Amazon ECR :

```
aws ecr create-repository --repository-name gpunet-inference-app --region ${AWS_REGION}
```

Connectez-vous à Amazon ECR :

```
aws ecr get-login-password --region ${AWS_REGION} | docker login --username AWS --
password-stdin ${AWS_ACCOUNT_ID}.dkr.ecr.${AWS_REGION}.amazonaws.com
```

Le résultat attendu devrait ressembler à ceci :

```
Login Succeeded
```

Balisez l'image :

```
docker tag gpunet-inference-app:latest ${AWS_ACCOUNT_ID}.dkr.ecr.
${AWS_REGION}.amazonaws.com/gpunet-inference-app:latest
```

Poussez l'image vers votre référentiel Amazon ECR :

```
docker push ${AWS_ACCOUNT_ID}.dkr.ecr.${AWS_REGION}.amazonaws.com/gpunet-inference-
app:latest
```

Cette dernière étape prend plusieurs minutes.

7. (Facultatif) Exposez le modèle de service

Au cours de cette étape, vous allez exposer votre service de modèle d'inférence en temps réel en externe sur Amazon EKS à l'aide du AWS Load Balancer Controller (LBC). Cela implique la configuration du LBC, le montage des poids de modèle depuis Amazon S3 en tant que volume persistant à l'aide du pilote Mountpoint S3 CSI, le déploiement d'un pod d'application accéléré par GPU, la création d'un service et d'une entrée pour provisionner un Application Load Balancer (ALB) et le test du point de terminaison.

Vérifiez d'abord l'association Pod Identity pour le AWS LBC, en vous assurant que le compte de service est correctement lié au rôle IAM requis :

```
eksctl get podidentityassociation --cluster ${EKS_CLUSTER_NAME} --namespace kube-system
 --service-account-name aws-load-balancer-controller
```

Le résultat attendu devrait ressembler à ceci :

```
ASSOCIATION ARN                                                    NAMESPACE    SERVICE
 ACCOUNT NAME        IAM ROLE ARN    OWNER ARN
arn:aws:eks:us-east-1:143095308808:podidentityassociation/eks-rt-inference-us-
east-1/a-buavluu2wp1jropya    kube-system     aws-load-balancer-controller
 arn:aws:iam::143095308808:role/AmazonEKSLoadBalancerControllerRole
```

Baliser votre groupe de sécurité de cluster

Le AWS Load Balancer Controller ne prend en charge qu'un seul groupe de sécurité avec la clé de balise `karpenter.sh/discovery: "${EKS_CLUSTER_NAME}"` pour la sélection du groupe de sécurité par Karpenter. Lors de la création d'un cluster avec eksctl, le groupe de sécurité par défaut du cluster (qui possède la balise `"kubernetes.io/cluster/<cluster-name>: owned"`) n'est pas automatiquement balisé avec des balises `karpenter.sh/discovery`. Cette balise est essentielle pour que Karpenter puisse détecter et associer ce groupe de sécurité aux nœuds qu'il provisionne. L'attachement de ce groupe de sécurité garantit la compatibilité avec le AWS Load Balancer Controller (LBC), ce qui lui permet de gérer automatiquement les règles de trafic entrant pour les services exposés via Ingress, tels que le service modèle décrit dans ces étapes.

Exportez l'ID VPC de votre cluster :

```
CLUSTER_VPC_ID="$(aws eks describe-cluster --name ${EKS_CLUSTER_NAME} --query
 cluster.resourcesVpcConfig.vpcId --output text)"
```

Exportez le groupe de sécurité par défaut de votre cluster :

```
CLUSTER_SG_ID="$(aws ec2 describe-security-groups --filters Name=vpc-id,Values=
$CLUSTER_VPC_ID Name=tag-key,Values=kubernetes.io/cluster/${EKS_CLUSTER_NAME} --query
 'SecurityGroups[].[GroupId]' --output text)"
```

Ajoutez la balise `karpenter.sh/discovery` au groupe de sécurité du cluster par défaut. Cela permettra à nos EC2 NodeClass sélecteurs de CPU et de GPU de l'utiliser :

```
aws ec2 create-tags --resources ${CLUSTER_SG_ID} --tags Key=karpenter.sh/
discovery,Value=${EKS_CLUSTER_NAME}
```

Vérifiez que le tag a été ajouté :

```
aws ec2 describe-security-groups --group-ids ${CLUSTER_SG_ID} --query
 "SecurityGroups[].Tags"
```

Parmi les résultats, vous devriez voir ce qui suit avec le tag et le nom de votre cluster. Par exemple :

```
{
  "Key": "karpenter.sh/discovery",
  "Value": "eks-rt-inference-us-east-1"
}
```

Configuration du AWS Load Balancer Controller (LBC)

Le AWS LBC est essentiel pour gérer le trafic entrant vers les charges de AI/ML travail sur Amazon EKS, en garantissant l'accès aux points de terminaison d'inférence ou aux pipelines de traitement des données. En s'intégrant aux équilibreurs de charge d' AWS application (ALB) et aux équilibreurs de charge réseau (NLB), le LBC achemine dynamiquement le trafic vers des applications conteneurisées, telles que celles qui exécutent de grands modèles linguistiques, des modèles de vision par ordinateur ou des services d'inférence en temps réel. Comme nous avons déjà créé le compte de service et l'association d'identité de pod lors de la création du cluster, nous définissons le paramètre `serviceAccount.name` pour qu'il corresponde à ce qui est défini dans la configuration de notre cluster (`aws-load-balancer-controller`).

Ajoutez le référentiel de cartes eks-charts Helm AWS appartenant à eks-charts :

```
helm repo add eks https://aws.github.io/eks-charts
```

Actualisez vos référentiels Helm locaux avec les graphiques les plus récents :

```
helm repo update eks
```

Déployez le AWS LBC à l'aide de Helm, en spécifiant le nom du cluster EKS et en faisant référence au compte de service pré-créé :

```
helm install aws-load-balancer-controller eks/aws-load-balancer-controller \
  -n kube-system \
  --set clusterName=${EKS_CLUSTER_NAME} \
  --set serviceAccount.create=false \
  --set serviceAccount.name=aws-load-balancer-controller
```

Le résultat attendu devrait ressembler à ceci :

```
NAME: aws-load-balancer-controller
LAST DEPLOYED: Wed Jul 9 15:03:31 2025
NAMESPACE: kube-system
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
AWS Load Balancer controller installed!
```

Monter le modèle dans un volume persistant

Au cours de cette étape, vous allez monter les pondérations des modèles à partir de votre compartiment Amazon S3 à l'aide d'un PersistentVolume (PV) soutenu par le pilote CSI Mountpoint for Amazon S3. Cela permet aux pods Kubernetes d'accéder aux objets S3 comme s'il s'agissait de fichiers locaux, éliminant ainsi les téléchargements gourmands en ressources vers le stockage éphémère des pods ou les conteneurs d'initialisation, ce qui est idéal pour les modèles volumineux de plusieurs gigaoctets.

Le PV monte l'ensemble de la racine du bucket (aucun chemin spécifié dans `volumeAttributes`), prend en charge l'accès en lecture seule simultané par plusieurs pods et expose les fichiers tels que les poids du modèle (`/models/gpunet-0.pth`) à l'intérieur du conteneur pour l'inférence. Cela garantit que la fonction de secours « téléchargement » de notre application (`app.py`) ne se déclenche pas, car le fichier existe via le montage. En dissociant le modèle de l'image du conteneur, cela permet un accès partagé et des mises à jour indépendantes des versions du modèle sans avoir à reconstruire l'image.

Créez le PersistentVolume (PV)

Créez une ressource PersistentVolume (PV) pour monter le compartiment S3 contenant les poids de votre modèle, en permettant un accès en lecture seule à plusieurs pods sans télécharger de fichiers lors de l'exécution :

```
cat <<EOF | envsubst | kubectl apply -f -
apiVersion: v1
kind: PersistentVolume
metadata:
  name: s3-model-pv
spec:
  capacity:
    storage: 5Gi  # Ignored by the driver; can be any value
  accessModes:
    - ReadOnlyMany  # Read only
  persistentVolumeReclaimPolicy: Retain
  storageClassName: ""  # Required for static provisioning
  claimRef:
    namespace: default  # Adjust if you prefer a different namespace
    name: s3-model-pvc
  mountOptions:
    - allow-other  # Enables multi-user access (useful for non-root pods)
    - region ${AWS_REGION} # Optional, include if your bucket is in a different region
 than the cluster
  csi:
    driver: s3.csi.aws.com
    volumeHandle: gpunet-model-volume  # Must be unique across all PVs
    volumeAttributes:
      bucketName: ${S3_BUCKET_NAME}
EOF
```

Créez le PersistentVolumeClaim (PVC)

Créez un PersistentVolumeClaim (PVC) à lier au PV, en demandant un accès en lecture seule aux données du modèle S3 monté :

```
cat <<EOF | envsubst | kubectl apply -f -
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: s3-model-pvc
spec:
```

```
  accessModes:
    - ReadOnlyMany
  storageClassName: ""  # Required for static provisioning
  resources:
    requests:
      storage: 5Gi  # Ignored, match PV capacity
  volumeName: s3-model-pv  # Bind to the PV created above
EOF
```

Déployer l'application

Déployez l'application d'inférence en tant que déploiement Kubernetes, en montant le volume persistant pris en charge par S3 pour l'accès au modèle, en appliquant les sélecteurs et les tolérances de nœuds GPU, et en définissant les variables d'environnement pour le chemin d'accès au modèle. Ce déploiement définit le chemin d'accès au modèle (variable d'environnement `"/models/gpunet-0.pth"`), de sorte que notre application (dans `app.py`) utilisera ce chemin par défaut. Avec le volume de déploiement monté à `/models` (en lecture seule), le téléchargement du modèle ne se déclenchera pas si le fichier est déjà présent via le PVC.

```
cat <<EOF | envsubst | kubectl apply -f -
apiVersion: apps/v1
kind: Deployment
metadata:
  name: gpunet-inference-app
spec:
  replicas: 1
  selector:
    matchLabels:
      app: gpunet-inference-app
  template:
    metadata:
      labels:
        app: gpunet-inference-app
    spec:
      tolerations:
      - key: "nvidia.com/gpu"
        operator: "Exists"
        effect: "NoSchedule"
      nodeSelector:
        role: gpu-worker
      containers:
      - name: inference
```

```
        image: ${AWS_ACCOUNT_ID}.dkr.ecr.${AWS_REGION}.amazonaws.com/gpunet-inference-
app:latest
        ports:
        - containerPort: 80
        env:
        - name: MODEL_PATH
          value: "/models/gpunet-0.pth"
        resources:
          limits:
            nvidia.com/gpu: 1
          requests:
            nvidia.com/gpu: 1
        volumeMounts:
        - name: model-volume
          mountPath: /models
          readOnly: true
      volumes:
      - name: model-volume
        persistentVolumeClaim:
          claimName: s3-model-pvc
EOF
```

Il faudra quelques minutes à Karpenter pour provisionner un nœud GPU si aucun n'est déjà disponible. Vérifiez que le pod d'inférence est dans l'état « En cours d'exécution » :

```
kubectl get pods -l app=gpunet-inference-app
```

Le résultat attendu devrait ressembler à ceci :

```
NAME                                  READY   STATUS    RESTARTS   AGE
gpunet-inference-app-5d4b6c7f8-abcde        1/1     Running   0          2m
```

Exposez le service avec Ingress et Load Balancer

Créez un service ClusterIP pour exposer le déploiement d'inférence en interne au sein du cluster EKS, en ciblant le port de l'application :

```
cat <<EOF | envsubst | kubectl apply -f -
apiVersion: v1
kind: Service
metadata:
```

```
  name: gpunet-model-service
spec:
  type: ClusterIP
  ports:
  - port: 80
    targetPort: 80
  selector:
    app: gpunet-inference-app
EOF
```

Créez une ressource Ingress pour approvisionner un Application Load Balancer (ALB) connecté à Internet via le LBC, en acheminant le trafic externe vers AWS le service d'inférence :

```
cat <<EOF | envsubst | kubectl apply -f -
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: gpunet-model-ingress
  annotations:
    alb.ingress.kubernetes.io/scheme: internet-facing
    alb.ingress.kubernetes.io/target-type: ip
spec:
  ingressClassName: alb
  rules:
  - http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: gpunet-model-service
            port:
              number: 80
EOF
```

Donnez à Application Load Balancer (ALB) la fin du provisionnement en l'espace de quelques minutes. Surveillez l'état des ressources d'entrée pour confirmer que l'ALB a été provisionné :

```
kubectl get ingress gpunet-model-ingress
```

La sortie attendue devrait ressembler à ceci (avec le champ ADDRESS rempli) :

```
NAME                    CLASS   HOSTS   ADDRESS
 PORTS   AGE
gpunet-model-ingress   alb     *       k8s-default-gpunetmo-183de3f819-516310036.us-
east-1.elb.amazonaws.com   80      6m58s
```

Extrayez et exportez le nom d'hôte ALB à partir de l'état d'entrée pour l'utiliser lors de tests ultérieurs :

```
export ALB_HOSTNAME=$(kubectl get ingress gpunet-model-ingress -o
 jsonpath='{.status.loadBalancer.ingress[0].hostname}')
```

Testez le Model Service

Validez le point de terminaison d'inférence exposé en envoyant une demande POST avec une URL d'image échantillon (par exemple, provenant de du jeu de données COCO), simulant une prédiction en temps réel :

```
curl -X POST "http://${ALB_HOSTNAME}/predict?image_url=http://images.cocodataset.org/
test-stuff2017/000000024309.jpg"
```

Le résultat attendu devrait être une réponse JSON avec les 5 meilleures prédictions, similaire à ceci (les étiquettes et probabilités réelles peuvent varier légèrement en fonction de l'image et de la précision du modèle) :

```
{"predictions":[{"label":"desk","probability":0.2888975441455841},
{"label":"laptop","probability":0.2464350312948227},
{"label":"notebook","probability":0.08554483205080032},
{"label":"library","probability":0.030612602829933167},
{"label":"monitor","probability":0.02989667281508457}]}
```

Vous pouvez éventuellement continuer à tester d'autres images dans une nouvelle requête POST. Par exemple :

```
http://images.cocodataset.org/test-stuff2017/000000024309.jpg
http://images.cocodataset.org/test-stuff2017/000000028117.jpg
http://images.cocodataset.org/test-stuff2017/000000006149.jpg
http://images.cocodataset.org/test-stuff2017/000000004954.jpg
```

Conclusion

Dans ce guide, vous allez configurer un cluster Amazon EKS optimisé pour les charges de travail d'inférence en temps réel accélérées par GPU. Vous avez provisionné un cluster avec des instances G5 EC2, installé le pilote CSI Mountpoint S3, l'agent EKS Pod Identity, l'agent de surveillance des nœuds EKS, l'AMI Bottlerocket, le AWS Load Balancer Controller (LBC) et Karpenter pour gérer le processeur et le GPU. NodePools Vous avez utilisé le plug-in NVIDIA Device pour activer la planification du GPU et vous avez configuré S3 avec PersistentVolume et PersistentVolumeClaim pour l'accès aux modèles. Vous avez validé la configuration en déployant un exemple de pod GPU, en configurant l'accès au modèle NVIDIA GPUNet-0 sur Amazon S3, en activant l'initialisation du pod et en exposant le service d'inférence via Application Load Balancer. Pour exploiter pleinement votre cluster, configurez EKS Node Monitoring Agent avec la fonction de réparation automatique. Veillez à effectuer des tests de référence, notamment des évaluations des performances, de la latence et du débit du GPU afin d'optimiser les temps de réponse. Pour en savoir plus, consultez la section Utilisation des outils de surveillance et d'observabilité pour vos charges AI/ML de travail.

Nettoyage

Pour éviter d'encourir de futurs frais, vous devez supprimer manuellement la CloudFormation pile associée pour supprimer toutes les ressources créées au cours de ce guide, y compris le réseau VPC.

Supprimez la CloudFormation pile en utilisant le `--wait` drapeau avec eksctl :

```
eksctl delete cluster --region ${AWS_REGION} --name ${EKS_CLUSTER_NAME} --wait
```

Une fois l'opération terminée, vous devriez voir s'afficher la réponse suivante :

```
2025-07-29 13:03:55 [#]  all cluster resources were deleted
```

Supprimez le compartiment Amazon S3 créé au cours de ce guide à l'aide de la console Amazon S3.

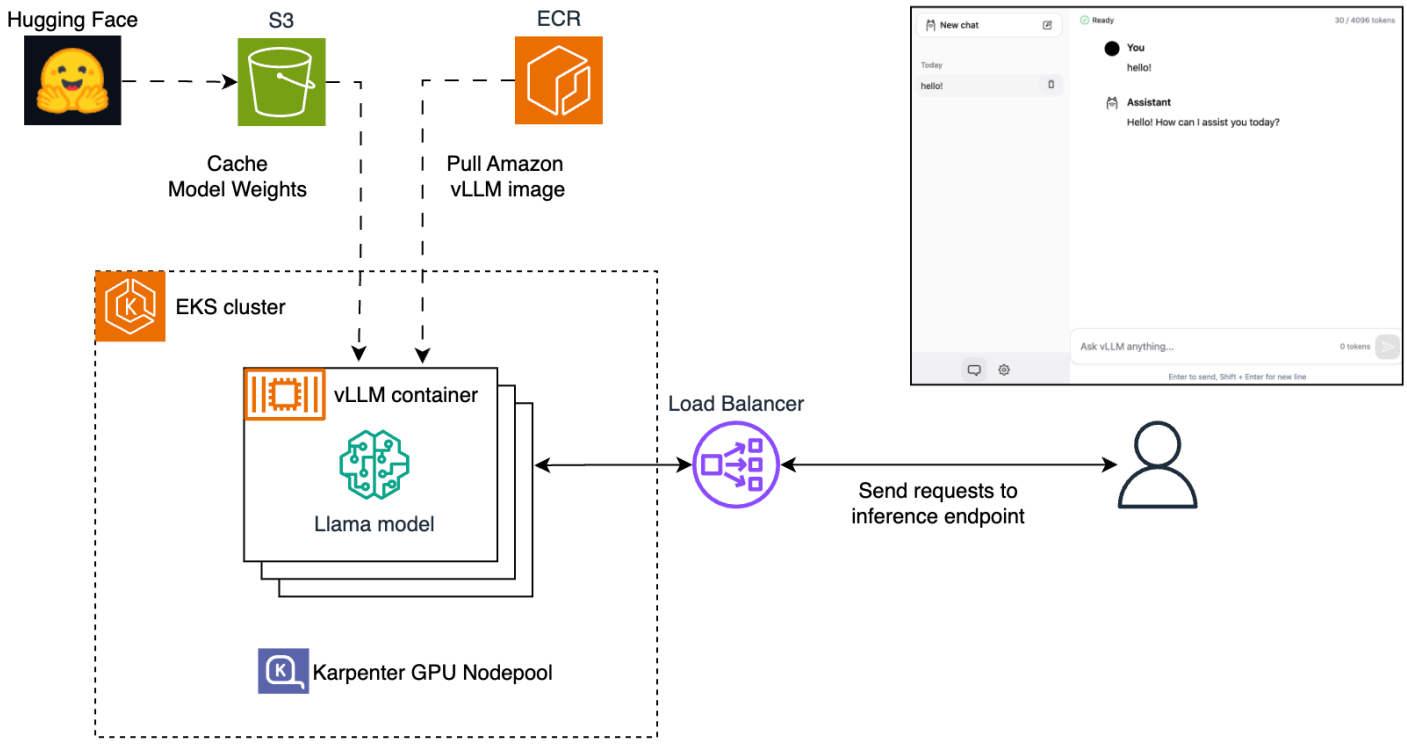
Démarrage rapide : inférence LLM à haut débit avec vLLM sur Amazon EKS

Introduction

Ce guide de démarrage rapide fournit une procédure pas à pas pour déployer de grands modèles linguistiques (LLMs) sur Amazon EKS à l'aide de vLLM et GPUs pour les applications d'inférence en temps réel basées sur du texte.

La solution s'appuie sur Amazon EKS pour l'orchestration de conteneurs et sur vLLM pour un service de modèles efficace, ce qui vous permet de créer des applications d'IA évolutives avec accélération GPU et service d'inférence à haut débit. Le modèle Llama 3.1 8B Instruct est utilisé à des fins de démonstration, mais vous pouvez déployer tout autre LLM pris en charge par vLLM (consultez la documentation vLLM pour obtenir la liste des modèles pris en charge). Pour tester l'inférence LLM, nous utilisons un exemple d'application de chatbot basé sur le projet. nextjs-vllm-ui Enfin, nous utilisons GuideLM pour comparer et ajuster les paramètres de configuration vLLM afin d'optimiser les performances d'inférence.

Architecture vLLM sur EKS



Lorsque vous aurez terminé cette procédure, vous disposerez d'un point de terminaison d'inférence vLLM optimisé pour le débit et une faible latence, et vous pourrez interagir avec un modèle de lama via une application frontale de chat, illustrant ainsi un cas d'utilisation typique des assistants chatbot et d'autres applications basées sur le LLM.

Pour obtenir des conseils supplémentaires et des ressources de déploiement avancées, consultez notre guide des meilleures pratiques EKS pour les AI/ML charges de travail et l'IA prête à la production sur les graphiques d'inférence EKS.

Avant de commencer

Avant de commencer, assurez-vous d'avoir :

- Un cluster Amazon EKS avec les principaux composants suivants : des pools de nœuds Karpenter dotés de la famille d' EC2 instances G5 ou G6, le plug-in de périphérique NVIDIA installé sur vos nœuds de travail compatibles GPU et le pilote S3 Mountpoint CSI installé. Pour créer cette configuration de base, suivez les étapes décrites dansthe section called "Créer un cluster", jusqu'à l'étape #4.
- Un compte Hugging Face. Pour vous inscrire, rendez-vous sur https://huggingface.co/login.

Configurer Model Storage avec Amazon S3

Stockez efficacement des fichiers LLM volumineux dans Amazon S3 afin de séparer le stockage des ressources de calcul. Cette approche rationalise les mises à jour des modèles, réduit les coûts et simplifie la gestion des configurations de production. S3 gère les fichiers volumineux de manière fiable, tandis que l'intégration à Kubernetes via le pilote Mountpoint CSI permet aux pods d'accéder à des modèles tels que le stockage local, sans avoir à effectuer de longs téléchargements au démarrage. Suivez ces étapes pour créer un compartiment S3, télécharger un LLM et le monter en tant que volume dans votre conteneur de service d'inférence.

D'autres solutions de stockage sont également disponibles sur EKS pour la mise en cache des modèles, telles que EFS et FSx Lustre. Pour plus d'informations, consultez les meilleures pratiques d'EKS.

Définir les variables d'environnement

Créez un nom unique pour un nouveau compartiment Amazon S3 que nous créerons plus loin dans ce guide. Une fois créé, utilisez le même nom de compartiment pour toutes les étapes. Par exemple :

```
MY_BUCKET_NAME=model-store-$(date +%s)
```

Définissez les variables d'environnement et stockez-les dans un fichier :

```
cat << EOF > .env-quickstart-vllm
export BUCKET_NAME=${MY_BUCKET_NAME}
export AWS_REGION=us-east-1
export AWS_ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
EOF
```

Chargez les variables d'environnement dans votre environnement shell. Si vous fermez l'environnement shell actuel et que vous en ouvrez un nouveau, assurez-vous de réapprovisionner les variables d'environnement à l'aide de cette même commande :

```
source .env-quickstart-vllm
```

Création d'un compartiment S3 pour stocker les fichiers de modèles

Créez un compartiment S3 pour stocker les fichiers de modèles :

```
aws s3 mb s3://${BUCKET_NAME} --region ${AWS_REGION}
```

Télécharger le modèle de Hugging Face

Hugging Face est l'un des principaux centres de modélisation permettant d'accéder aux modèles LLM. Pour télécharger le modèle Llama, vous devez accepter la licence du modèle et configurer l'authentification par jeton :

1. Acceptez la licence du modèle Llama 3.1 8B Instruct sur https://huggingface. co/meta-llama/Llama-3.1-8B-Instruire.
2. Générez un jeton d'accès (accédez à votre profil > Paramètres > Jetons d'accès, puis créez un nouveau jeton en utilisant le type de jeton de lecture).

Définissez une variable d'environnement avec votre jeton Hugging Face :

```
export HF_TOKEN=your_token_here
```

Installez le package pip3 s'il n'est pas déjà installé dans votre environnement. Exemple de commande dans Amazon Linux 2023 :

```
sudo dnf install -y python3-pip
```

Installez la CLI Hugging Face :

```
pip install huggingface-hub
```

Téléchargez le modèle Llama-3.1-8B-Instruct depuis Hugging Face (~15 Go) en `--exclude` utilisant le drapeau pour ignorer le format existant et ne télécharger que les fichiers au format Safetensors

optimisés, ce qui réduit PyTorch la taille du téléchargement tout en garantissant une compatibilité totale avec les moteurs d'inférence les plus courants :

```
huggingface-cli download meta-llama/Meta-Llama-3.1-8B-Instruct \
  --exclude "original/*" \
  --local-dir ./llama-3.1-8b-instruct \
  --token $HF_TOKEN
```

Vérifiez les fichiers téléchargés :

```
$ ls llama-3.1-8b-instruct
```

Le résultat attendu devrait ressembler à ceci :

```
LICENSE        config.json                      model-00002-of-00004.safetensors
 model.safetensors.index.json  tokenizer_config.json
README.md      generation_config.json           model-00003-of-00004.safetensors
 special_tokens_map.json
USE_POLICY.md  model-00001-of-00004.safetensors  model-00004-of-00004.safetensors
 tokenizer.json
```

Téléchargez des fichiers de modèles

Activez AWS Common Runtime (CRT) pour améliorer les performances de transfert S3. Le client de transfert basé sur CRT fournit un débit et une fiabilité améliorés pour les opérations de fichiers volumineux :

```
aws configure set s3.preferred_transfer_client crt
```

Téléchargez le modèle :

```
aws s3 cp ./llama-3.1-8b-instruct s3://$BUCKET_NAME/llama-3.1-8b-instruct \
  --recursive
```

Le résultat attendu devrait ressembler à ceci :

```
...
upload: llama-3.1-8b-instruct/tokenizer.json to s3://model-store-1753EXAMPLE/
llama-3.1-8b-instruct/tokenizer.json
upload: llama-3.1-8b-instruct/model-00004-of-00004.safetensors to s3://model-
store-1753890326/llama-3.1-8b-instruct/model-00004-of-00004.safetensors
```

```
upload: llama-3.1-8b-instruct/model-00002-of-00004.safetensors to s3://model-
store-1753890326/llama-3.1-8b-instruct/model-00002-of-00004.safetensors
upload: llama-3.1-8b-instruct/model-00003-of-00004.safetensors to s3://model-
store-1753890326/llama-3.1-8b-instruct/model-00003-of-00004.safetensors
upload: llama-3.1-8b-instruct/model-00001-of-00004.safetensors to s3://model-
store-1753890326/llama-3.1-8b-instruct/model-00001-of-00004.safetensors
```

Configurer les autorisations CSI de S3 Mountpoint

Le pilote S3 Mountpoint CSI permet une intégration native entre Kubernetes et S3, permettant aux pods d'accéder directement aux fichiers du modèle comme s'il s'agissait d'un stockage local, éliminant ainsi le besoin de copies locales lors du démarrage du conteneur.

Créez une politique IAM pour autoriser le point de montage S3 à lire depuis votre compartiment S3 :

```
aws iam create-policy \
  --policy-name S3BucketAccess-${BUCKET_NAME} \
  --policy-document "{\"Version\": \"2012-10-17\", \"Statement\": [{\"Effect\":
 \"Allow\", \"Action\": [\"s3:GetObject\", \"s3:GetObjectVersion\", \"s3:ListBucket
\", \"s3:GetBucketLocation\"], \"Resource\": [\"arn:aws:s3:::${BUCKET_NAME}\",
 \"arn:aws:s3:::${BUCKET_NAME}/*\"]}]}"
```

Trouvez le nom du rôle IAM utilisé par le pilote S3 Mountpoint CSI en vérifiant les annotations du compte de service S3 CSI Driver :

```
ROLE_NAME=$(kubectl get serviceaccount s3-csi-driver-sa -n kube-system -o
 jsonpath='{.metadata.annotations.eks\.amazonaws\.com/role-arn}' | cut -d'/' -f2)
```

Associez votre politique IAM au rôle S3 Mountpoint CSI :

```
aws iam attach-role-policy \
 --role-name ${ROLE_NAME} \
 --policy-arn arn:aws:iam::${AWS_ACCOUNT_ID}:policy/S3BucketAccess-${BUCKET_NAME}
```

Si S3 Mountpoint CSI n'est pas installé dans le cluster, suivez les étapes de déploiement décrites dans. the section called “Créer un cluster”

Monter le bucket S3 en tant que volume Kubernetes

Créez un volume persistant (PV) et une réclamation de volume persistant (PVC) pour fournir un accès en lecture seule au compartiment S3 via plusieurs modules d'inférence. Le mode

ReadOnlyMany d'accès garantit un accès simultané aux fichiers du modèle, tandis que le pilote CSI gère le montage du compartiment S3 :

```
cat <<EOF | envsubst | kubectl apply -f -
apiVersion: v1
kind: PersistentVolume
metadata:
  name: model-store
spec:
  storageClassName: ""
  capacity:
    storage: 100Gi
  accessModes:
    - ReadOnlyMany
  persistentVolumeReclaimPolicy: Retain
  mountOptions:
    - region ${AWS_REGION}
  csi:
    driver: s3.csi.aws.com
    volumeHandle: model-store
    volumeAttributes:
      bucketName: ${BUCKET_NAME}
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: model-store
spec:
  storageClassName: ""
  volumeName: model-store
  accessModes:
    - ReadOnlyMany
  resources:
    requests:
      storage: 100Gi
EOF
```

Configuration de l'infrastructure GPU

Nœuds de cluster

Nous utilisons le cluster EKS créé dansthe section called “Créer un cluster”. Ce cluster inclut des pools de nœuds Karpenter qui peuvent fournir aux nœuds compatibles GPU un stockage de nœuds

suffisant pour télécharger l'image du conteneur vLLM. Si vous utilisez votre cluster EKS personnalisé, assurez-vous qu'il peut lancer des nœuds compatibles avec le GPU.

Sélection d'instance

Pour sélectionner correctement les instances pour l'inférence LLM, il faut s'assurer que la mémoire GPU disponible est suffisante pour charger les pondérations du modèle. Le poids du modèle pour Llama 3.1 8B Instruct est d'environ 16 Go (taille des fichiers de modèle .safetensor). Nous devons donc fournir au moins cette quantité de mémoire au processus vllm pour charger le modèle.

EC2 Les instances Amazon G5 avec A10G GPUs et les EC2 instances G6 avec L4 fournissent GPUs toutes deux 24 Go de VRAM par GPU, ce qui est suffisant pour charger des poids d'instructions Llama 3.1 8B. Si vous déployez un modèle avec des poids plus importants, pensez à utiliser un type d'instance multi-GPU ou une configuration multi-nœuds.

Pilotes de périphériques NVIDIA

Les pilotes NVIDIA fournissent l'environnement d'exécution nécessaire aux conteneurs pour accéder efficacement aux ressources du GPU. Il permet l'allocation et la gestion des ressources GPU au sein de Kubernetes, en les rendant GPUs disponibles sous forme de ressources planifiables.

Notre cluster utilise EKS Bottlerocket AMIs, qui inclut tous les pilotes de périphériques et plug-ins NVIDIA nécessaires sur tous les nœuds compatibles avec le GPU, garantissant ainsi un accès immédiat au GPU pour les charges de travail conteneurisées sans configuration supplémentaire. Si vous utilisez d'autres types de nœuds EKS, vous devez vous assurer que tous les pilotes et plug-ins nécessaires sont installés.

Tester l'infrastructure GPU

Testez les capacités GPU de votre cluster en exécutant les étapes ci-dessous pour vous assurer que les pods peuvent accéder aux ressources GPU NVIDIA et planifier correctement sur les nœuds compatibles avec le GPU.

Déployez un module de test Nvidia SMI :

```
cat <<EOF | envsubst | kubectl apply -f -
apiVersion: v1
kind: Pod
metadata:
  name: gpu-nvidia-smi-test
spec:
```

```
  restartPolicy: OnFailure
  tolerations:
  - key: "nvidia.com/gpu"
    operator: "Exists"
    effect: "NoSchedule"
  nodeSelector:
    role: gpu-worker  # Matches GPU NodePool's label
  containers:
  - name: cuda-container
    image: nvidia/cuda:12.9.1-base-ubuntu20.04
    command: ["nvidia-smi"]
    resources:
      requests:
        memory: "24Gi"
      limits:
        nvidia.com/gpu: 1
EOF
```

Consultez les journaux des modules pour vérifier que les détails du GPU sont répertoriés, comme dans le résultat ci-dessous (il ne s'agit pas nécessairement du même modèle de GPU) :

```
$ kubectl wait --for=jsonpath='{.status.phase}'=Succeeded pod/gpu-nvidia-smi-test
$ kubectl logs gpu-nvidia-smi-test
```

```
Wed Jul 30 15:39:58 2025
+-----------------------------------------------------------------------------------------
+
| NVIDIA-SMI 570.172.08             Driver Version: 570.172.08     CUDA Version: 12.9
   |
|-----------------------------------------+------------------------
+----------------------+
| GPU  Name                 Persistence-M | Bus-Id          Disp.A | Volatile Uncorr.
 ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |           Memory-Usage | GPU-Util  Compute
 M. |
|                                         |                        |               MIG
 M. |
|=========================================+========================
+======================|
|   0  NVIDIA A10G                    On  |   00000000:00:1E.0 Off |
 0 |
|  0%   30C    P8              9W /  300W |       0MiB /  23028MiB |      0%
 Default |
```

```
|                                         |                        |
 N/A |
+-----------------------------------------+------------------------
+----------------------+

+-----------------------------------------------------------------------------------------
+
| Processes:
   |
|  GPU   GI   CI              PID   Type   Process name                        GPU
 Memory |
|        ID   ID                                                               Usage
   |
|
=============================================================================================|
|  No running processes found
   |
+-----------------------------------------------------------------------------------------
+
```

Ce résultat montre que les pods peuvent accéder avec succès aux ressources du GPU.

IMPORTANT : Ce pod utilise une configuration NodeSelector qui s'aligne sur les pools de nœuds Karpenter. the section called “Créer un cluster” Si vous utilisez différents pools de nœuds, assurez-vous que le pod correspond à NodeSelector et Tolerations en conséquence.

Déployer un conteneur d'inférence

La pile de service détermine à la fois les performances et les capacités d'évolutivité de votre infrastructure d'inférence. vLLM est devenue une solution de pointe pour les déploiements de production. L'architecture de vLLM permet un traitement par lots continu pour le traitement dynamique des demandes, des optimisations du noyau pour une inférence plus rapide et une gestion efficace de la mémoire du GPU grâce à. PagedAttention Ces fonctionnalités, associées à une API REST prête pour la production et à la prise en charge des formats de modèles courants, en font un choix optimal pour les déploiements d'inférence à hautes performances.

Sélectionnez l'image du conteneur AWS Deep Learning

AWS Les Deep Learning Containers (DLCs) fournissent des environnements préoptimisés avec des mises à jour de sécurité, une compatibilité AWS d'infrastructure et des configurations de pilotes optimisées. Cela réduit la complexité du déploiement et les frais de maintenance tout en garantissant la disponibilité de la production.

Pour ce déploiement, nous utiliserons le AWS DLC pour vLLM 0.9, qui inclut des bibliothèques Nvidia et des configurations de performances GPU optimisées spécialement conçues pour l'inférence du modèle de transformateur sur les instances de GPU. AWS

```
image: 763104351884.dkr.ecr.us-east-1.amazonaws.com/vllm:0.9-gpu-py312-ec2
```

Appliquer les manifestes vLLM Kubernetes

Il existe plusieurs manières de déployer vLLM dans EKS. Ce guide explique le déploiement de vLLM à l'aide d'un déploiement Kubernetes, qui est un moyen simple et natif de Kubernetes pour démarrer. Pour les options de déploiement avancées, consultez la documentation vLLM et les plans AI on EKS.

Définissez les paramètres de déploiement via les manifestes Kubernetes pour contrôler l'allocation des ressources, le placement des nœuds, les sondes de santé, l'exposition du service, etc. Configurez votre déploiement pour exécuter un pod compatible GPU à l'aide de l'image AWS Deep Learning Container pour vLLM. Définissez des paramètres optimisés pour l'inférence LLM et exposez le point de terminaison compatible avec vLLM OpenAPI via le service Load Balancer : AWS

```
cat <<EOF | envsubst | kubectl apply -f -
apiVersion: apps/v1
kind: Deployment
metadata:
  name: vllm-inference-app
spec:
  replicas: 1
  selector:
    matchLabels:
      app: vllm-inference-app
  template:
    metadata:
      labels:
        app: vllm-inference-app
    spec:
      tolerations:
      - key: "nvidia.com/gpu"
        operator: "Exists"
        effect: "NoSchedule"
      nodeSelector:
        role: gpu-worker
      containers:
      - name: vllm-inference
        image: 763104351884.dkr.ecr.us-east-1.amazonaws.com/vllm:0.9-gpu-py312-ec2
```

```
        ports:
        - containerPort: 8000
        env:
        - name: MODEL_PATH
          value: "/mnt/models/llama-3.1-8b-instruct"
        args:
        - "--model=/mnt/models/llama-3.1-8b-instruct"
        - "--host=0.0.0.0"
        - "--port=8000"
        - "--tensor-parallel-size=1"
        - "--gpu-memory-utilization=0.9"
        - "--max-model-len=8192"
        - "--max-num-seqs=1"
        readinessProbe:
          httpGet:
            path: /health
            port: 8000
          initialDelaySeconds: 30
          periodSeconds: 5
          timeoutSeconds: 10
        resources:
          limits:
            nvidia.com/gpu: 1
          requests:
            memory: "24Gi"
            cpu: "4"
            ephemeral-storage: "25Gi"  # Ensure enough node storage for vLLM container
 image
        volumeMounts:
        - name: models
          mountPath: /mnt/models
          readOnly: true
      volumes:
      - name: models
        persistentVolumeClaim:
          claimName: model-store
---
apiVersion: v1
kind: Service
metadata:
  name: vllm-inference-svc
  annotations:
    service.beta.kubernetes.io/aws-load-balancer-type: nlb
    service.beta.kubernetes.io/aws-load-balancer-scheme: internet-facing
```

```
spec:
  type: LoadBalancer
  ports:
    - port: 80
      targetPort: 8000
      protocol: TCP
  selector:
    app: vllm-inference-app
EOF
```

Vérifiez que le pod vLLM est en `Ready` `1/1` bon état :

```
kubectl get pod -l app=vllm-inference-app -w
```

Sortie attendue :

```
NAME                                 READY   UP-TO-DATE   AVAILABLE   AGE
vllm-inference-app-65df5fddc8-5kmjm   1/1     1            1           5m
```

L'extraction de l'image du conteneur peut prendre plusieurs minutes et vLLM charge les fichiers du modèle dans la mémoire du GPU. Procédez uniquement lorsque le module est prêt et disponible.

Exposez le service

Exposez le point de terminaison d'inférence localement via la redirection de port Kubernetes pour le développement et les tests locaux. Laissez cette commande s'exécuter dans une fenêtre de terminal séparée :

```
export POD_NAME=$(kubectl get pod -l app=vllm-inference-app -o
 jsonpath='{.items[0].metadata.name}')
kubectl port-forward pod/$POD_NAME 8000:8000
```

Le AWS Load Balancer Controller crée automatiquement un Network Load Balancer qui expose le point de terminaison du service vLLM à l'extérieur. Récupérez le point de terminaison NLB en exécutant :

```
NLB=$(kubectl get service vllm-inference-svc -o
 jsonpath='{.status.loadBalancer.ingress[0].hostname}')
```

Vous devez installer le AWS Load Balancer Controller ? Suivez les étapes de déploiement décrites dansthe section called “ AWS Contrôleur Load Balancer”.

Exécution de l'inférence

Valider le module d'inférence

Validez la fonctionnalité du conteneur d'inférence localement via le port transféré. Envoyez une demande de connexion et assurez-vous que la réponse inclut le code HTTP 200 :

```
$ curl -IX GET "http://localhost:8000/v1/models"
```

```
HTTP/1.1 200 OK
date: Mon, 13 Oct 2025 23:24:57 GMT
server: uvicorn
content-length: 516
content-type: application/json
```

Testez les capacités d'inférence et validez la connectivité externe en envoyant une demande d'achèvement au LLM via le point de terminaison NLB :

```
curl -X POST "http://$NLB:80/v1/completions" \
  -H "Content-Type: application/json" \
  -d '{
    "model": "/mnt/models/llama-3.1-8b-instruct",
    "prompt": "Explain artificial intelligence:",
    "max_tokens": 512,
    "temperature": 0.7
  }'
```

Ce point de terminaison suit le format de l'API OpenAI, ce qui le rend compatible avec les applications existantes tout en fournissant des paramètres de génération configurables tels que la longueur de réponse et la température pour contrôler la diversité des sorties.

Exécuter l'application chatbot

À titre de démonstration, ce guide exécute un exemple d'application de chatbot utilisant un projet nextjs-vllm-uipour présenter les interactions des utilisateurs avec le modèle.

Exécutez une interface utilisateur de chatbot en tant que conteneur Docker qui mappe le port 3000 à localhost et se connecte au point de terminaison vLLM NLB :

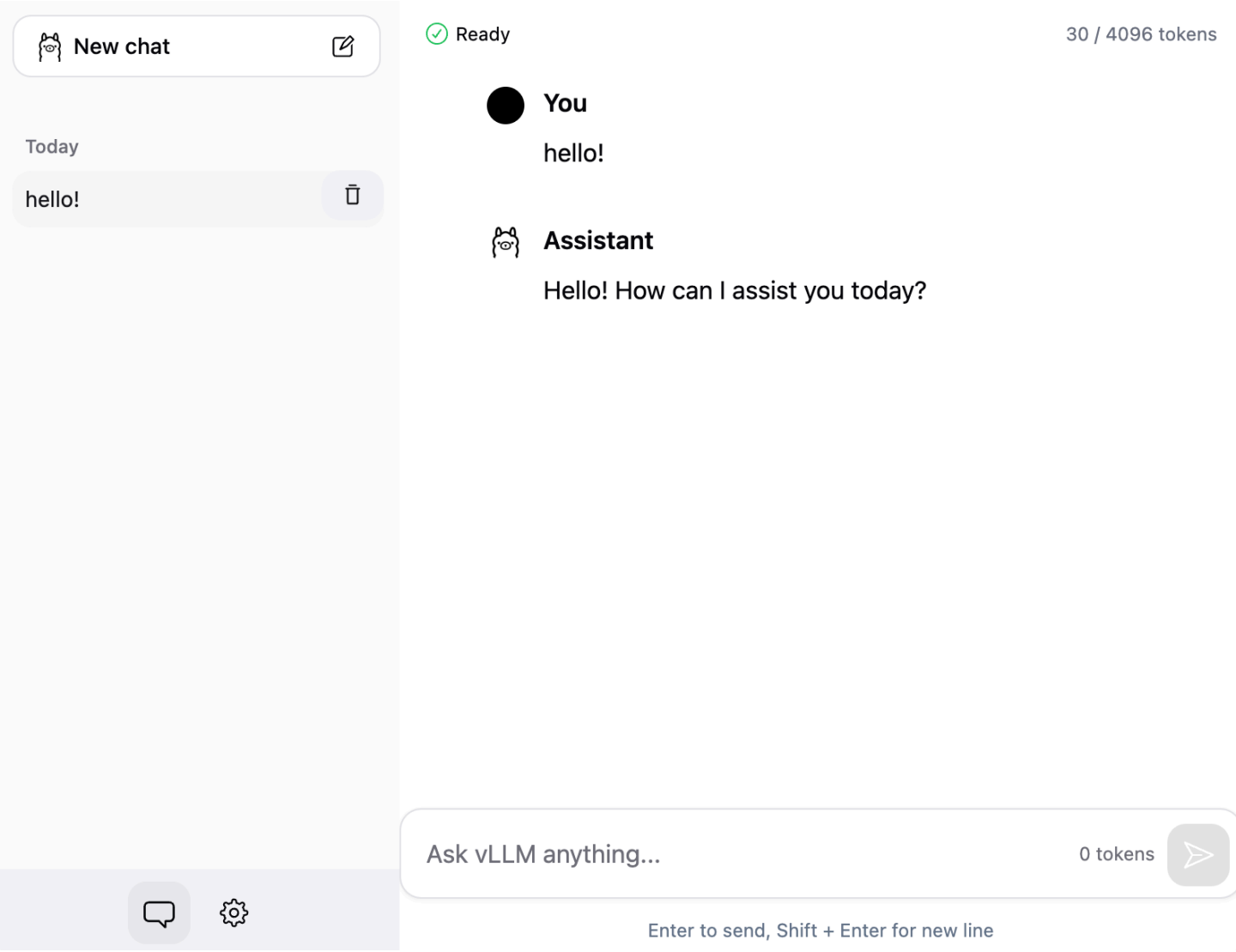
```
docker run --rm \
```

```
  -p 3000:3000 \
  -e VLLM_URL="http://${NLB}:80" \
  --name nextjs-vllm-ui-demo \
  ghcr.io/yoziru/nextjs-vllm-ui:latest
```

Ouvrez votre navigateur Web et accédez à : http://localhost:3000/

Vous devriez voir l'interface de chat où vous pouvez interagir avec le modèle Llama.

Interface utilisateur de chat



Optimisez les performances d'inférence

Les moteurs d'inférence spécialisés tels que vLLM fournissent des fonctionnalités avancées qui améliorent considérablement les performances d'inférence, notamment le traitement par lots continu,

une mise en cache KV efficace et des mécanismes d'attention mémoire optimisés. Vous pouvez ajuster les paramètres de configuration vLLM pour améliorer les performances d'inférence tout en répondant aux exigences spécifiques de vos cas d'utilisation et à vos modèles de charge de travail. Une configuration appropriée est essentielle pour atteindre la saturation du GPU, afin de tirer le meilleur parti des ressources GPU coûteuses tout en garantissant un débit élevé, une faible latence et des opérations rentables. Les optimisations suivantes vous aideront à optimiser les performances de votre déploiement vLLM sur EKS.

Configurations vLLM de référence

Pour ajuster les paramètres de configuration vLLM en fonction de votre cas d'utilisation, comparez différents paramètres à l'aide d'un outil d'analyse d'inférence complet tel que GuidelLM. Cela collectera des indicateurs clés tels que le débit des demandes par seconde (RPS), la end-to-end latence (E2E), le délai jusqu'au premier jeton (TTFT) et la latence finale (TPOT) afin de comparer différentes configurations.

Configuration vLLM de base

Voici la configuration de base qui a été utilisée pour exécuter vLLM :

Paramètre vLLM	Description
taille_parallèle du tenseur : 1	Répartissez le modèle sur 1 GPU
utilisation de la mémoire graphique : 0,90	Réservez 10 % de mémoire GPU pour la surcharge du système
longueur_séquence maximale : 8192	Longueur de séquence totale maximale (entrée+sortie)
max_num_seqs : 1	Nombre maximal de demandes simultanées par GPU (traitement par lots)

Exécutez GuideLLM avec cette configuration de base pour établir une référence de performance. Pour ce test, GuidelLM est configuré pour générer une demande par seconde, avec 256 requêtes à jetons et des réponses à 128 jetons.

```
guidellm benchmark \
--target "http://${NLB}:80" \
```

```
--processor meta-llama/Llama-3.1-8B-Instruct \
--rate-type constant \
--rate 1 \
--max-seconds 30 \
--data "prompt_tokens=256,output_tokens=128"
```

Sortie attendue :

Résultats de référence

```
Benchmarks Info:
=================================================================================================================================================
Metadata                                  |||| Requests Made  ||| Prompt Tok/Req ||| Output Tok/Req ||| Prompt Tok Total||| Output Tok Total||
    Benchmark| Start Time| End Time| Duration (s)|  Comp|  Inc|  Err|  Comp|   Inc| Err|  Comp|   Inc| Err|  Comp|   Inc|  Err|  Comp|   Inc|  Err
-------------|----------|---------|-------------|------|-----|-----|------|------|----|------|------|----|------|------|-----|------|------|-----
constant@1.00|   03:09:34| 03:10:04|         30.0|     6|   24|    0| 256.0| 256.0| 0.0| 128.0| 126.8| 0.0|  1536|  6144|    0|   768|  3043|    0
=================================================================================================================================================


Benchmarks Stats:
=================================================================================================================================================
Metadata     | Request Stats         || Out Tok/sec| Tot Tok/sec| Req Latency (sec) ||| TTFT (ms)           ||| ITL (ms)        ||| TPOT (ms)       ||
    Benchmark| Per Second| Concurrency|        mean|        mean|  mean| median|   p99|   mean| median|     p99| mean| median|  p99| mean| median|  p99
-------------|----------|------------|------------|------------|------|-------|------|-------|-------|--------|-----|-------|-----|-----|-------|-----
constant@1.00|      0.23|        2.94|        28.9|        86.8| 12.99|  11.27| 21.54| 8637.2| 6923.1| 17185.8| 34.2|   34.2| 34.3| 34.0|   34.0| 34.0
=================================================================================================================================================
```

Configuration vLLM optimisée

Ajustez les paramètres vLLM pour mieux utiliser les ressources du GPU et la parallélisation :

Paramètre vLLM	Description
taille_parallèle du tenseur : 1	Limitez votre utilisation à 1 GPU. La parallélisation des tenseurs doit correspondre au nombre de GPUs à utiliser par VLLm
utilisation de la mémoire graphique : 0,92	Réduisez la surcharge de mémoire du GPU si possible, tout en garantissant que vLLM continue de fonctionner sans erreur
longueur_séquence maximale : 4096	Ajustez la séquence maximale selon les exigences de votre cas d'utilisation ; une séquence maximale inférieure libère des ressources qui peuvent être utilisées pour augmenter la paralléli sation
max_num_seqs : 8	L'augmentation de max seq augmente le débit mais augmente également la latence. Augmentez cette valeur pour optimiser le débit tout en garantissant que la latence reste conforme aux exigences de votre cas d'utilisation

Appliquez ces modifications au déploiement en cours à l'aide de la commande kubectl patch :

```
kubectl patch deployment vllm-inference-app --type='json' -p='[
  {"op": "replace", "path": "/spec/template/spec/containers/0/args/4", "value": "--gpu-
memory-utilization=0.92"},
  {"op": "replace", "path": "/spec/template/spec/containers/0/args/5", "value": "--max-
model-len=4096"},
  {"op": "replace", "path": "/spec/template/spec/containers/0/args/6", "value": "--max-
num-seqs=8"}
]'
```

Vérifiez que le pod vLLM est en `Ready 1/1` bon état :

```
kubectl get pod -l app=vllm-inference-app -w
```

Sortie attendue :

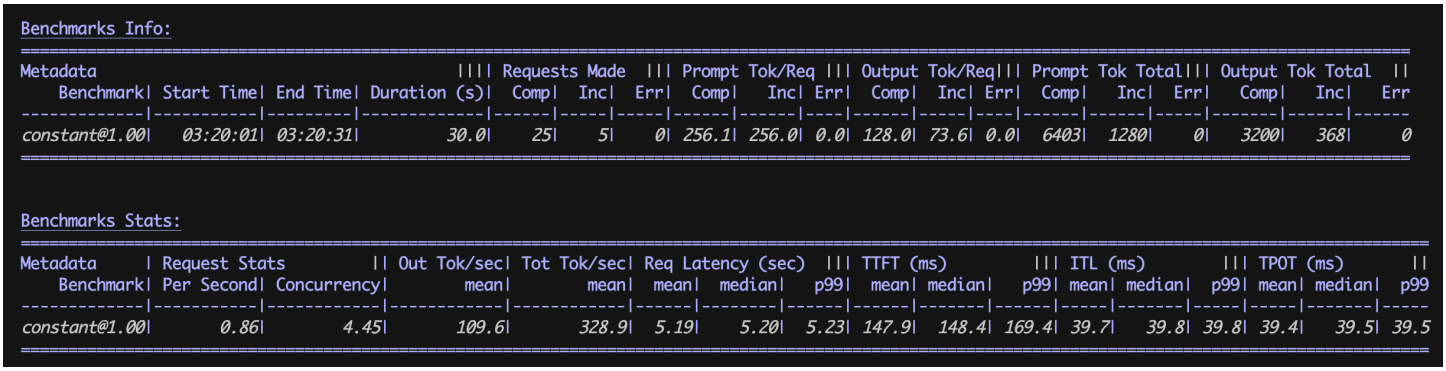
```
NAME                                  READY   UP-TO-DATE   AVAILABLE   AGE
vllm-inference-app-65df5fddc8-5kmjm   1/1     1            1           5m
```

Exécutez ensuite à nouveau GuidelLM en utilisant les mêmes valeurs de référence que précédemment :

```
guidellm benchmark \
--target "http://${NLB}:80" \
--processor meta-llama/Llama-3.1-8B-Instruct \
--rate-type constant \
--rate 1 \
--max-seconds 30 \
--data "prompt_tokens=256,output_tokens=128"
```

Sortie attendue :

Résultats de référence optimisés



```
Benchmarks Info:
═══════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════
Metadata                                  |||| Requests Made  ||| Prompt Tok/Req ||| Output Tok/Req||| Prompt Tok Total||| Output Tok Total  ||
    Benchmark| Start Time| End Time| Duration (s)|  Comp|  Inc|  Err|  Comp|   Inc| Err|  Comp|  Inc| Err|  Comp|   Inc|  Err|   Comp|   Inc|   Err
-------------|-----------|---------|-------------|------|-----|-----|------|------|----|------|-----|----|------|------|-----|-------|------|------
constant@1.00|   03:20:01| 03:20:31|         30.0|    25|    5|    0| 256.1| 256.0| 0.0| 128.0| 73.6| 0.0|  6403|  1280|    0|   3200|   368|     0
═══════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════


Benchmarks Stats:
═══════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════
Metadata     | Request Stats         || Out Tok/sec| Tot Tok/sec| Req Latency (sec)  ||| TTFT (ms)         ||| ITL (ms)        ||| TPOT (ms)       ||
    Benchmark| Per Second| Concurrency|       mean|        mean|  mean|  median|   p99|  mean| median|   p99| mean| median|  p99| mean| median|  p99
-------------|-----------|------------|-----------|------------|------|--------|------|------|-------|------|-----|-------|-----|-----|-------|-----
constant@1.00|       0.86|        4.45|      109.6|       328.9|  5.19|    5.20|  5.23| 147.9|  148.4| 169.4| 39.7|   39.8| 39.8| 39.4|   39.5| 39.5
═══════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════════
```

Résultats de l'analyse comparative

Calculez les résultats de l'analyse comparative dans un tableau pour la configuration de base et la configuration vLLM optimisée :

Valeurs moyennes	Configuration de base	Configuration optimisée
RPS	0,23 requet/sec	0,86 requêt/sec
E2E	12,99 s	5,19 s
TTFT	8637,2 ms	147,9 ms
TPOT	34,0 ms	39,5 ms

Les configurations vLLM optimisées ont considérablement amélioré le débit d'inférence (RPS) et réduit la latence (E2E, TTFT) avec une augmentation mineure de la latence de queue (TPOT) en millisecondes seulement. Ces résultats montrent comment vLLM améliore considérablement les performances d'inférence, permettant à chaque conteneur de traiter plus de demandes en moins de temps pour un fonctionnement rentable.

Configuration du cluster Amazon EKS pour les charges AI/ML de travail

Cette section est conçue pour vous aider à configurer des clusters Amazon EKS optimisés pour les AI/ML charges de travail. Vous y trouverez des conseils sur l'exécution de conteneurs accélérés par GPU optimisés pour Linux et Windows AMIs, sur la configuration de clusters d'entraînement avec Elastic Fabric Adapter (EFA) pour un réseau à hautes performances et sur la création de clusters

d'inférence avec des instances AWS Inferentia, y compris les conditions préalables, les procédures et les considérations relatives au déploiement. step-by-step

Rubriques

- Utiliser l'accélérateur optimisé pour EKS AMIs pour les instances GPU
- Installez le plugin d'appareil Kubernetes pour GPUs
- Exécutez des conteneurs accélérés par GPU (Windows sur EC2 G-Series)
- Exécuter un entraînement de machine learning sur Amazon EKS avec Elastic Fabric Adapter
- Utiliser des instances AWS Inferentia avec Amazon EKS pour le Machine Learning

Utiliser l'accélérateur optimisé pour EKS AMIs pour les instances GPU

Amazon EKS prend en charge Amazon Linux et AMIs Bottlerocket optimisés pour EKS pour les instances GPU. Les accélérateurs optimisés pour EKS AMIs simplifient l'exécution des charges de travail d'intelligence artificielle et de machine learning dans les clusters EKS en fournissant des images de système d'exploitation prédéfinies et validées pour la pile Kubernetes accélérée. Outre les principaux composants Kubernetes inclus dans la version standard optimisée pour EKS AMIs, l'accélérateur optimisé pour EKS AMIs inclut les modules de noyau et les pilotes nécessaires pour exécuter le GPU NVIDIA et les instances EC2, ainsi que les instances AWS GPU GInferentia et P Trainium EC2 dans les clusters EKS.

Le tableau ci-dessous indique les types d'instances GPU pris en charge pour chaque variante d'AMI accélérée optimisée pour EKS. Consultez les AL2023 versions optimisées pour EKS et les versions de Bottlerocket GitHub pour connaître les dernières mises à jour des variantes de l'AMI.

Variante EKS AMI	Types d'instances EC2
AL2023 x86_64 NVIDIA	p6-b300, p6-b200, p5, p5e, p5en, p4d, p4de, p3, p3dn, g7e, gr6, g6, g6e, g6f, gr6f, g5, g4dn
AL2023 ARM NVIDIA	p6e-gb200, 5 g, 5 g
AL2023 Neurone x86_64	inf1, inf2, trn1, trn2
Bottlerocket x86_64 aws-k8s-nvidia	p6-b300, p6-b200, p5, p5e, p5en, p4d, p4de, p3, p3dn, g7e, gr6, g6, g6e, g6f, gr6f, g5, g4dn

Variante EKS AMI	Types d'instances EC2
Bottlerocket aarch64/arm64 aws-k8s-nvidia	g/5g
Bottlerocket x86_64 aws-k8s	inf1, inf2, trn1, trn2

NVIDIA optimisé pour EKS AMIs

En utilisant le NVIDIA optimisé pour EKS AMIs, vous acceptez le contrat de licence utilisateur final (EULA) Cloud de NVIDIA.

Pour découvrir la dernière version de NVIDIA optimisée pour EKS AMIs, consultez the section called "Recevez les dernières nouvelles IDs" et. the section called "Obtenir les derniers identifiants"

Lorsque vous utilisez Amazon Elastic Fabric Adaptor (EFA) avec EKS Optimized AL2023 ou Bottlerocket NVIDIA AMIs, vous devez installer le plug-in pour appareil EFA séparément. Pour de plus amples informations, veuillez consulter the section called "Configurer des clusters d'entraînement avec EFA".

EKS AL2023 NVIDIA AMIs

Lorsque vous utilisez l'opérateur GPU NVIDIA avec le AL2023 NVIDIA optimisé pour EKS AMIs, vous devez désactiver l'installation du pilote et du kit d'outils par l'opérateur, car ceux-ci sont déjà inclus dans l'EKS. AMIs Les AL2023 cartes NVIDIA optimisées pour AMIs EKS n'incluent pas le plug-in de périphérique NVIDIA Kubernetes ni le pilote NVIDIA DRA, et ceux-ci doivent être installés séparément. Pour de plus amples informations, veuillez consulter the section called "Installer le plugin pour appareil NVIDIA Kubernetes".

Outre les composants EKS AMI standard, le AL2023 NVIDIA optimisé pour EKS AMIs inclut les composants suivants.

- Pilote NVIDIA
- Pilote en mode utilisateur NVIDIA CUDA
- Boîte à outils pour conteneurs NVIDIA
- Gestionnaire de tissus NVIDIA
- NVIDIA a persisté
- pilote NVIDIA IMEX

- Gestionnaire de NVLink sous-réseaux NVIDIA
- EFA minimal (module noyau et rdma-core)

Pour plus de détails sur le pilote de mode utilisateur NVIDIA CUDA et le CUDA runtime/libraries utilisés dans les conteneurs d'applications, consultez la documentation NVIDIA. La version CUDA présentée ici `nvidia-smi` est la version du pilote de mode utilisateur NVIDIA CUDA installé sur l'hôte, qui doit être compatible avec le CUDA runtime/libraries utilisé dans les conteneurs d'applications.

Le noyau de AMIs support AL2023 NVIDIA 6.12 optimisé pour EKS pour les versions 1.33 et supérieures de Kubernetes, et la version 580 du pilote NVIDIA pour toutes les versions de Kubernetes. Le pilote NVIDIA 580 est requis pour utiliser CUDA 13+.

Consultez les AL2023 versions optimisées pour EKS sur GitHub pour plus de détails sur les versions des composants incluses dans le. AMIs Consultez le script d'installation de l'AMI EKS AL2023 NVIDIA et le script de chargement du noyau pour plus de détails sur la façon dont les AMI EKS configurent les dépendances NVIDIA. Vous pouvez trouver la liste des packages installés et de leurs versions sur une instance EC2 en cours d'exécution à l'aide de la `dnf list installed` commande.

Lors de la création personnalisée AMIs avec l'EKS Optimized AMIs comme base, il n'est pas recommandé ou pris en charge d'exécuter une mise à niveau du système d'exploitation (par ex. `dnf upgrade`) ou mettez à niveau l'un des packages Kubernetes ou GPU inclus dans l'EKS Optimized AMIs, car cela risque de compromettre la compatibilité des composants. Si vous mettez à niveau le système d'exploitation ou les packages inclus dans l'EKS Optimized AMIs, il est recommandé de procéder à des tests approfondis dans un environnement de développement ou de préparation avant le déploiement en production.

Lors de la création d'instances personnalisées AMIs pour le GPU, il est recommandé de créer une AMIs configuration personnalisée distincte pour chaque génération et famille d'instances que vous allez exécuter. L'accélérateur optimisé pour EKS installe de AMIs manière sélective les pilotes et les packages au moment de l'exécution en fonction de la génération et de la famille du type d'instance sous-jacent. Pour plus d'informations, consultez les scripts EKS AMI pour l'installation et l'exécution.

EKS Bottlerocket NVIDIA AMIs

Lorsque vous utilisez l'opérateur GPU NVIDIA avec le Bottlerocket NVIDIA optimisé pour EKS AMIs, vous devez désactiver l'installation par l'opérateur du pilote, du kit d'outils et du plug-in de périphérique, car ceux-ci sont déjà inclus dans l'EKS. AMIs

Outre les composants EKS AMI standard, le Bottlerocket NVIDIA AMIs optimisé pour EKS inclut les composants suivants. Les dépendances minimales pour EFA (module noyau et rdma-core) sont installées dans toutes les variantes de Bottlerocket.

- Plug-in pour appareil NVIDIA Kubernetes
- Pilote NVIDIA
- Pilote en mode utilisateur NVIDIA CUDA
- Boîte à outils pour conteneurs NVIDIA
- Gestionnaire de tissus NVIDIA
- NVIDIA a persisté
- pilote NVIDIA IMEX
- Gestionnaire de NVLink sous-réseaux NVIDIA
- Gestionnaire NVIDIA MIG

Pour plus de détails sur le pilote de mode utilisateur NVIDIA CUDA et le CUDA runtime/libraries utilisés dans les conteneurs d'applications, consultez la documentation NVIDIA. La version CUDA présentée ici `nvidia-smi` est la version du pilote de mode utilisateur NVIDIA CUDA installé sur l'hôte, qui doit être compatible avec le CUDA runtime/libraries utilisé dans les conteneurs d'applications.

Consultez les informations de version de Bottlerocket dans la documentation de Bottlerocket pour plus de détails sur les packages installés et leurs versions. Le Bottlerocket NVIDIA, optimisé pour EKS, prend en AMIs charge le noyau 6.12 pour les versions 1.33 et supérieures de Kubernetes, et la version 580 du pilote NVIDIA pour les versions 1.34 et supérieures de Kubernetes. Le pilote NVIDIA 580 est requis pour utiliser CUDA 13+.

Neuron optimisé pour EKS AMIs

Pour en savoir plus sur la façon d'exécuter des charges de travail d'entraînement et d'inférence à l'aide de Neuron avec Amazon EKS, consultez les références suivantes :

- Conteneurs - Kubernetes - Commencer à utiliser la documentation Neuron AWS
- Exemple de formation dans AWS Neuron EKS Samples on GitHub
- Déployez des charges de travail d'inférence ML avec Inferentia sur Amazon EKS

Pour trouver le dernier Neuron optimisé pour EKS AMIs, consultez et. the section called “Recevez les dernières nouvelles IDs” the section called “Obtenir les derniers identifiants”

Lorsque vous utilisez Amazon Elastic Fabric Adaptor (EFA) avec l'EKS Optimized AL2023 ou le Bottlerocket Neuron AMIs, vous devez installer le plug-in pour appareil EFA séparément. Pour de plus amples informations, veuillez consulter the section called “Configurer des clusters d'entraînement avec EFA”.

AL2023 Neurone EKS AMIs

Les AL2023 Neuron optimisés pour EKS n'incluent AMIs pas le plug-in de périphérique Neuron Kubernetes ni l'extension du planificateur Neuron Kubernetes, et ceux-ci doivent être installés séparément. Pour de plus amples informations, veuillez consulter the section called “Installer le plugin pour appareil Neuron Kubernetes”.

Outre les composants EKS AMI standard, le AL2023 Neuron optimisé pour EKS AMIs inclut les composants suivants.

- pilote neuronal () aws-neuronx-dkms
- Outils Neuron () aws-neuronx-tools
- EFA minimal (module noyau et rdma-core)

Consultez le script d'installation de l'AMI EKS AL2023 Neuron pour plus de détails sur la façon dont l'EKS AMIs configure les dépendances des neurones. Consultez les AL2023 versions optimisées pour EKS ci-dessous GitHub pour voir les versions des composants incluses dans les AMI. Vous pouvez trouver la liste des packages installés et de leurs versions sur une instance EC2 en cours d'exécution à l'aide de la `dnf list installed` commande.

EKS Bottlerocket Neuron AMIs

Les variantes standard de Bottlerocket (aws-k8s) incluent les dépendances Neuron qui sont automatiquement détectées et chargées lors de l'exécution sur des instances Inferentia ou Trainium EC2. AWS

Les Bottlerocket optimisés pour EKS n'incluent AMIs pas le plug-in d'appareil Neuron Kubernetes ni l'extension de planificateur Neuron Kubernetes, et ceux-ci doivent être installés séparément. Pour de plus amples informations, veuillez consulter the section called “Installer le plugin pour appareil Neuron Kubernetes”.

Outre les composants EKS AMI standard, le Bottlerocket AMIs Neuron optimisé pour EKS inclut les composants suivants.

- pilote neuronal () aws-neuronx-dkms
- EFA minimal (module noyau et rdma-core)

Lorsque vous utilisez le Bottlerocket optimisé pour EKS AMIs avec des instances de Neuron, les éléments suivants doivent être configurés dans les données utilisateur de Bottlerocket. Ce paramètre permet au conteneur de s'approprier le dispositif Neuron monté en fonction des `runAsGroup` valeurs `runAsUser` et fournies dans la spécification de charge de travail. Pour plus d'informations sur le support des neurones dans Bottlerocket, consultez le fichier Quickstart on EKS readme on. GitHub

```
[settings]
[settings.kubernetes]
device-ownership-from-security-context = true
```

Consultez le journal des modifications du kit de noyau Bottlerocket pour plus d'informations sur la version du pilote Neuron incluse dans le Bottlerocket optimisé pour EKS. AMIs

Installez le plugin d'appareil Kubernetes pour GPUs

Les plug-ins d'appareils Kubernetes ont été le principal mécanisme utilisé pour promouvoir des infrastructures spécialisées telles que les interfaces réseau et les adaptateurs réseau en tant que GPUs ressources consommables pour les charges de travail Kubernetes. Alors que l'allocation dynamique des ressources (DRA) est considérée comme le futur de la gestion des appareils dans Kubernetes, la plupart des fournisseurs d'infrastructures spécialisées prennent très tôt en charge les pilotes DRA. Les plug-ins pour appareils Kubernetes restent une approche largement disponible à utiliser GPUs dans les clusters Kubernetes aujourd'hui.

Considérations

- Lorsque vous utilisez le AL2 023 optimisé pour EKS AMIs avec NVIDIA GPUs, vous devez installer le plug-in pour appareil NVIDIA Kubernetes. Vous pouvez installer et gérer le plug-in pour appareil NVIDIA Kubernetes avec Helm, l'outil Kubernetes de votre choix ou l'opérateur du GPU NVIDIA.
- Lorsque vous utilisez le Bottlerocket optimisé pour EKS AMIs avec NVIDIA GPUs, vous n'avez pas besoin d'installer le plug-in pour appareil NVIDIA Kubernetes, car il est déjà inclus dans le Bottlerocket optimisé pour EKS. AMIs Cela inclut lorsque vous utilisez des instances GPU avec le mode automatique EKS.

- Lorsque vous utilisez le AL2 023 ou le Bottlerocket optimisé pour EKS AMIs avec AWS Inferentia ou Trainium GPUs, vous devez installer le plug-in de périphérique Neuron Kubernetes et éventuellement installer l'extension de planificateur Neuron Kubernetes. Pour plus d'informations, consultez la documentation Neuron relative à l'exécution sur EKS.

Installer le plugin pour appareil NVIDIA Kubernetes

La procédure suivante décrit comment installer le plug-in pour appareil NVIDIA Kubernetes et exécuter un exemple de test sur des instances de GPU NVIDIA.

Conditions préalables

- Cluster EKS créé
- Nœuds GPU NVIDIA exécutés dans le cluster à l'aide de l'AMI NVIDIA AL2 023 optimisée pour EKS
- Helm installé dans votre environnement de ligne de commande, consultez les Instructions de configuration de Helm.

Procédure

1. Ajoutez le référentiel de `nvdp` diagrammes Helm.

```
helm repo add nvdp https://nvidia.github.io/k8s-device-plugin
```

2. Mettez à jour votre référentiel Helm local pour vous assurer que vous disposez des graphiques les plus récents.

```
helm repo update
```

3. Téléchargez la dernière version du plugin pour appareil NVIDIA Kubernetes

```
helm search repo nvdp --devel
```

```
NAME                        CHART VERSION APP VERSION DESCRIPTION
nvdp/gpu-feature-discovery 0.17.4        0.17.4      ...
nvdp/nvidia-device-plugin  0.17.4        0.17.4      ...
```

4. Installez le plug-in pour appareil NVIDIA Kubernetes sur votre cluster, en le `0.17.4` remplaçant par la dernière version de la commande ci-dessus.

```
helm install nvdp nvdp/nvidia-device-plugin \
  --namespace nvidia \
  --create-namespace \
  --version 0.17.4 \
  --set gfd.enabled=true
```

5. Vérifiez que le plug-in de l'appareil NVIDIA Kubernetes est en cours d'exécution dans votre cluster. La sortie ci-dessous montre la sortie avec deux nœuds dans le cluster.

```
kubectl get ds -n nvidia nvdp-nvidia-device-plugin
```

```
NAME                        DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE
 SELECTOR   AGE
nvdp-nvidia-device-plugin   2         2         2       2            2
 <none>          11m
```

6. Vérifiez que vos nœuds sont allouables à l' GPUs aide de la commande suivante.

```
kubectl get nodes "-o=custom-
columns=NAME:.metadata.name,GPU:.status.allocatable.nvidia\.com/gpu"
```

```
NAME                                          GPU
ip-192-168-11-225.us-west-2.compute.internal   1
ip-192-168-24-96.us-west-2.compute.internal    1
```

7. Créez un fichier nommé `nvidia-smi.yaml` avec les contenus suivants. Ce manifeste lance une image de conteneur AL2 023 minimale qui s'exécute `nvidia-smi` sur un nœud.

```
apiVersion: v1
kind: Pod
metadata:
  name: nvidia-smi
spec:
  restartPolicy: OnFailure
  containers:
    - name: gpu-demo
      image: public.ecr.aws/amazonlinux/amazonlinux:2023-minimal
      command: ['/bin/sh', '-c']
```

```
      args: ['nvidia-smi && tail -f /dev/null']
      resources:
        limits:
          nvidia.com/gpu: 1
  tolerations:
    - key: 'nvidia.com/gpu'
      operator: 'Equal'
      value: 'true'
      effect: 'NoSchedule'
```

8. Appliquez le manifeste ci-dessus avec la commande suivante.

```
kubectl apply -f nvidia-smi.yaml
```

9. Une fois que le pod n'est plus en cours d'exécution, affichez ses journaux à l'aide de la commande suivante.

```
kubectl logs nvidia-smi
```

L'exemple qui suit illustre un résultat.

```
+-----------------------------------------------------------------------------------------
+
| NVIDIA-SMI XXX.XXX.XX            Driver Version: XXX.XXX.XX     CUDA Version: XX.X
     |
|-----------------------------------------+------------------------
+----------------------+
| GPU  Name                 Persistence-M | Bus-Id          Disp.A | Volatile Uncorr.
 ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |           Memory-Usage | GPU-Util
 Compute M. |
|                                         |                        |
 MIG M. |
|=========================================+========================
+======================|
|   0  NVIDIA L4                      On  |   00000000:31:00.0 Off |
   0 |
| N/A   27C    P8             11W /   72W |       0MiB /  23034MiB |      0%
 Default |
|                                         |                        |
 N/A |
```

```
+-----------------------------------------+-----------------------
+----------------------+

+-----------------------------------------------------------------------------------------
+
| Processes:
     |
|  GPU   GI   CI              PID   Type   Process name                          GPU
 Memory |
|        ID   ID                                                               Usage
     |
|
=============================================================================================|
|  No running processes found
     |
+-----------------------------------------------------------------------------------------
+
```

Installer le plugin pour appareil Neuron Kubernetes

La procédure suivante décrit comment installer le plug-in de périphérique Neuron Kubernetes et exécuter un exemple de test sur une instance d'Inferentia.

Conditions préalables

- Cluster EKS créé
- Nœuds GPU Neuron exécutés dans le cluster à l'aide de l'AMI AL2 023 Neuron ou de l'AMI Bottlerocket optimisées pour EKS
- Helm installé dans votre environnement de ligne de commande, consultez les Instructions de configuration de Helm.

Procédure

1. Installez le plug-in pour appareil Neuron Kubernetes sur votre cluster.

```
helm upgrade --install neuron-helm-chart oci://public.ecr.aws/neuron/neuron-helm-
chart \
    --set "npd.enabled=false"
```

2. Vérifiez que le plug-in de l'appareil Neuron Kubernetes est en cours d'exécution dans votre cluster. La sortie ci-dessous montre la sortie avec un seul nœud neuronal dans le cluster.

```
kubectl get ds -n kube-system neuron-device-plugin
```

```
NAME                   DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE
 SELECTOR   AGE
neuron-device-plugin   1         1         1       1            1           <none>
       72s
```

3. Vérifiez que vos nœuds sont allouables à l' NueronCores aide de la commande suivante.

```
kubectl get nodes "-o=custom-
columns=NAME:.metadata.name,NeuronCore:.status.allocatable.aws\.amazon\.com/
neuroncore"
```

```
NAME                                          NeuronCore
ip-192-168-47-173.us-west-2.compute.internal   2
```

4. Vérifiez que vos nœuds sont allouables à l' NueronDevices aide de la commande suivante.

```
kubectl get nodes "-o=custom-
columns=NAME:.metadata.name,NeuronDevice:.status.allocatable.aws\.amazon\.com/neuron"
```

```
NAME                                          NeuronDevice
ip-192-168-47-173.us-west-2.compute.internal   1
```

5. Créez un fichier nommé `neuron-ls.yaml` avec les contenus suivants. Ce manifeste lance un conteneur Neuron Monitor sur lequel l'`neuron-ls`outil est installé.

```
apiVersion: v1
kind: Pod
metadata:
  name: neuron-ls
spec:
  restartPolicy: Never
  containers:
  - name: neuron-container
    image: public.ecr.aws/g4h4h0b5/neuron-monitor:1.0.0
    command: ["/bin/sh"]
```

```
    args: ["-c", "neuron-ls"]
    resources:
      limits:
        aws.amazon.com/neuron: 1
  tolerations:
  - key: "aws.amazon.com/neuron"
    operator: "Exists"
    effect: "NoSchedule"
```

6. Appliquez le manifeste ci-dessus avec la commande suivante.

```
kubectl apply -f neuron-ls.yaml
```

7. Une fois que le pod n'est plus en cours d'exécution, affichez ses journaux à l'aide de la commande suivante.

```
kubectl logs neuron-ls
```

Un exemple de résultat est présenté ci-dessous.

```
instance-type: inf2.xlarge
instance-id: ...
+--------+--------+--------+---------+
| NEURON | NEURON | NEURON |   PCI   |
| DEVICE | CORES  | MEMORY |   BDF   |
+--------+--------+--------+---------+
| 0      | 2      | 32 GB  | 00:1f.0 |
+--------+--------+--------+---------+
```

Exécutez des conteneurs accélérés par GPU (Windows sur EC2 G-Series)

> **Important**
>
> Le plug-in Kubernetes Device Plugin for DirectX de TensorWorks est un outil tiers qui n'est ni approuvé, ni pris en charge, ni maintenu par AWS. AWS n'assume aucune responsabilité quant à la sécurité, la fiabilité ou les performances de ce plug-in.

Découvrez comment exécuter des charges de travail de conteneurs Windows accélérées par GPU sur Amazon EKS (Elastic Kubernetes Service) à l'aide de GPU NVIDIA avec le plug-in Kubernetes Device Plugin for DirectX de TensorWorks. Pour plus d'informations, consultez Kubernetes Device Plugin for DirectX.

Il existe deux approches principales pour configurer l'accélération GPU pour vos conteneurs Windows :

- Option 1 : créez une AMI EKS personnalisée optimisée pour Windows avec les pilotes de GPU requis préinstallés.
  - Utilisez cette approche lorsque vous avez besoin d'un environnement cohérent et préconfiguré, prêt à exécuter des conteneurs Windows accélérés par GPU, et que vous êtes en mesure d'investir les efforts supplémentaires nécessaires pour créer et maintenir l'AMI personnalisée.
- Option 2 : installez les pilotes GPU nécessaires sur vos composants master EKS après le lancement de votre instance.
  - Utilisez cette approche lorsque vous souhaitez un processus de configuration plus simple et que vous n'avez pas d'objection à installer les pilotes GPU sur chaque nouveau composant master. Plus adapté à un environnement de développement lorsque vous évaluez ou prototypez des charges de travail accélérées par GPU.

Les deux approches peuvent être mises à profit en suivant les étapes décrites dans ce guide.

Considérations

Ce guide fournit les étapes à suivre pour installer et configurer l'accélération GPU pour vos conteneurs Windows à l'aide des GPU NVIDIA, des pilotes NVIDIA GRID et du plug-in Kubernetes Device Plugin for DirectX de TensorWorks. Les étapes ont été testées et vérifiées afin de fournir une accélération GPU pour vos charges de travail de conteneurs Windows sur Amazon EKS. Consultez the section called "Limitations connues" pour plus d'informations sur les pilotes et les plug-ins d'appareils compatibles. Avant de démarrer, notez les points suivants :

- Seuls les types d'instances de la famille G dotés de pilotes NVIDIA GRID ont été testés et vérifiés pour fonctionner avec ce guide. Bien que d'autres types d'instances et combinaisons de pilotes puissent également être capables d'exécuter des conteneurs Windows accélérés par GPU, ils peuvent nécessiter des étapes de configuration supplémentaires qui ne sont pas abordées dans ce guide.

- Seules les charges de travail basées sur DirectX ont été testées et vérifiées pour fonctionner avec ce guide. Bien que d'autres API GPU telles que OpenGL, Vulkan et OpenCL puissent être compatibles avec l'exécution de conteneurs Windows accélérés par GPU, elles peuvent nécessiter des étapes de configuration supplémentaires qui ne sont pas abordées dans ce guide.
- Il existe certaines limitations connues dont il faut tenir compte avant d'exécuter des conteneurs Windows accélérés par GPU. Pour plus d'informations, consultez la section the section called "Limitations connues".

Prérequis

Pour activer l'accélération GPU pour vos conteneurs Windows sur Amazon EKS, vous devez remplir les conditions suivantes avant de continuer :

- Lancez un cluster Amazon EKS avec Kubernetes v1.27 ou une version plus récente.
- Fournir des nœuds Windows avec Windows Server 2022 ou une version plus récente.
- Fournissez des nœuds Windows dans la famille G de types d'instances, tels que G4 ou G5.
- Provisionnez des nœuds Windows avec une exécution de conteneur avec containerd `1.7.x` ou `2.x.x`. (Consultez the section called "Obtenir les informations sur la version" pour vérifier la version de containerd dans votre AMI optimisée Amazon EKS.)

Installation du pilote GPU sur chaque nœud Windows

Pour installer les pilotes NVIDIA GRID sur vos composants master EKS, suivez les étapes décrites dans Pilotes NVIDIA pour votre instance Amazon EC2. Accédez aux options d'installation – Option 3 : Pilotes GRID et suivez les étapes d'installation.

Installation pour Windows Server Core

Pour Windows Server Core, qui ne dispose pas d'une expérience de bureau, installez les pilotes NVIDIA GRID en mode silencieux à l'aide des commandes suivantes :

```
$nvidiaInstallerFilePath = nvidia-driver-installer.exe # Replace with path to installer
$installerArguments = "-s -clean -noreboot -noeula"
Start-Process -FilePath $nvidiaInstallerFilePath -ArgumentList $installerArguments -
Wait -NoNewWindow -PassThru
```

Vérification de votre installation

Exécutez la commande PowerShell suivante pour afficher les informations de diagnostic relatives aux GPU de l'instance :

```
nvidia-smi
```

Cette commande affiche la version du pilote NVIDIA, ainsi que des informations sur le matériel GPU. Assurez-vous que le résultat de cette commande correspond à la version du pilote NVIDIA GRID que vous pensiez avoir installée.

Déploiement du plug-in du périphérique GPU sur chaque nœud

Pour permettre la détection et l'exposition des ressources GPU aux conteneurs sur vos nœuds Windows, vous aurez besoin d'un plug-in de périphérique. Déployez le plug-in DirectX Device de Tensorworks sur chaque composant master en l'exécutant en tant que DaemonSet dans votre cluster EKS. Suivez le guide d'installation spécifié dans le fichier README.md, qui comprend les étapes suivantes. Il est recommandé de :

- Déployez le plug-in de l'appareil dans l'espace de noms `kube-system`.
- Définissez des limites de ressources appropriées pour le DaemonSet afin de vous assurer qu'il ne consomme pas trop de ressources sur vos nœuds.

> **Note**
>
> Le plug-in DaemonSet s'exécutera sur chaque nœud en tant que conteneur de processus hôte avec des privilèges élevés. Il est recommandé de mettre en œuvre des contrôles RBAC pour restreindre l'accès à ce DaemonSet afin que seuls les utilisateurs autorisés puissent exécuter des commandes privilégiées.

Lors de l'exécution de conteneurs accélérés par GPU, le plug-in de périphérique prend en charge deux modes :

- Mode à locataire unique : ce mode consacre toutes les ressources GPU à un seul conteneur sur l'instance. Installez les plug-ins de périphérique avec prise en charge de la location unique à l'aide de la commande suivante. Consultez README.md pour plus d'informations.

  ```
  kubectl apply -f "https://raw.githubusercontent.com/TensorWorks/directx-device-plugins/main/deployments/default-daemonsets.yml"
  ```

- Mode à locataire multiple : ce mode permet de partager les ressources GPU entre plusieurs conteneurs de l'instance. Installez les plug-ins de périphériques avec prise en charge multi-locataires à l'aide de la commande suivante. Consultez README.md pour plus d'informations.

```
kubectl apply -f "https://raw.githubusercontent.com/TensorWorks/directx-device-plugins/main/deployments/multitenancy-inline.yml"
```

  Vous pouvez également utiliser un ConfigMap pour spécifier la mutualisation.

```
kubectl apply -f "https://raw.githubusercontent.com/TensorWorks/directx-device-plugins/main/deployments/multitenancy-configmap.yml"
```

Vérification du déploiement du plug-in de l'appareil

Après avoir déployé le plug-in du périphérique, remplacez `<namespace>` et exécutez la commande suivante pour vérifier que le plug-in DirectX Device Plugin fonctionne correctement sur tous vos nœuds Windows.

```
kubectl get ds device-plugin-wddm -n <namespace>
```

Vérifier que les conteneurs sont prêts à être déployés

Une fois que le plug-in DaemonSet du périphérique est exécuté sur les composants master Windows équipés de GPU, utilisez la commande suivante pour vérifier que chaque nœud dispose de GPU allouables. Le nombre correspondant doit correspondre au nombre de périphériques DirectX sur chaque nœud.

```
kubectl get nodes "-o=custom-columns=NAME:.metadata.name,DirectX:.status.allocatable.directx\.microsoft\.com/display"
```

Exécution de conteneurs Windows avec accélération par GPU

Avant de lancer vos pods, spécifiez le nom de la ressource `directx.microsoft.com/display` dans `.spec.containers[].resources`. Cela indiquera que vos conteneurs nécessitent des capacités compatibles avec le GPU, et le `kube-scheduler` tentera de placer vos pods sur votre nœud Windows préconfiguré disposant de ressources GPU disponibles.

À titre d'exemple, voir la commande ci-dessous qui lance un `Job` pour exécuter une simulation Monte Carlo afin d'estimer la valeur de pi. Cet exemple est tiré du référentiel GitHub Kubernetes Device Plugins for DirectX, qui propose plusieurs exemples que vous pouvez exécuter pour tester les capacités GPU de votre nœud Windows.

```
cat <<EOF | kubectl apply -f -
apiVersion: batch/v1
kind: Job
metadata:
  name: example-cuda-montecarlo-wddm
spec:
  template:
    spec:
      containers:
      - name: example-cuda-montecarlo-wddm
        image: "index.docker.io/tensorworks/example-cuda-montecarlo:0.0.1"
        resources:
          limits:
            directx.microsoft.com/display: 1
      nodeSelector:
        "kubernetes.io/os": windows
      restartPolicy: Never
  backoffLimit: 0
EOF
```

Limitations connues

Tous les GPU sont utilisables

Tous les GPU de l'instance seront utilisables par chaque conteneur en cours d'exécution sur l'hôte, même si vous demandez un nombre spécifique de GPU pour un conteneur donné. De plus, par défaut, tous les conteneurs exécutés sur l'hôte utilisent le GPU avec l'index 0, même si plusieurs GPU sont disponibles sur le nœud. Ainsi, pour que les tâches multi-GPU fonctionnent correctement, vous devez désigner explicitement le périphérique GPU spécifique à utiliser dans le code de votre application.

La mise en œuvre exacte pour attribuer un périphérique à utiliser pour l'application dépendra du langage de programmation ou du cadre que vous utilisez. Par exemple, si vous utilisez la programmation CUDA, pour sélectionner un GPU spécifique, vous pouvez spécifier explicitement le périphérique à utiliser dans le code de votre application à l'aide de la fonction cudaSetDevice().

La nécessité de spécifier explicitement le périphérique est due à un problème connu affectant les conteneurs Windows. Vous pouvez suivre les progrès réalisés dans la résolution de ce problème dans le numéro #333 de Microsoft/Windows-Containers. Le tableau suivant représente une représentation visuelle et un exemple pratique de ce comportement d'allocation de GPU.

Considérons un scénario dans lequel il existe un seul nœud Windows de l'instance EC2 type `g4dn.12xlarge`, équipé de quatre GPU. Imaginons un scénario dans lequel trois pods sont lancés sur cette instance. Le tableau montre que, quel que soit le nombre de GPU demandé par chaque conteneur, les trois pods ont accès aux quatre GPU de l'instance et utilisent par défaut le GPU avec l'index de périphérique 0.

Pod	GPU demandés	Accès GPU réel	Utilisation de GPU par défaut	Indices de GPU disponibles	Total des GPU d'instance
Pod 1	1 GPU	Les 4 GPU	GPU avec indice 0	0, 1, 2, 3	4
Pod 2	2 GPU	Les 4 GPU	GPU avec indice 0	0, 1, 2, 3	4
Pod 3	1 GPU	Les 4 GPU	GPU avec indice 0	0, 1, 2, 3	4

Prise en charge du plug-in Kubernetes

L'implémentation officielle par NVIDIA du plug-in pour appareil Kubernetes n'est pas compatible avec Windows. Vous pouvez suivre la progression de l'ajout du support officiel pour Windows dans le numéro #419 de NVIDIA/k8s-device-plug-in.

Limitations des instances de calcul GPU

En fonction de la configuration de votre compte AWS, vous pouvez être soumis à des limites de service concernant le nombre et les types d'instances de calcul GPU Amazon EC2 que vous pouvez lancer. Si vous avez besoin d'une capacité supplémentaire, vous pouvez demander une augmentation de quota.

Doit créer une AMI optimisée pour le GPU pour Windows

Amazon EKS ne fournit aucun composant géré EKS Windows GPU Optimized AMI ou EC2 Image Builder. Vous devrez suivre les étapes décrites dans ce guide pour créer une AMI EKS Windows Optimized personnalisée avec les pilotes GPU requis préinstallés, ou installer les pilotes GPU nécessaires sur vos composants master EKS après avoir lancé vos instances.

Inferentia et Trainium ne sont pas pris en charge

Les charges de travail basées sur AWS Inferentia et AWSTrainium ne sont pas prises en charge sous Windows.

Exécuter un entraînement de machine learning sur Amazon EKS avec Elastic Fabric Adapter

Cette rubrique décrit comment intégrer Elastic Fabric Adapter (EFA) aux pods déployés dans votre cluster Amazon EKS. Elastic Fabric Adapter (EFA) est une interface réseau pour les instances Amazon EC2 qui vous permet d'exécuter des applications nécessitant des niveaux élevés de communications inter-nœuds à grande échelle sur AWS. Son interface matérielle sur mesure de contournement du système d'exploitation améliore les performances des communications entre instances, ce qui est essentiel à la mise à l'échelle de ces applications. Avec EFA, les applications de calcul haute performance (HPC) utilisant l'interface de passage de message (MPI) et les applications Machine Learning (ML) utilisant la bibliothèque de communications collectives (NCCL) de NVIDIA peuvent atteindre des milliers de ou. CPUs GPUs Vous bénéficiez ainsi des performances applicatives des clusters HPC sur site avec l'élasticité et la flexibilité à la demande du AWS cloud. L'intégration d'EFA aux applications exécutées sur des clusters Amazon EKS permet de réduire le temps nécessaire à l'exécution des applications d'entraînement distribué à grande échelle sans avoir à ajouter d'autres instances à votre cluster. Pour plus d'informations sur l'EFA, consultez Elastic Fabric Adapter.

Types d'instances avec EFA

Le plug-in AWS EFA Kubernetes Device prend en charge tous les types d'instances Amazon EC2 dotés d'EFA. Pour consulter la liste de tous les types d'instances qui disposent d'EFA, consultez Types d'instances pris en charge dans le Guide de l'utilisateur Amazon EC2. Toutefois, pour exécuter rapidement des applications ML, nous recommandons qu'une instance dispose de puces d'accélération matérielle telles que les puces NVidia GPUs, AWS Inferentia ou AWS Trainium, en plus de l'EFA. Pour obtenir la liste des types d'instances disposant de puces d'accélération matérielle et d'EFA, consultez Calcul accéléré dans le Guide de l'utilisateur Amazon EC2.

Lorsque vous comparez les types d'instance afin de faire votre choix, tenez compte du nombre de cartes réseau EFA disponibles pour ce type d'instance, ainsi que du nombre de cartes accélératrices, de la quantité de CPU et de la quantité de mémoire. Vous pouvez attribuer un EFA maximum par carte réseau. Une EFA compte comme une interface réseau. Pour connaître le nombre d'EFA disponibles pour chaque type d'instance disposant d'EFA, consultez la liste des cartes réseau dans le Guide de l'utilisateur Amazon EC2.

Interfaces EFA et EFA uniquement

Un Elastic Fabric Adapter (EFA) est une interface réseau qui combine les fonctionnalités d'un adaptateur réseau élastique (ENA) et d'une interface de contournement du système d'exploitation, alimentée par AWS le protocole Scalable Reliable Datagram (SRD). Les fonctionnalités EFA permettent aux applications de communiquer directement avec le matériel pour un transport à faible latence. Vous pouvez choisir d'accéder uniquement aux capacités EFA à l'aide d'interfaces EFA uniquement, limitant ainsi la communication aux interfaces au sein de la même zone de disponibilité.

Pour créer des nœuds pouvant disposer d'interfaces EFA uniquement, vous devez utiliser un modèle de lancement EC2 personnalisé et définir le `InterfaceType` sur `efa-only`. Dans votre modèle de lancement personnalisé, vous ne pouvez pas définir la carte réseau `0` comme interface EFA uniquement, car il s'agit de la carte réseau principale et de l'interface réseau de l'instance EC2. Vous devez disposer de la version `1.18.5` ou ultérieure de CNI VPC pour les interfaces EFA uniquement. Si vous utilisez Amazon Linux 2, la version ami doit être `v20240928` ou ultérieure pour les interfaces EFA uniquement.

La procédure suivante vous guide pour créer un cluster EKS avec des nœuds `eksctl` dotés d'interfaces NVidia GPUs et EFA. Vous ne pouvez pas utiliser `eksctl` pour créer des nœuds et des groupes de nœuds qui utilisent des interfaces EFA uniquement.

Conditions préalables

- Un cluster Amazon EKS existant. Si vous ne disposez pas d'un cluster existant, veuillez en créer un à l'aide de Démarrer. Votre cluster doit être déployé dans un VPC disposant d'au moins un sous-réseau privé ayant suffisamment d'adresses IP disponibles dans lequel déployer des nœuds. Le sous-réseau privé doit disposer d'un accès Internet sortant fourni par un appareil externe, tel qu'une passerelle NAT.

  Si vous prévoyez d'utiliser `eksctl` pour créer votre groupe de nœuds, `eksctl` peut également créer un cluster pour vous.

- Version `2.12.3` ou version ultérieure `1.27.160` ou version ultérieure de l'interface de ligne de AWS commande (AWS CLI) installée et configurée sur votre appareil ou AWS CloudShell. Pour vérifier votre version actuelle, utilisez `aws --version | cut -d / -f2 | cut -d ' ' -f1`. Les gestionnaires de packages tels que `yum` Homebrew pour macOS ont souvent plusieurs versions de retard sur la dernière version de la AWS CLI. `apt-get` Pour installer la dernière version, consultez la section Installation et configuration rapide avec aws configure dans le Guide de l'utilisateur de l'interface de ligne de AWS commande. La version de la AWS CLI installée AWS CloudShell peut également avoir plusieurs versions de retard par rapport à la dernière version. Pour le mettre à jour, consultez la section Installation de la AWS CLI dans votre répertoire de base dans le guide de AWS CloudShell l'utilisateur.
- L'outil de ligne de commande `kubectl` est installé sur votre appareil ou AWS CloudShell. La version peut correspondre à celle utilisée par votre cluster Kubernetes, ou différer d'au plus une version mineure, qu'elle soit antérieure ou plus récente. Par exemple, si la version de votre cluster est `1.29`, vous pouvez utiliser la version `kubectl` `1.28`, `1.29` ou `1.30`. Pour installer ou mettre à niveau `kubectl`, veuillez consulter the section called "Configurer kubectl et eksctl".
- Vous devez avoir installé le plug-in CNI Amazon VPC pour Kubernetes version `1.7.10` ou ultérieure avant de lancer des composants master qui prennent en charge plusieurs adaptateurs Elastic Fabric, tels que `p4d` ou `p5`. Pour plus d'informations sur la mise à jour de votre plug-in CNI Amazon VPC pour Kubernetes, consultez the section called "CNI Amazon VPC".
- Pour les instances p6-b200, vous devez utiliser le plug-in périphérique EFA Device Plugin version v0.5.6 ou ultérieure.

> **Important**
>
> Une considération importante requise pour adopter EFA avec Kubernetes est la configuration et la gestion de Huge Pages en tant que ressource dans le cluster. Pour plus d'informations, consultez Gérer Huge Pages dans la documentation Kubernetes. Les instances Amazon EC2 sur lesquelles le pilote EFA est installé pré-allouent 5128 pages géantes de 2 Mo, que vous pouvez demander comme ressources à utiliser dans vos spécifications de tâche.

Créer un groupe de nœuds

La procédure suivante vous aide à créer un groupe de nœuds avec un groupe de nœuds `p4d.24xlarge` sauvegardé avec des interfaces EFA et GPUDirect RDMA, et à exécuter un exemple de test NVIDIA Collective Communications Library (NCCL) pour les performances NCCL multi-nœuds

à l'aide de. EFAs L'exemple peut être utilisé comme modèle de formation en profondeur distribuée sur Amazon EKS à l'aide de EFAs.

1. Déterminez quels types d'instances Amazon EC2 prenant en charge l'EFA sont disponibles dans la AWS région dans laquelle vous souhaitez déployer des nœuds. *region-code*Remplacez-le par la AWS région dans laquelle vous souhaitez déployer votre groupe de nœuds.

   ```
   aws ec2 describe-instance-types --region region-code \
       --filters Name=network-info.efa-supported,Values=true \
       --query "InstanceTypes[*].[InstanceType]" --output text
   ```

   Lorsque vous déployez des nœuds, le type d'instance que vous souhaitez déployer doit être disponible dans la AWS région dans laquelle se trouve votre cluster.

2. Déterminez dans quelles zones de disponibilité le type d'instance que vous souhaitez déployer est disponible. Dans ce didacticiel, le type d'`p5.48xlarge`instance est utilisé et doit être renvoyé dans la sortie pour la AWS région que vous avez spécifiée à l'étape précédente. Lorsque vous déployez des nœuds dans un cluster de production, remplacez-les *p5.48xlarge* par n'importe quel type d'instance renvoyé à l'étape précédente.

   ```
   aws ec2 describe-instance-type-offerings --region region-code \
       --location-type availability-zone --filters Name=instance-
   type,Values=p4d.24xlarge,p5.48xlarge \
       --query 'InstanceTypeOfferings[*].Location' --output text
   ```

   L'exemple qui suit illustre un résultat.

   ```
   us-west-2a    us-west-2c    us-west-2b
   ```

   Notez les zones de disponibilité renvoyées pour une utilisation ultérieure. Lorsque vous déployez des nœuds sur un cluster, votre VPC doit disposer de sous-réseaux avec des adresses IP disponibles dans l'une des zones de disponibilité renvoyées dans la sortie.

3. Créez un groupe de nœuds à l'aide de `eksctl`. Vous avez besoin d'`0.215.0`une version ou d'une version ultérieure de l'outil de ligne de `eksctl` commande installée sur votre appareil ou AWS CloudShell. Pour installer ou mettre à jour `eksctl`, veuillez consulter Installation dans la documentation de `eksctl`.

   a. Copiez le contenu suivant dans un fichier nommé *efa-cluster.yaml*. Remplacez les exemples de valeurs par les vôtres. Vous pouvez remplacer `p5.48xlarge` par une instance

différente. Dans ce cas, assurez-vous que les valeurs de `availabilityZones` sont des zones de disponibilité renvoyées pour le type d'instance à l'étape 1.

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-efa-cluster
  region: region-code
  version: "1.XX"

iam:
  withOIDC: true

availabilityZones: ["us-west-2a", "us-west-2c"]

managedNodeGroups:
  - name: my-efa-ng
    instanceType: p5.48xlarge
    minSize: 1
    desiredCapacity: 2
    maxSize: 3
    availabilityZones: ["us-west-2a"]
    volumeSize: 300
    privateNetworking: true
    efaEnabled: true
```

b. Créez un groupe de nœuds gérés dans un cluster existant.

```
eksctl create nodegroup -f efa-cluster.yaml
```

Si vous ne disposez pas d'un cluster existant, vous pouvez exécuter la commande suivante pour créer un cluster et le groupe de nœuds.

```
eksctl create cluster -f efa-cluster.yaml
```

> **Note**
>
> Comme le type d'instance utilisé dans cet exemple l'est GPUs, le plug-in pour appareil NVIDIA Kubernetes est `eksctl` automatiquement installé sur chaque instance lorsque vous utilisez Amazon Linux 2. Cela n'est pas nécessaire pour Bottlerocket, car le

plug-in NVIDIA device est créé dans la variante EKS NVIDIA de Bottlerocket. Lorsque `efaEnabled` est défini sur `true` dans la configuration du groupe de nœuds, `eksctl` déploie également automatiquement le plug-in de périphérique EFA sur les nœuds.

Utilisation de Bottlerocket avec EFA

La version 1.28.0 et les versions ultérieures de l'AMI Bottlerocket incluent la prise en charge officielle d'EFA. Pour utiliser Bottlerocket pour les nœuds compatibles EFA, spécifiez `amiFamily: Bottlerocket` dans votre configuration. Si vous devez utiliser un identifiant AMI personnalisé, vous devez utiliser des `nodeGroups` standard au lieu de `managedNodeGroups`.

Voici un exemple de configuration :

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-efa-bottlerocket-cluster
  region: region-code
  version: "1.XX"

iam:
  withOIDC: true

availabilityZones: ["us-west-2a", "us-west-2c"]

managedNodeGroups:
  - name: my-efa-bottlerocket-ng
    instanceType: p5.48xlarge
    minSize: 1
    desiredCapacity: 2
    maxSize: 3
    availabilityZones: ["us-west-2a"]
    volumeSize: 300
    privateNetworking: true
    efaEnabled: true
    amiFamily: Bottlerocket
    bottlerocket:
      enableAdminContainer: true
      settings:
        kernel:
```

```
        sysctl:
          "vm.nr_hugepages": "3000"  # Configures 3000 * 2Mi = 6000Mi hugepages
```

Le paramètre sysctl `vm.nr_hugepages` ci-dessus configure le nombre de pages géantes de 2 Mi. Dans cet exemple, 3000 signifie 3000 * 2 Mi = 6000 Mi de pages géantes.

Vérifier l'installation du plug-in de périphérique EFA

Lorsque vous créez un groupe de nœuds avec `efaEnabled: true`, `eksctl` déploie automatiquement le plug-in du périphérique Kubernetes EFA pour vous. Vous pouvez vérifier que le plug-in de périphérique est installé et fonctionne correctement :

1. Vérifiez le DaemonSet statut :

   ```
   kubectl get daemonsets -n kube-system
   ```

   Exemple de sortie :

   ```
   NAME                                  DESIRED   CURRENT   READY   UP-TO-DATE
    AVAILABLE   NODE SELECTOR   AGE
   aws-efa-k8s-device-plugin-daemonset   2         2         2       2            2
         <none>          6m16s
   ...
   ```

   Ici, le plug-in de l'appareil EFA DaemonSet s'exécute sur deux nœuds. Les deux sont READY et AVAILABLE.
2. Vérifiez ensuite les pods créés par DaemonSet :

   ```
   kubectl get pods -n kube-system -l name=aws-efa-k8s-device-plugin
   ```

   Exemple de sortie :

   ```
   NAME                                        READY   STATUS    RESTARTS   AGE
   aws-efa-k8s-device-plugin-daemonset-d68bs   1/1     Running   0          6m16s
   aws-efa-k8s-device-plugin-daemonset-w4l8t   1/1     Running   0          6m16s
   ```

   Les pods du plug-in de périphérique EFA sont dans un état Running, ce qui confirme que le plug-in est correctement déployé et opérationnel.
3. Vérifiez l'enregistrement des ressources :

Vous pouvez vérifier que la ressource `vpc.amazonaws.com/efa` est enregistrée auprès du kubelet en décrivant les nœuds :

```
kubectl describe nodes
```

Si la ressource EFA est correctement enregistrée, elle apparaîtra dans la liste sous Capacity et Allocatable resources du nœud. Par exemple :

```
Capacity:
  ...
  vpc.amazonaws.com/efa:  4
Allocatable:
  ...
  vpc.amazonaws.com/efa:  4
```

Cette sortie confirme que le nœud reconnaît la ressource EFA, la rendant disponible pour les pods qui la demandent.

(Facultatif) Tester les performances de l'EFA

Nous vous recommandons de tester la configuration EFA. Vous pouvez utiliser les tests NCCL dans le `aws-samples/awsome-distributed-training` référentiel sur. GitHub Les tests NCCL évaluent les performances du réseau à l'aide de la bibliothèque Nvidia Collective Communication Library. Les étapes suivantes permettent de soumettre des tests NCCL sur Amazon EKS.

1. Déployez l'opérateur Kubeflow MPI :

   Pour les tests NCCL, vous pouvez appliquer l'opérateur MPI Kubeflow. L'opérateur MPI facilite l'exécution d'un entraînement distribué de style AllReduce sur Kubernetes. Pour plus d'informations, consultez la section Opérateur MPI activé GitHub.
2. Exécutez le test de performance NCCL multi-nœuds pour vérifier le RDMA/EFA : GPUDirect

   Pour vérifier les performances du NCCL avec GPUDirect RDMA sur EFA, exécutez le test de performance NCCL standard. Pour plus d'informations, consultez le dépôt officiel des tests NCCL sur. GitHub

   Procédez comme indiqué ci-dessous pour exécuter un test de performance NCCL à deux nœuds. Dans l'exemple de tâche de test NCCL, chaque travailleur demande huit GPUs, 5210

Mohugepages-2Mi, quatre et 8 000 Mo de mémoire EFAs, ce qui signifie en fait que chaque travailleur consomme toutes les ressources d'une instance. p5.48xlarge

a. Créez le MPIJob manifeste :

Copiez ce qui suit dans un fichier nommé nccl-tests.yaml :

```
apiVersion: kubeflow.org/v2beta1
kind: MPIJob
metadata:
  name: nccl-tests
spec:
  runPolicy:
    cleanPodPolicy: Running
    backoffLimit: 20
  slotsPerWorker: 8
  mpiReplicaSpecs:
    Launcher:
      replicas: 1
      template:
         spec:
          restartPolicy: OnFailure
          containers:
          - image: public.ecr.aws/hpc-cloud/nccl-tests:latest
            imagePullPolicy: IfNotPresent
            name: test-nccl-launcher
            env:
             - name: PATH
               value: $PATH:/opt/amazon/efa/bin:/usr/bin
            command:
            - /opt/amazon/openmpi/bin/mpirun
            - --allow-run-as-root
            - --tag-output
            - -np
            - "16"
            - -N
            - "8"
            - --bind-to
            - none
            - -x
            - PATH
            - -x
            - LD_LIBRARY_PATH
            - -x
```

```
            - NCCL_DEBUG=INFO
            - -x
            - NCCL_BUFFSIZE=8388608
            - -x
            - NCCL_P2P_NET_CHUNKSIZE=524288
            - -x
            - NCCL_TUNER_PLUGIN=/opt/amazon/ofi-nccl/lib/x86_64-linux-gnu/libnccl-
ofi-tuner.so
            - --mca
            - pml
            - ^cm,ucx
            - --mca
            - btl
            - tcp,self
            - --mca
            - btl_tcp_if_exclude
            - lo,docker0,veth_def_agent
            - /opt/nccl-tests/build/all_reduce_perf
            - -b
            - "8"
            - -e
            - "16G"
            - -f
            - "2"
            - -g
            - "1"
            - -c
            - "1"
            - -n
            - "100"
    Worker:
      replicas: 2
      template:
        spec:
          nodeSelector:
            node.kubernetes.io/instance-type: "p5.48xlarge"
          containers:
          - image: public.ecr.aws/hpc-cloud/nccl-tests:latest
            imagePullPolicy: IfNotPresent
            name: nccl-tests-worker
            volumeMounts:
            - name: shmem
              mountPath: /dev/shm
            resources:
```

```
            limits:
              nvidia.com/gpu: 8
              hugepages-2Mi: 5120Mi
              vpc.amazonaws.com/efa: 32
              memory: 32000Mi
            requests:
              nvidia.com/gpu: 8
              hugepages-2Mi: 5120Mi
              vpc.amazonaws.com/efa: 32
              memory: 32000Mi
        volumes:
        - name: shmem
          hostPath:
            path: /dev/shm
```

b. Appliquez les tests NCCL : MPIJob

Soumettez le `MPIJob` en appliquant le manifeste. Cela créera deux instances Amazon EC2 `p5.48xlarge`.

```
kubectl apply -f nccl-tests.yaml
```

L'exemple qui suit illustre un résultat.

```
mpijob.kubeflow.org/nccl-tests created
```

c. Vérifiez que la tâche a démarré les pods :

Affichez vos pods en cours d'exécution.

```
kubectl get pods
```

L'exemple qui suit illustre un résultat.

```
NAME                             READY   STATUS     RESTARTS   AGE
nccl-tests-launcher-nbql9    0/1     Init:0/1   0          2m49s
nccl-tests-worker-0          1/1     Running    0          2m49s
nccl-tests-worker-1          1/1     Running    0          2m49s
```

L'opérateur MPI crée un pod de lancement et deux pods de travail (un sur chaque nœud).

d. Vérifiez que la tâche s'exécute correctement avec les journaux :

Consultez le journal du pod `nccl-tests-launcher`. Remplacez *nbql9* par la valeur de votre sortie.

```
kubectl logs -f nccl-tests-launcher-nbql9
```

Si le test s'est terminé avec succès, vous pouvez déployer vos applications qui utilisent la bibliothèque Nvidia Collective Communication Library.

Utiliser des instances AWS Inferentia avec Amazon EKS pour le Machine Learning

Cette rubrique décrit comment créer un cluster Amazon EKS avec des nœuds exécutant des instances Amazon EC2 Inf1 et (éventuellement) déployer un exemple d'application. Les instances Amazon EC2 Inf1 sont alimentées par des puces AWS Inferentia, conçues sur mesure AWS pour fournir des performances élevées et une inférence à moindre coût dans le cloud. Les modèles d'apprentissage automatique sont déployés sur des conteneurs à l'aide de AWS Neuron, un kit de développement logiciel (SDK) spécialisé composé d'un compilateur, d'un environnement d'exécution et d'outils de profilage qui optimisent les performances d'inférence par apprentissage automatique des puces Inferentia. AWS Neuron prend en charge les frameworks d'apprentissage automatique populaires tels que TensorFlow PyTorch, et MXNet.

> **Note**
>
> La logique du dispositif neuronal IDs doit être contiguë. Si un pod demandant plusieurs appareils Neuron est planifié sur un type d'`inf1.24xlarge`instance `inf1.6xlarge` ou (qui possède plusieurs appareils Neuron), ce pod ne démarrera pas si le planificateur Kubernetes sélectionne un appareil non contigu. IDs Pour plus d'informations, consultez la section Le périphérique logique IDs doit être contigu. GitHub

Conditions préalables

- Installez `eksctl` sur votre ordinateur. Si vous ne l'avez pas installé, consultez Installation dans la documentation `eksctl`.
- Installez `kubectl` sur votre ordinateur. Pour de plus amples informations, veuillez consulter the section called "Configurer kubectl et eksctl".

- (Facultatif) Installez `python3` sur votre ordinateur. S'il n'est pas installé, consultez les téléchargements Python pour obtenir des instructions d'installation.

Créer un cluster

1. Créez un cluster avec des nœuds d' EC2 instance Amazon Inf1. Vous pouvez remplacer *`inf1.2xlarge`* par n'importe quel type d'instance Inf1. L'`eksctl`utilitaire détecte que vous lancez un groupe de nœuds avec un type d'`Inf1`instance et démarrera vos nœuds à l'aide de l'un des systèmes accélérés optimisés pour Amazon EKS d'Amazon Linux AMIs.

> **Note**
>
> Vous ne pouvez pas utiliser de rôles IAM pour les comptes de service avec TensorFlow Serving.

```
eksctl create cluster \
    --name inferentia \
    --region region-code \
    --nodegroup-name ng-inf1 \
    --node-type inf1.2xlarge \
    --nodes 2 \
    --nodes-min 1 \
    --nodes-max 4 \
    --ssh-access \
    --ssh-public-key your-key \
    --with-oidc
```

> **Note**
>
> Notez la valeur de la ligne suivante de la sortie. Elle est utilisée lors d'une étape ultérieure (facultative).

```
[9]  adding identity "arn:aws: iam::111122223333:role/eksctl-inferentia-nodegroup-ng-in-NodeInstanceRole-FI7HIYS3BS09" to auth ConfigMap
```

Lorsque vous lancez un groupe de nœuds avec des `Inf1` instances, le plug-in d'appareil AWS Neuron Kubernetes est `eksctl` automatiquement installé. Ce plug-in annonce les appareils Neuron comme une ressource système au planificateur Kubernetes, qui peut être demandé par un conteneur. Outre les stratégies IAM de nœud Amazon EKS par défaut, la stratégie d'accès en lecture seule Amazon S3 est ajoutée afin que l'exemple d'application, traité dans une étape ultérieure, puisse charger un modèle formé à partir d'Amazon S3.

2. Assurez-vous que tous les pods ont démarré correctement.

```
kubectl get pods -n kube-system
```

Sortie abrégée :

```
NAME                                   READY   STATUS    RESTARTS   AGE
[...]
neuron-device-plugin-daemonset-6djhp   1/1     Running   0          5m
neuron-device-plugin-daemonset-hwjsj   1/1     Running   0          5m
```

(Facultatif) Déployez une image d'application TensorFlow Serving

Un modèle formé doit être compilé sur une cible Inferentia avant de pouvoir être déployé sur des instances Inferentia. Pour continuer, vous aurez besoin d'un TensorFlow modèle optimisé pour Neuron enregistré dans Amazon S3. Si vous n'en avez pas encore SavedModel, veuillez suivre le didacticiel pour créer un modèle ResNet 50 compatible avec Neuron et télécharger le résultat SavedModel sur S3. ResNet-50 est un modèle d'apprentissage automatique populaire utilisé pour les tâches de reconnaissance d'images. Pour plus d'informations sur la compilation de modèles de neurones, consultez The AWS Inferentia Chip with DLAMI dans le Deep Learning Developer Guide. AWS AMIs

L'exemple de manifeste de déploiement gère un conteneur de service d'inférence prédéfini TensorFlow fourni par AWS Deep Learning Containers. À l'intérieur du conteneur se trouvent le AWS Neuron Runtime et l'application TensorFlow Serving. Une liste complète des Deep Learning Containers prédéfinis optimisés pour Neuron est disponible GitHub sous Images disponibles. Au démarrage, le DLC récupérera votre modèle sur Amazon S3, lancera Neuron TensorFlow Serving avec le modèle enregistré et attendra les demandes de prédiction.

Le nombre d'appareils Neuron alloués à votre application de service peut être ajusté en changeant la ressource `aws.amazon.com/neuron` dans le yaml de déploiement. Veuillez noter que la

communication entre TensorFlow Serving et le runtime Neuron se fait via GRPC, ce qui nécessite de transmettre la `IPC_LOCK` capacité au conteneur.

1. Ajoutez la politique IAM `AmazonS3ReadOnlyAccess` au rôle d'instance de nœud créé à l'étape 1 de la section Créer un cluster. Ceci est nécessaire pour que l'application exemple puisse charger un modèle formé à partir de Amazon S3.

```
aws iam attach-role-policy \
    --policy-arn arn:aws: iam::aws:policy/AmazonS3ReadOnlyAccess \
    --role-name eksctl-inferentia-nodegroup-ng-in-NodeInstanceRole-FI7HIYS3BS09
```

2. Créez un fichier nommé `rn50_deployment.yaml` avec les contenus suivants. Mettez à jour le code régional et le chemin du modèle pour correspondre aux paramètres souhaités. Le nom du modèle est utilisé à des fins d'identification lorsqu'un client fait une demande au TensorFlow serveur. Cet exemple utilise un nom de modèle correspondant à un exemple de ResNet 50 scripts client qui sera utilisé ultérieurement pour envoyer des demandes de prédiction.

```
aws ecr list-images --repository-name neuron-rtd --registry-id 790709498068 --region
 us-west-2
```

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: eks-neuron-test
  labels:
    app: eks-neuron-test
    role: master
spec:
  replicas: 2
  selector:
    matchLabels:
      app: eks-neuron-test
      role: master
  template:
    metadata:
      labels:
        app: eks-neuron-test
        role: master
    spec:
      containers:
        - name: eks-neuron-test
```

```
          image: 763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference-
neuron:1.15.4-neuron-py37-ubuntu18.04
          command:
            - /usr/local/bin/entrypoint.sh
          args:
            - --port=8500
            - --rest_api_port=9000
            - --model_name=resnet50_neuron
            - --model_base_path=s3://${your-bucket-of-models}/resnet50_neuron/
          ports:
            - containerPort: 8500
            - containerPort: 9000
          imagePullPolicy: IfNotPresent
          env:
            - name: AWS_REGION
              value: "us-east-1"
            - name: S3_USE_HTTPS
              value: "1"
            - name: S3_VERIFY_SSL
              value: "0"
            - name: S3_ENDPOINT
              value: s3.us-east-1.amazonaws.com
            - name: AWS_LOG_LEVEL
              value: "3"
          resources:
            limits:
              cpu: 4
              memory: 4Gi
              aws.amazon.com/neuron: 1
            requests:
              cpu: "1"
              memory: 1Gi
          securityContext:
            capabilities:
              add:
                - IPC_LOCK
```

3. Déployez le modèle.

```
kubectl apply -f rn50_deployment.yaml
```

4. Créez un fichier nommé `rn50_service.yaml` avec les contenus suivants. Les ports HTTP et gRPC sont ouverts pour accepter les demandes de prédiction.

```
kind: Service
apiVersion: v1
metadata:
  name: eks-neuron-test
  labels:
    app: eks-neuron-test
spec:
  type: ClusterIP
  ports:
    - name: http-tf-serving
      port: 8500
      targetPort: 8500
    - name: grpc-tf-serving
      port: 9000
      targetPort: 9000
  selector:
    app: eks-neuron-test
    role: master
```

5. Créez un service Kubernetes pour votre application TensorFlow Model Serving.

```
kubectl apply -f rn50_service.yaml
```

(Facultatif) Faites des prédictions par rapport à votre TensorFlow service de service

1. Pour tester localement, transférez le port gRPC au service `eks-neuron-test`.

```
kubectl port-forward service/eks-neuron-test 8500:8500 &
```

2. Créez un script Python appelé `tensorflow-model-server-infer.py` avec le contenu suivant. Ce script exécute l'inférence via gRPC, qui est un framework de service.

```
import numpy as np
   import grpc
   import tensorflow as tf
   from tensorflow.keras.preprocessing import image
   from tensorflow.keras.applications.resnet50 import preprocess_input
   from tensorflow_serving.apis import predict_pb2
   from tensorflow_serving.apis import prediction_service_pb2_grpc
   from tensorflow.keras.applications.resnet50 import decode_predictions
```

```
   if __name__ == '__main__':
       channel = grpc.insecure_channel('localhost:8500')
       stub = prediction_service_pb2_grpc.PredictionServiceStub(channel)
       img_file = tf.keras.utils.get_file(
           "./kitten_small.jpg",
           "https://raw.githubusercontent.com/awslabs/mxnet-model-server/master/docs/
images/kitten_small.jpg")
       img = image.load_img(img_file, target_size=(224, 224))
       img_array = preprocess_input(image.img_to_array(img)[None, ...])
       request = predict_pb2.PredictRequest()
       request.model_spec.name = 'resnet50_inf1'
       request.inputs['input'].CopyFrom(
           tf.make_tensor_proto(img_array, shape=img_array.shape))
       result = stub.Predict(request)
       prediction = tf.make_ndarray(result.outputs['output'])
       print(decode_predictions(prediction))
```

3. Exécutez le script pour soumettre des prédictions à votre service.

```
python3 tensorflow-model-server-infer.py
```

L'exemple qui suit illustre un résultat.

```
[[(u'n02123045', u'tabby', 0.68817204), (u'n02127052', u'lynx', 0.12701613),
 (u'n02123159', u'tiger_cat', 0.08736559), (u'n02124075', u'Egyptian_cat',
 0.063844085), (u'n02128757', u'snow_leopard', 0.009240591)]]
```

Gérez les ressources informatiques pour les AI/ML charges de travail sur Amazon EKS

Cette section est conçue pour vous aider à gérer les ressources de calcul pour les charges de travail de machine learning dans Amazon Elastic Kubernetes Service (EKS). Vous trouverez des informations sur la réservation à GPUs l'aide de blocs de capacité pour les groupes de nœuds gérés et les nœuds autogérés, notamment les conditions préalables, la configuration du modèle de lancement, les configurations de dimensionnement, la préparation de la charge de travail, ainsi que les principales considérations relatives à la gestion des cycles de vie des réservations et à la résiliation progressive des nœuds.

Rubriques

- Création d'un groupe de nœuds géré avec des blocs de capacité pour ML
- Création de nœuds autogérés avec des blocs de capacité pour ML
- Utiliser le P6e- GB2 00 UltraServers avec Amazon EKS

Création d'un groupe de nœuds géré avec des blocs de capacité pour ML

Les blocs de capacité pour le Machine Learning (ML) vous permettent de réserver des instances GPU à une date ultérieure pour prendre en charge vos charges de travail de ML de courte durée. Pour plus d'informations, consultez la section Blocs de capacité pour ML dans le Guide de l'utilisateur Amazon EC2 pour les instances Linux.

Considérations

⚠ Important

- Les blocs de capacité ne sont disponibles que pour certains types d'instances Amazon EC2 et dans certaines régions AWS. Pour plus d'informations sur la compatibilité, consultez la section Prérequis pour l'utilisation des blocs de capacité dans le Guide de l'utilisateur Amazon EC2 pour les instances Linux.
- Pour plus d'informations, consultez la section Utiliser des blocs de capacité pour les charges de travail de machine learning dans le Guide de l'utilisateur Amazon EC2 Auto Scaling.
- Les groupes de nœuds gérés avec des blocs de capacité ne peuvent être créés qu'avec des modèles de lancement personnalisés.
- Lorsque vous mettez à niveau des groupes de nœuds gérés avec des blocs de capacité, assurez-vous que la taille souhaitée du groupe de nœuds est définie sur `0`.

Création d'un groupe de nœuds géré avec des blocs de capacité Amazon EC2

Vous pouvez utiliser des blocs de capacité avec les groupes de nœuds gérés par Amazon EKS afin de provisionner et de mettre à l'échelle des composants master GPU accélérés. Les exemples de modèles AWS CloudFormation ci-dessous ne couvrent pas tous les éléments nécessaires pour un cluster en production. En pratique, vous devrez également inclure un script d'amorçage permettant

au nœud de rejoindre le cluster et spécifier une AMI Amazon EKS accélérée. Pour de plus amples informations, consultez the section called "Créer".

1. Créez un modèle de lancement adapté à vos charges de travail et compatible avec les groupes de nœuds gérés par Amazon EKS. Pour de plus amples informations, consultez the section called "Modèles de lancement".

   En plus des exigences mentionnées dans les procédures ci-dessus, assurez-vous que le `LaunchTemplateData` inclut les éléments suivants :

   - `InstanceMarketOptions` avec `MarketType` défini sur `"capacity-block"`
   - `CapacityReservationSpecification: CapacityReservationTarget` avec `CapacityReservationId` défini sur le bloc de capacité (par exemple : `cr-02168da1478b509e0`)
   - `InstanceType` défini sur un type d'instance qui prend en charge les blocs de capacité (par exemple : *`p5.48xlarge`*)

   L'exemple ci-dessous présente un modèle CloudFormation permettant de créer un modèle de lancement conçu pour utiliser un bloc de capacité. Pour créer un groupe de nœuds gérés avec une AMI personnalisée, vous pouvez également ajouter les paramètres `ImageId` et `UserData`.

   ```
   NodeLaunchTemplate:
     Type: "AWS::EC2::LaunchTemplate"
     Properties:
       LaunchTemplateData:
         InstanceMarketOptions:
           MarketType: "capacity-block"
         CapacityReservationSpecification:
           CapacityReservationTarget:
             CapacityReservationId: "cr-02168da1478b509e0"
         InstanceType: p5.48xlarge
   ```

2. Ensuite, utilisez le modèle de lancement pour créer le groupe de nœuds gérés.

   L'exemple ci-dessous illustre une commande de création d'un groupe de nœuds utilisant des blocs de capacité. Remplacez *`example-values`* par les valeurs propres à votre cluster.

   Lorsque vous créez un groupe de nœuds gérés utilisant des blocs de capacité, procédéz comme suit :

- Définissez `capacity-type` sur `"CAPACITY_BLOCK"`. Si le type de capacité n'est pas défini sur `"CAPACITY_BLOCK"` ou si l'une des valeurs requises ci-dessus dans le modèle de lancement est absente, la demande de création sera rejetée.
- Lorsque vous spécifiez `subnets` dans la demande de création, assurez-vous de n'indiquer que le sous-réseau situé dans la même zone de disponibilité que la réserve de capacité.
- Si vous spécifiez une valeur `desiredSize` non nulle dans la demande de création, Amazon EKS respectera cette valeur lors de la création du groupe Auto Scaling (ASG). Toutefois, si la demande de création est effectuée avant l'activation de la réserve de capacité, l'ASG ne pourra pas lancer d'instances Amazon EC2 tant que la réserve n'est pas active. Par conséquent, les activités de mise à l'échelle de l'ASG entraîneront des erreurs de lancement. Dès que la réservation devient active, le lancement des instances réussira et l'ASG sera mis à l'échelle jusqu'à la valeur `desiredSize` définie lors de la création.

```
aws eks create-nodegroup \
    --cluster-name my-cluster \
    --nodegroup-name my-mng \
    --node-role node-role-arn \
    --region region-code \
    --subnets subnet-id \
    --scaling-config minSize=node-group-min-size,maxSize=node-group-max-
size,desiredSize=node-group-desired-size \
    --ami-type "AL2023_x86_64_NVIDIA" \
    --capacity-type "CAPACITY_BLOCK" \
    --launch-template id="lt-id",version=1
```

3. Assurez-vous que les nœuds rejoignent bien le cluster après la mise à l'échelle pour l'augmentation verticale. Les clusters Amazon EKS utilisant des groupes de nœuds gérés avec blocs de capacité ne vérifient pas que les instances lancées rejoignent et s'enregistrent correctement auprès du cluster.
4. Si vous avez défini `desiredSize` sur `0` au moment de la création, vous disposez de plusieurs options pour mettre à l'échelle le groupe de nœuds en vue de l'augmentation verticale lorsque la réserve de capacité devient active :
   - Créez une stratégie de mise à l'échelle programmée pour l'ASG, alignée sur l'heure de début de la réservation du bloc de capacité. Pour plus d'informations, consultez la section Mise à l'échelle programmée pour Amazon EC2 Auto Scaling dans le Guide de l'utilisateur d'Amazon EC2 Auto Scaling.

- Utiliser la console Amazon EKS ou `eks update-nodegroup-config` pour mettre à jour la configuration d'échelle et définir la taille souhaitée du groupe de nœuds.
- Utiliser l'outil Cluster Autoscaler de Kubernetes. Pour plus d'informations, consultez Cluster Autoscaler sur AWS.

5. Le groupe de nœuds est maintenant prêt à accueillir les charges de travail et à programmer des pods.
6. Afin que vos pods soient correctement vidés avant la fin de la réserve, Amazon EKS utilise une stratégie de mise à l'échelle planifiée pour réduire verticalement la taille du groupe de nœuds à `0`. Cette mise à l'échelle planifiée portera le nom `Amazon EKS Node Group Capacity Scaledown Before Reservation End`. Nous vous recommandons de ne pas modifier ni supprimer cette action.

   Amazon EC2 commence à arrêter les instances 30 minutes avant la fin de la réservation. Par conséquent, Amazon EKS configure une réduction verticale planifiée de la taille du groupe de nœuds 40 minutes avant la fin de la réservation afin de permettre une éviction sécurisée et progressive des pods.

Création de nœuds autogérés avec des blocs de capacité pour ML

Les blocs de capacité pour le Machine Learning (ML) vous permettent de réserver des instances GPU à une date ultérieure pour prendre en charge vos charges de travail de ML de courte durée. Pour plus d'informations, consultez la section Blocs de capacité pour ML dans le Guide de l'utilisateur Amazon EC2 pour les instances Linux.

Considérations

Important

- Les blocs de capacité ne sont disponibles que pour certains types d'instances Amazon EC2 et dans certaines régions AWS. Pour plus d'informations sur la compatibilité, consultez la section Prérequis pour l'utilisation des blocs de capacité dans le Guide de l'utilisateur Amazon EC2 pour les instances Linux.
- Si vous créez un groupe de nœuds autogérés avant que la réserve de capacité de capacité ne devienne active, définissez alors la capacité souhaitée sur `0`.

- Afin de disposer de suffisamment de temps pour vidanger le(s) nœud(s) de manière gracieuse, nous vous conseillons de programmer une mise à l'échelle jusqu'à zéro plus de 30 minutes avant l'heure de fin de la réserve du bloc de capacité.
- Pour permettre une éviction correcte des pods, nous recommandons de configurer le gestionnaire d'arrêt des nœuds AWS Node Termination Handler comme indiqué dans les étapes de l'exemple.

Utilisation des blocs de capacité avec des nœuds autogérés

Vous pouvez utiliser les blocs de capacité avec Amazon EKS pour l'approvisionnement et la mise à l'échelle de vos nœuds autogérés. Les étapes suivantes vous donnent un aperçu général de l'exemple. Les exemples de modèles AWS CloudFormation fournis ne couvrent pas tous les éléments nécessaires pour une charge de travail en production. Il est généralement recommandé d'ajouter un script de démarrage pour joindre le nœud au cluster, de spécifier une AMI Amazon EKS accélérée, ainsi qu'un profil d'instance approprié pour intégrer le nœud au cluster. Pour de plus amples informations, consultez the section called "Amazon Linux".

1. Créez un modèle de lancement applicable à votre charge de travail. Pour plus d'informations, consultez la section Utiliser des blocs de capacité pour les charges de travail de machine learning dans le Guide de l'utilisateur Amazon EC2 Auto Scaling.

   Assurez-vous que les `LaunchTemplateData` incluent les éléments suivants :

   - `InstanceMarketOptions` avec `MarketType` défini sur `"capacity-block"`
   - `CapacityReservationSpecification: CapacityReservationTarget` avec `CapacityReservationId` défini sur le bloc de capacité (par exemple : `cr-`*`02168da1478b509e0`*)
   - `IamInstanceProfile` avec `Arn` défini sur le *`iam-instance-profile-arn`* applicable
   - `ImageId` défini sur l'*`image-id`* applicable
   - `InstanceType` défini sur un type d'instance qui prend en charge les blocs de capacité (par exemple : *`p5.48xlarge`*)
   - `SecurityGroupIds` défini sur les identifiants applicables (par exemple : *`sg-05b1d815d1EXAMPLE`*)
   - `UserData` défini sur les *`user-data`* applicables pour votre groupe de nœuds autogéré

L'exemple ci-dessous présente un modèle CloudFormation permettant de créer un modèle de lancement conçu pour utiliser un bloc de capacité.

```
NodeLaunchTemplate:
  Type: "aws::EC2::LaunchTemplate"
  Properties:
    LaunchTemplateData:
      InstanceMarketOptions:
        MarketType: "capacity-block"
      CapacityReservationSpecification:
        CapacityReservationTarget:
          CapacityReservationId: "cr-02168da1478b509e0"
      IamInstanceProfile:
        Arn: iam-instance-profile-arn
      ImageId: image-id
      InstanceType: p5.48xlarge
      KeyName: key-name
      SecurityGroupIds:
      - sg-05b1d815d1EXAMPLE
      UserData: user-data
```

Vous devez transmettre le sous-réseau de la zone de disponibilité dans laquelle la réservation est effectuée, car les blocs de capacité sont répartis par zone.

2. Utilisez le modèle de lancement pour créer un groupe de nœuds autogérés. Si vous le faites avant que la réserve de capacité ne soit active, définissez la capacité souhaitée sur `0`. Lorsque vous créez le groupe de nœuds, assurez-vous que vous ne spécifiez que le sous-réseau correspondant à la zone de disponibilité dans laquelle la capacité est réservée.

   L'exemple ci-dessous présente un modèle CloudFormation que vous pouvez utiliser comme référence pour créer un modèle adapté à votre charge de travail. Cet exemple récupère le `LaunchTemplateId` et la `Version` de la ressource `AWS::Amazon EC2::LaunchTemplate` indiquée à l'étape précédente. Il obtient également les valeurs pour `DesiredCapacity`, `MaxSize`, `MinSize`, et `VPCZoneIdentifier` qui sont déclarées ailleurs dans le même modèle.

```
NodeGroup:
  Type: "AWS::AutoScaling::AutoScalingGroup"
  Properties:
    DesiredCapacity: !Ref NodeAutoScalingGroupDesiredCapacity
    LaunchTemplate:
      LaunchTemplateId: !Ref NodeLaunchTemplate
```

```
      Version: !GetAtt NodeLaunchTemplate.LatestVersionNumber
    MaxSize: !Ref NodeAutoScalingGroupMaxSize
    MinSize: !Ref NodeAutoScalingGroupMinSize
    VPCZoneIdentifier: !Ref Subnets
    Tags:
      - Key: Name
        PropagateAtLaunch: true
        Value: !Sub ${ClusterName}-${NodeGroupName}-Node
      - Key: !Sub kubernetes.io/cluster/${ClusterName}
        PropagateAtLaunch: true
        Value: owned
```

3. Une fois que le groupe de nœuds a été créé avec succès, assurez-vous d'enregistrer le `NodeInstanceRole` pour le groupe de nœuds qui a été créé. Vous en avez besoin pour vous assurer que, lorsque le groupe de nœuds est mis à l'échelle, les nouveaux nœuds rejoignent bien le cluster et que Kubernetes les reconnaisse. Pour plus d'informations, consultez les instructions AWS Management Console de la section Création de nœuds Amazon Linux autogérés.
4. Nous vous recommandons de créer une politique de mise à l'échelle programmée pour le groupe Auto Scaling qui s'aligne sur les heures de réserve du bloc de capacité. Pour plus d'informations, consultez la section Mise à l'échelle programmée pour Amazon EC2 Auto Scaling dans le Guide de l'utilisateur d'Amazon EC2 Auto Scaling.

   Vous pouvez utiliser toutes les instances que vous avez réservées jusqu'à 30 minutes avant l'heure de fin du bloc de capacité. Les instances encore en cours d'exécution à ce moment-là commenceront à prendre fin. Afin de disposer de suffisamment de temps pour vidanger le(s) nœud(s) de manière gracieuse, nous vous conseillons de programmer une mise à l'échelle jusqu'à zéro plus de 30 minutes avant l'heure de fin de la réserve du bloc de capacité.

   Si, au lieu de cela, vous souhaitez effectuer l'augmentation verticale manuellement la capacité chaque fois que la réserve de capacité devient `Active`, vous devez mettre à jour la capacité souhaitée du groupe Auto Scaling au moment du début de la réserve du bloc de capacité. Ensuite, vous devrez également réduire manuellement la capacité plus de 30 minutes avant l'heure de fin de la réserve du bloc de capacité.
5. Le groupe de nœuds est maintenant prêt à accueillir les charges de travail et à programmer des pods.
6. Pour permettre l'évacuation correcte de vos pods, nous recommandons de configurer le gestionnaire d'arrêt des nœuds AWS Node Termination Handler. Ce gestionnaire surveille les événements de cycle de vie « Réduction ASG » provenant d'Amazon EC2 Auto Scaling via

EventBridge et permet au plan de contrôle Kubernetes de prendre les mesures nécessaires avant que l'instance ne devienne indisponible. Dans le cas contraire, vos pods et objets Kubernetes risquent de rester bloqués à l'état d'attente. Pour plus d'informations, consultez la section AWS Node Termination Handler sur GitHub.

Si vous ne configurez pas de gestionnaire d'arrêt des nœuds Node Termination Handler, nous vous recommandons de commencer à évacuer vos pods manuellement avant la fenêtre des 30 minutes, afin de leur laisser suffisamment de temps pour être évacués correctement.

Utiliser le P6e- GB2 00 UltraServers avec Amazon EKS

Cette rubrique décrit comment configurer et utiliser Amazon EKS avec GB2 P6e-00 UltraServers. Le type d'`p6e-gb200.36xlarge`instance avec 4 NVIDIA Blackwell n' GPUs est disponible qu'en GB2 P6e-00. UltraServers Il existe deux types de GB2 P6e-00 UltraServers. `u-p6e-gb200x36` UltraServer Il a 9 `p6e-gb200.36xlarge` instances et 18 `p6e-gb200.36xlarge` instances. `u-p6e-gb200x72` UltraServer

Pour en savoir plus, consultez la UltraServers page Web Amazon EC2 P6e- GB2 00.

Considérations

- Amazon EKS prend en charge le protocole GB2 P6e-00 UltraServers pour les versions 1.33 et supérieures de Kubernetes. Cette version de Kubernetes prend en charge l'allocation dynamique des ressources (DRA), activée par défaut dans EKS et dans l'accélérateur AL2023 optimisé pour EKS. AMIs Le DRA est obligatoire pour utiliser le GB2 P6e-00 UltraServers avec EKS. Le DRA n'est pas pris en charge en mode Karpenter ou EKS Auto, et il est recommandé d'utiliser des groupes de nœuds autogérés par EKS ou des groupes de nœuds gérés par EKS lorsque vous utilisez le P6e- GB2 00 UltraServers avec EKS.
- Les P6e- GB2 00 UltraServers sont disponibles via EC2 Capacity Blocks for ML. Consultez the section called "Gestion de capacité" pour plus d'informations sur le lancement de nœuds EKS avec des blocs de capacité.
- Lorsque vous utilisez des groupes de nœuds gérés par EKS avec des blocs de capacité, vous devez utiliser des modèles de lancement personnalisés. Lors de la mise à niveau de groupes de nœuds gérés par EKS avec P6e- GB2 00 UltraServers, vous devez définir la taille souhaitée du groupe de nœuds `0` avant la mise à niveau.
- Il est recommandé d'utiliser la variante NVIDIA AL2 023 ARM de l'accélérateur optimisé pour EKS. AMIs Cette AMI inclut les composants de nœud et la configuration requis pour fonctionner avec

GB2 P6e-00 UltraServers. Si vous décidez de créer votre propre AMI, vous êtes responsable de l'installation et de la validation de la compatibilité du nœud et du logiciel système, y compris les pilotes. Pour de plus amples informations, veuillez consulter the section called "Utiliser le GPU EKS Linux AMIs".

- Il est recommandé d'utiliser la version AMI optimisée pour EKS `v20251103` ou une version ultérieure, qui inclut la version 580 du pilote NVIDIA. Cette version du pilote NVIDIA permet à la mémoire CDMM (Coherent Driver-Based Memory Memory) de remédier à un éventuel sursignalement de mémoire. Lorsque CDMM est activé, les fonctionnalités suivantes ne sont pas prises en charge : GPU NVIDIA Multi-Instance (MIG) et vGPU. Pour plus d'informations sur le CDMM, consultez la section Gestion de la mémoire basée sur le pilote NVIDIA Coherent (CDMM).
- Lorsque vous utilisez l'opérateur GPU NVIDIA avec l'AMI AL2 NVIDIA 023 optimisée pour EKS, vous devez désactiver l'installation du pilote et du kit d'outils par l'opérateur, car ceux-ci sont déjà inclus dans l'AMI. Les AL2 023 NVIDIA optimisés pour EKS AMIs n'incluent pas le plug-in de périphérique NVIDIA Kubernetes ni le pilote NVIDIA DRA, et ceux-ci doivent être installés séparément.
- Chaque `p6e-gb200.36xlarge` instance peut être configurée avec un maximum de 17 cartes réseau et peut utiliser l'EFA pour la communication entre les deux UltraServers. Le trafic réseau peut se croiser UltraServers, mais pour des performances optimales, il est recommandé de planifier les charges de travail de la même manière en UltraServer tirant parti de l'IMEX pour les communications intra-GPU. UltraServer Pour plus d'informations, consultez la section Configuration EFA pour les instances P6e- GB2 00.
- Chaque `p6e-gb200.36xlarge` instance dispose de 3 fois 7,5 To de stockage d'instance. Par défaut, l'AMI optimisée pour EKS ne formate ni ne monte les magasins d'instances. Le stockage éphémère du nœud peut être partagé entre les pods qui demandent un stockage éphémère et les images des conteneurs téléchargées sur le nœud. Si vous utilisez l'AMI AL2 023 optimisée pour EKS, celle-ci peut être configurée dans le cadre du bootstrap des nœuds dans les données utilisateur en définissant la politique NodeConfigde stockage local de l'instance sur RAID0. Le réglage sur RAID0 répartit l'instance pour stocker et configurer le runtime du conteneur et le kubelet pour utiliser ce stockage éphémère.

Éléments

Les composants suivants sont recommandés pour exécuter des charges de travail sur EKS avec le GB2 P6e-00. UltraServers Vous pouvez éventuellement utiliser l'opérateur GPU NVIDIA pour installer les composants du nœud NVIDIA. Lorsque vous utilisez l'opérateur GPU NVIDIA avec l'AMI AL2

NVIDIA 023 optimisée pour EKS, vous devez désactiver l'installation du pilote et du kit d'outils par l'opérateur, car ceux-ci sont déjà inclus dans l'AMI.

Pile	Composant
AMI accélérée optimisée pour EKS	Noyau 6.12
	pilote de carte graphique NVIDIA
	Pilote en mode utilisateur NVIDIA CUDA
	Boîte à outils pour conteneurs NVIDIA
	Gestionnaire de tissus NVIDIA
	pilote NVIDIA IMEX
	Gestionnaire de NVLink sous-réseaux NVIDIA
	pilote EFA
Composants exécutés sur le nœud	PVC CNI
	Plug-in pour appareil EFA
	Plug-in pour appareil NVIDIA K8s
	pilote NVIDIA DRA
	Découverte des fonctionnalités des nœuds NVIDIA (NFD)
	Découverte des fonctionnalités du GPU NVIDIA (GFD)

Les composants du nœud présentés dans le tableau ci-dessus exécutent les fonctions suivantes :

- VPC CNI : alloue le VPC IPs comme interface réseau principale pour les pods exécutés sur EKS
- Plug-in d'appareil EFA : alloue des appareils EFA en tant que réseaux secondaires pour les pods fonctionnant sur EKS. Responsable du trafic réseau sur GB2 P6e-00 UltraServers. Pour les charges de travail à nœuds multiples, pour les flux GPU-to-GPU internes et de UltraServer canalisation sur plusieurs nœuds. NVLink

- Plug-in pour appareils NVIDIA Kubernetes : alloue en GPUs tant qu'appareils aux pods exécutés sur EKS. Il est recommandé d'utiliser le plug-in de périphérique NVIDIA Kubernetes jusqu'à ce que la fonctionnalité d'allocation du GPU du pilote NVIDIA DRA soit terminée à la fin de la phase expérimentale. Consultez les versions du pilote NVIDIA DRA pour obtenir des informations actualisées.
- Pilote NVIDIA DRA : active des ressources ComputeDomain personnalisées qui facilitent la création de domaines IMEX qui suivent les charges de travail exécutées sur P6e-00. GB2 UltraServers
  - La ComputeDomain ressource décrit un domaine IMEX (Internode Memory Exchange). Lorsque des charges de travail dotées ResourceClaim de la forme a ComputeDomain sont déployées sur le cluster, le pilote NVIDIA DRA crée automatiquement un IMEX DaemonSet qui s'exécute sur les nœuds correspondants et établit le ou les canaux IMEX entre les nœuds avant le démarrage de la charge de travail. Pour en savoir plus sur l'IMEX, consultez la présentation de NVIDIA IMEX pour les systèmes à nœuds multiples NVLink .
  - Le pilote NVIDIA DRA utilise une étiquette d'identification par clic (`nvidia.com/gpu.clique`) appliquée par NVIDIA GFD qui transmet la connaissance de la topologie et du domaine du réseau. NVLink
  - Il est recommandé de créer une tâche ComputeDomain par charge de travail.
- NVIDIA Node Feature Discovery (NFD) : dépendance requise pour que le GFD applique des étiquettes de nœuds en fonction des attributs découverts au niveau du nœud.
- NVIDIA GPU Feature Discovery (GFD) : applique une étiquette topologique standard NVIDIA appelée `nvidia.com/gpu.clique` aux nœuds. Les nœuds d'un même nœud `nvidia.com/gpu.clique` sont accessibles à plusieurs NVLink nœuds, et vous pouvez utiliser les affinités des modules dans votre application pour planifier des nœuds vers le même domaine. NVlink

Procédure

La section suivante suppose que vous disposez d'un cluster EKS exécutant Kubernetes version 1.33 ou supérieure avec un ou plusieurs groupes de nœuds avec GB2 p6E-00 UltraServers exécutant l'AMI accélérée AL2 023 ARM NVIDIA EKS optimisée pour NVIDIA EKS. Consultez les liens ci-dessous the section called “Gestion de capacité” pour connaître les étapes préalables relatives aux nœuds autogérés et aux groupes de nœuds gérés par EKS.

La procédure suivante utilise les composants ci-dessous.

Name	Version	Description
Opérateur GPU NVIDIA	25,3,4 et plus	Pour la gestion du cycle de vie des plug-ins requis tels que le plug-in pour appareils NVIDIA Kubernetes et le NFD/GFD.
Pilotes NVIDIA DRA	25,8,0 et plus	Pour ComputeDomain CRDs et gestion de domaines IMEX.
Plug-in pour appareil EFA	0,5,14 et plus	Pour la UltraServer communication croisée.

Installer l'opérateur GPU NVIDIA

L'opérateur GPU NVIDIA simplifie la gestion des composants nécessaires à l'utilisation GPUs dans les clusters Kubernetes. Comme le pilote GPU NVIDIA et le kit d'outils de conteneur sont installés dans le cadre de l'AMI accélérée optimisée pour EKS, ils doivent être définis `false` dans la configuration des valeurs Helm.

1. Créez un fichier de valeurs Helm nommé `gpu-operator-values.yaml` avec la configuration suivante.

```
devicePlugin:
  enabled: true
nfd:
  enabled: true
gfd:
  enabled: true
driver:
  enabled: false
toolkit:
  enabled: false
migManager:
  enabled: false
```

2. Installez l'opérateur GPU NVIDIA pour votre cluster à l'aide du `gpu-operator-values.yaml` fichier que vous avez créé à l'étape précédente.

```
helm repo add nvidia https://helm.ngc.nvidia.com/nvidia
```

```
helm repo update
```

```
helm install gpu-operator nvidia/gpu-operator \
 --namespace gpu-operator \
 --create-namespace \
 --version v25.3.4 \
 --values gpu-operator-values.yaml
```

Installez le pilote NVIDIA DRA

À partir de la version opérateur du GPU NVIDIA`v25.3.4`, le pilote NVIDIA DRA doit être installé séparément. Il est recommandé de suivre les notes de mise à jour des opérateurs du GPU NVIDIA, car cela pourrait changer dans une future version.

1. Créez un fichier de valeurs Helm nommé `dra-values.yaml` avec la configuration suivante. Notez le `nodeAffinity` et `tolerations` qui configure le pilote DRA pour qu'il soit déployé uniquement sur des nœuds dotés d'un GPU NVIDIA.

```
resources:
  gpus:
    enabled: false # set to false to disable experimental gpu support
  computeDomains:
    enabled: true

controller:
  nodeSelector: null
  affinity: null
  tolerations: []

kubeletPlugin:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: "nvidia.com/gpu.present"
            operator: In
            values:
            - "true"
  tolerations:
    - key: "nvidia.com/gpu"
```

```
      operator: Exists
      effect: NoSchedule
```

2. Installez le pilote NVIDIA DRA pour votre cluster à l'aide du `dra-values.yaml` fichier que vous avez créé à l'étape précédente.

```
helm repo add nvidia https://helm.ngc.nvidia.com/nvidia
helm repo update
```

```
helm install nvidia-dra-driver-gpu nvidia/nvidia-dra-driver-gpu \
  --version="25.8.0" \
  --namespace nvidia-dra-driver-gpu \
  --create-namespace \
  -f dra-values.yaml
```

3. Après l'installation, le pilote DRA crée `DeviceClass` des ressources qui permettent à Kubernetes de comprendre et d'allouer des `ComputeDomain` ressources, rendant ainsi possible la gestion IMEX pour les charges de travail GPU distribuées sur P6e-00. GB2 UltraServers

   Vérifiez que les ressources DRA sont disponibles à l'aide des commandes suivantes.

```
kubectl api-resources | grep resource.k8s.io
```

```
deviceclasses           resource.k8s.io/v1  false        DeviceClass
resourceclaims          resource.k8s.io/v1  true         ResourceClaim
resourceclaimtemplates  resource.k8s.io/v1  true         ResourceClaimTemplate
resourceslices          resource.k8s.io/v1  false        ResourceSlice
```

```
kubectl get deviceclasses
```

```
NAME
compute-domain-daemon.nvidia.com
compute-domain-default-channel.nvidia.com
```

Installez le plugin pour appareil EFA

Pour utiliser la communication EFA entre eux UltraServers, vous devez installer le plug-in d'appareil Kubernetes pour EFA. Les instances P6e- GB2 00 peuvent être configurées avec un maximum de 17

cartes réseau et le NCI principal (indice 0) doit être de type `interface` et prendre en charge jusqu'à 100 Gbit/s de bande passante ENA. Configurez vos interfaces EFA et ENA selon vos besoins lors du provisionnement des nœuds. Consultez la AWS documentation relative à la configuration EFA pour une instance P6e- GB2 00 pour plus de détails sur la configuration EFA.

1. Créez un fichier de valeurs Helm nommé `efa-values.yaml` avec la configuration suivante.

```
tolerations:
  - key: nvidia.com/gpu
    operator: Exists
    effect: NoSchedule
```

2. Installez l'opérateur NVIDIA DRA pour votre cluster à l'aide du `dra-values.yaml` fichier que vous avez créé à l'étape précédente.

```
helm repo add eks https://aws.github.io/eks-charts
helm repo update
```

```
helm install efa eks/aws-efa-k8s-device-plugin -n kube-system \
  --version="0.5.14" \
  -f efa-values.yaml
```

Par exemple, si vous avez configuré vos instances avec une interface EFA uniquement dans chaque groupe NCI, lors de la description d'un nœud, il est prévu de voir 4 périphériques EFA allouables par nœud.

```
kubectl describe node/<gb200-node-name>
```

```
Capacity:
  ...
  vpc.amazonaws.com/efa:  4
Allocatable:
  ...
  vpc.amazonaws.com/efa:  4
```

Valider l'IMEX sur plusieurs nœuds NVLink

Pour un test NCCL NVLINK multi-nœuds et d'autres microbenchmarks, consultez le référentiel. awesome-distributed-training GitHub Les étapes suivantes montrent comment exécuter un NVLink test multi-nœuds avec nvbandwidth.

1. Pour exécuter un test de bande passante multi-nœuds sur deux nœuds du NVL72 domaine, installez d'abord l'opérateur MPI :

```
kubectl create -f https://github.com/kubeflow/mpi-operator/releases/download/v0.7.0/
mpi-operator.yaml
```

2. Créez un fichier de valeurs Helm nommé `nvbandwidth-test-job.yaml` qui définit le manifeste de test. Notez l'affinité du `nvidia.com/gpu.clique` pod pour planifier les travailleurs dans le même NVLink domaine accessible à plusieurs nœuds NVLink .

   À partir de la version du pilote NVIDIA DRA, `v25.8.0` ComputeDomains elles sont élastiques et `.spec.numNodes` peuvent être définies `0` dans la ComputeDomain définition. Consultez les dernières notes de mise à jour du pilote NVIDIA DRA.

```
---
apiVersion: resource.nvidia.com/v1beta1
kind: ComputeDomain
metadata:
  name: nvbandwidth-test-compute-domain
spec:
  numNodes: 0 # This can be set to 0 from NVIDIA DRA Driver version v25.8.0+
  channel:
    resourceClaimTemplate:
      name: nvbandwidth-test-compute-domain-channel

---
apiVersion: kubeflow.org/v2beta1
kind: MPIJob
metadata:
  name: nvbandwidth-test
spec:
  slotsPerWorker: 4 # 4 GPUs per worker node
  launcherCreationPolicy: WaitForWorkersReady
  runPolicy:
    cleanPodPolicy: Running
  sshAuthMountPath: /home/mpiuser/.ssh
```

```
  mpiReplicaSpecs:
    Launcher:
      replicas: 1
      template:
        metadata:
          labels:
            nvbandwidth-test-replica: mpi-launcher
        spec:
          containers:
          - image: ghcr.io/nvidia/k8s-samples:nvbandwidth-v0.7-8d103163
            name: mpi-launcher
            securityContext:
              runAsUser: 1000
            command:
            - mpirun
            args:
            - --bind-to
            - core
            - --map-by
            - ppr:4:node
            - -np
            - "8"
            - --report-bindings
            - -q
            - nvbandwidth
            - -t
            - multinode_device_to_device_memcpy_read_ce
    Worker:
      replicas: 2 # 2 worker nodes
      template:
        metadata:
          labels:
            nvbandwidth-test-replica: mpi-worker
        spec:
          affinity:
            podAffinity:
              requiredDuringSchedulingIgnoredDuringExecution:
              - labelSelector:
                  matchExpressions:
                  - key: nvbandwidth-test-replica
                    operator: In
                    values:
                    - mpi-worker
                topologyKey: nvidia.com/gpu.clique
```

```
          containers:
          - image: ghcr.io/nvidia/k8s-samples:nvbandwidth-v0.7-8d103163
            name: mpi-worker
            securityContext:
              runAsUser: 1000
            env:
            command:
            - /usr/sbin/sshd
            args:
            - -De
            - -f
            - /home/mpiuser/.sshd_config
            resources:
              limits:
                nvidia.com/gpu: 4  # Request 4 GPUs per worker
              claims:
              - name: compute-domain-channel # Link to IMEX channel
          resourceClaims:
          - name: compute-domain-channel
            resourceClaimTemplateName: nvbandwidth-test-compute-domain-channel
```

3. Créez le ComputeDomain et lancez-le avec la commande suivante.

```
kubectl apply -f nvbandwidth-test-job.yaml
```

4. ComputeDomain création, vous pouvez voir que la charge de travail ComputeDomain comporte deux nœuds :

```
kubectl get computedomains.resource.nvidia.com -o yaml
```

```
status:
  nodes:
  - cliqueID: <ClusterUUID>.<Clique ID>
    ipAddress: <node-ip>
    name: <node-hostname>
  - cliqueID: <ClusterUUID>.<Clique ID>
    ipAddress: <node-ip>
    name: <node-hostname>
  status: Ready
```

5. Passez en revue les résultats de la tâche à l'aide de la commande suivante.

```
kubectl logs --tail=-1 -l job-name=nvbandwidth-test-launcher
```

6. Lorsque le test est terminé, supprimez-le à l'aide de la commande suivante.

```
kubectl delete -f nvbandwidth-test-job.yaml
```

Recettes pour optimiser votre cluster Amazon EKS en fonction des charges de AI/ML travail

> **Tip**
>
> Inscrivez-vous aux prochains AI/ML ateliers Amazon EKS.

Cette section est conçue pour fournir de petites recettes pour optimiser votre cluster Amazon EKS, en particulier pour les AI/ML charges de travail impliquant du matériel spécialisé. Vous trouverez des conseils pour empêcher la planification des pods sur des nœuds spécifiques en alourdissant les groupes de nœuds gérés, notamment les conditions préalables, les step-by-step procédures et les considérations relatives au déploiement.

Rubriques

- Recette : empêcher la planification de pods sur des nœuds spécifiques

Recette : empêcher la planification de pods sur des nœuds spécifiques

Présentation

Les nœuds dotés de processeurs spécialisés, tels que les GPU, peuvent être plus coûteux à exploiter que les nœuds sur des machines standard. Pour protéger ces nœuds des charges de travail qui ne nécessitent pas de matériel spécial, vous pouvez utiliser les rejets Kubernetes. Les rejets marquent les nœuds afin de repousser les pods qui ne disposent pas de tolérances correspondantes, garantissant ainsi que seules les charges de travail compatibles sont planifiées. Pour plus d'informations, consultez Rejets et tolérances dans la documentation de Kubernetes.

Les taints de nœuds Kubernetes peuvent être appliqués à des groupes de nœuds gérés, nouveaux ou existants, à l'aide de AWS Management Console ou de l'API Amazon EKS. Cette recette montre

comment appliquer des rejets à des groupes de nœuds gérés Amazon EKS à l'aide de l'AWS CLI. Pour plus d'informations sur la création d'un groupe de nœuds avec un rejet à l'aide de la AWS Management Console, consultez the section called "Créer".

Prérequis

- Un cluster Amazon EKS existant.
- AWS CLI installé et configuré avec les autorisations appropriées.

Étapes

Étape 1 : créer un groupe de nœuds avec des rejets

Utilisez la commande `aws eks create-nodegroup` pour créer un nouveau groupe de nœuds gérés avec des rejets. Cet exemple applique un rejet avec la clé `dedicated`, la valeur `gpuGroup` et l'effet `NO_SCHEDULE`.

```
aws eks create-nodegroup \
 --cli-input-json '
{
  "clusterName": "my-cluster",
  "nodegroupName": "node-taints-example",
  "subnets": [
     "subnet-1234567890abcdef0",
     "subnet-abcdef01234567890",
     "subnet-021345abcdef67890"
   ],
  "nodeRole": "arn:aws:iam::111122223333:role/AmazonEKSNodeRole",
  "taints": [
     {
         "key": "dedicated",
         "value": "gpuGroup",
         "effect": "NO_SCHEDULE"
     }
   ]
}'
```

Pour plus d'informations et d'exemples, consultez rejet dans la documentation de référence Kubernetes.

Étape 2 : mettre à jour les rejets sur un groupe de nœuds existant

Utilisez la commande AWS CLI aws eks update-nodegroup-config pour ajouter, supprimer ou remplacer des rejets pour les groupes de nœuds gérés.

```
aws eks update-nodegroup-config
  --cluster-name my-cluster
  --nodegroup-name node-taints-example
  --taints 'removeTaints=[{key=dedicated,value=gpuGroup,effect=NO_SCHEDULE}]'
```

Remarques

- Les rejets peuvent être mis à jour après avoir créé le groupe de nœuds à l'aide de l'API `UpdateNodegroupConfig`.
- La clé de rejet doit commencer par une lettre ou un chiffre. Il peut contenir des lettres, des chiffres, des traits d'union (`-`), des points (`.`) et des traits de soulignement (`_`). Il peut comporter jusqu'à 63 caractères.
- Éventuellement, la clé de rejet peut commencer par un préfixe de sous-domaine DNS et un simple `/`. Si elle commence par un préfixe de sous-domaine DNS, elle peut comporter 253 caractères.
- La valeur est facultative et doit commencer par une lettre ou un chiffre. Il peut contenir des lettres, des chiffres, des traits d'union (`-`), des points (`.`) et des traits de soulignement (`_`). Il peut comporter jusqu'à 63 caractères.
- Lorsque vous utilisez directement Kubernetes ou la AWS Management Console, l'effet du rejet doit être `NoSchedule`, `PreferNoSchedule` ou `NoExecute`. Cependant, lorsque vous utilisez l'AWS CLI ou l'API, l'effet de rejet doit être `NO_SCHEDULE`, `PREFER_NO_SCHEDULE` ou `NO_EXECUTE`.
- Un maximum de 50 rejets est autorisé pour un groupe de nœuds.
- Si les rejets créés à l'aide d'un groupe de nœuds gérés sont supprimés manuellement d'un nœud, Amazon EKS ne les ajoute pas à nouveau au nœud. Ceci est vrai même si les rejets sont spécifiés dans la configuration du groupe de nœuds gérés.

Ressources pour démarrer avec l'IA/ML sur Amazon EKS

Pour vous lancer dans le machine learning sur EKS, commencez par choisir parmi ces modèles prescriptifs afin de préparer rapidement un cluster EKS et le logiciel et le matériel ML nécessaires pour commencer à exécuter des charges de travail ML.

Ateliers

Atelier sur l'IA générative sur Amazon EKS

Découvrez comment vous lancer avec les applications et l'inférence de grands modèles de langage (LLM) sur Amazon EKS. Découvrez comment déployer et gérer des charges de travail LLM de niveau production. Grâce à des ateliers pratiques, vous explorerez comment tirer parti d'Amazon EKS ainsi que des services AWS et des outils open source pour créer des solutions LLM robustes. L'environnement de l'atelier fournit toute l'infrastructure et tous les outils nécessaires, vous permettant de vous concentrer sur l'apprentissage et la mise en œuvre.

IA générative sur Amazon EKS à l'aide de Neuron

Découvrez comment vous lancer avec les applications et l'inférence de grands modèles de langage (LLM) sur Amazon EKS. Découvrez comment déployer et gérer des charges de travail LLM de niveau production, mettre en œuvre des modèles RAG avancés avec des bases de données vectorielles et créer des applications LLM basées sur des données à l'aide de cadres open source. Grâce à des ateliers pratiques, vous explorerez comment tirer parti d'Amazon EKS ainsi que des services AWS et des outils open source pour créer des solutions LLM robustes. L'environnement de l'atelier fournit toute l'infrastructure et tous les outils nécessaires, vous permettant de vous concentrer sur l'apprentissage et la mise en œuvre.

Bonnes pratiques

Les rubriques consacrées à l'IA/ML dans le guide des bonnes pratiques Amazon EKS fournissent des recommandations détaillées dans les domaines suivants afin d'optimiser vos charges de travail IA/ML sur Amazon EKS.

Calcul et auto-scaling IA/ML

Cette section présente les bonnes pratiques pour optimiser le calcul et l'auto-scaling IA/ML dans Amazon EKS, en mettant l'accent sur la gestion des ressources GPU, la résilience des nœuds et le scaling des applications. Elle fournit des stratégies telles que la planification des charges de travail avec des étiquettes bien connues et l'affinité des nœuds, l'utilisation de blocs de capacité ML ou de réserves de capacité à la demande, et la mise en œuvre de la surveillance de l'état des nœuds avec des outils tels que EKS Node Monitoring Agent.

Mise en réseau IA/ML

Cette section présente les bonnes pratiques pour optimiser la mise en réseau AI/ML dans Amazon EKS afin d'améliorer les performances et la capacité de mise à l'échelle, notamment des stratégies telles que la sélection d'instances avec une bande passante du réseau plus élevée ou Elastic Fabric Adapter (EFA) pour l'entraînement distribué, l'installation d'outils tels que MPI et NCCL, et l'activation de la délégation de préfixe pour augmenter les adresses IP et améliorer les temps de lancement des pods.

Sécurité IA/ML

Cette section se concentre sur la sécurisation du stockage de données et la garantie de la conformité des charges de travail IA/ML sur Amazon EKS, notamment grâce à des pratiques telles que l'utilisation d'Amazon S3 avec AWS Key Management Service (KMS) pour le chiffrement côté serveur (SSE-KMS), la configuration de compartiments avec des clés KMS régionales et des clés de compartiment S3 pour réduire les coûts, l'octroi d'autorisations IAM pour les actions KMS telles que le déchiffrement aux pods EKS, et l'audit avec les journaux AWS CloudTrail.

Stockage IA/ML

Cette section présente les bonnes pratiques pour optimiser le stockage dans les charges de travail IA/ML sur Amazon EKS, notamment le déploiement de modèles à l'aide de pilotes CSI pour monter des services tels que S3, Amazon FSx pour Lustre ou EFS en tant que volumes persistants, la sélection du stockage en fonction des besoins de la charge de travail (par exemple, Amazon FSx pour Lustre pour la formation distribuée avec des options telles que Scratch-SSD ou Persistent-SSD) et l'activation de fonctionnalités telles que la compression et l'agrégation par bandes des données.

Observabilité IA/ML

Cette section se concentre sur la surveillance et l'optimisation de l'utilisation des GPU pour les charges de travail IA/ML sur Amazon EKS afin d'améliorer l'efficacité et de réduire les coûts, notamment grâce à des stratégies telles que le ciblage d'une utilisation élevée des GPU à l'aide d'outils tels que CloudWatch Container Insights et DCGM-Exporter de NVIDIA intégré à Prometheus et Grafana, ainsi que les métriques que nous vous recommandons d'analyser pour vos charges de travail IA/ML.

Performances IA/ML

Cette section se concentre sur l'amélioration de la mise à l'échelle et des performances des applications pour les charges de travail IA/ML sur Amazon EKS grâce à la gestion des images de

conteneurs et à l'optimisation du démarrage, notamment des pratiques telles que l'utilisation de petites images de base légères ou de conteneurs AWS Deep Learning Containers avec des builds en plusieurs étapes, le préchargement d'images via des instantanés EBS ou le préchargement dans le cache d'exécution à l'aide de DaemonSets ou de Deployments.

Architectures de référence

Veuillez explorer ces référentiels GitHub pour obtenir des architectures de référence, des exemples de code et des utilitaires permettant de mettre en œuvre l'entraînement et l'inférence distribuées pour les charges de travail IA/ML sur Amazon EKS et d'autres services AWS.

Entraînement distribué AWSome

Ce référentiel propose une collection de bonnes pratiques, d'architectures de référence, d'exemples d'entraînement des modèles et d'utilitaires pour entraîner de grands modèles sur AWS. Il prend en charge l'entraînement distribué avec Amazon EKS, y compris les modèles CloudFormation pour les clusters EKS, les constructions AMI et de conteneurs personnalisées, les cas de test pour des cadres tels que PyTorch (DDP/FSDP, MegatronLM, NeMo) et JAX, ainsi que des outils de validation, d'observabilité et de surveillance des performances tels que EFA Prometheus exporter et Nvidia Nsight Systems.

Inférence AWSome

Ce référentiel fournit des architectures de référence et des cas de test pour optimiser les solutions d'inférence sur AWS, en mettant l'accent sur Amazon EKS et les instances EC2 accélérées. Il comprend des configurations d'infrastructure pour les clusters VPC et EKS, des projets pour des cadres tels que NVIDIA NIMs, TensorRT-LLM, Triton Inference Server et RayService, avec des exemples pour des modèles tels que Llama3-8B et Llama 3.1 405B. Il propose des déploiements multi-nœuds utilisant K8s LeaderWorkerSet, l'autoscaling EKS, les GPU multi-instances (MIG) et des cas d'utilisation réels tels qu'un bot audio pour l'ASR, l'inférence et le TTS.

Didacticiels

Si vous souhaitez configurer des plateformes et des cadres de machine learning dans Amazon EKS, consultez les tutoriels décrits dans cette section. Ces tutoriels couvrent tous les aspects, des modèles permettant d'optimiser l'utilisation des processeurs GPU au choix des outils de modélisation, en passant par créer des cadres pour des secteurs spécialisés.

Créer des plateformes d'IA générative sur Amazon EKS

- Déployez des modèles d'IA générative sur Amazon EKS
- Création de plateformes JupyterHub multi-locataires sur Amazon EKS

Exécuter des cadres d'IA générative spécialisés sur Amazon EKS

- Accélérez vos charges de travail d'entraînement distribué d'IA générative avec le cadre NVIDIA NeMo sur Amazon EKS
- Exécution de TorchServe sur Amazon Elastic Kubernetes Service

Optimiser les performances des GPU NVIDIA pour le ML sur Amazon EKS

- Mettez en œuvre le partage de GPU afin d'utiliser efficacement les GPU NVIDIA pour vos clusters Amazon EKS :

  Partage de GPU sur Amazon EKS avec le time-slicing NVIDIA et les instances EC2 accélérées
- Utilisez les GPU multi-instances (MIG) et les microservices NIM pour exécuter davantage de pods par GPU sur vos clusters EKS :

  Optimisation de l'utilisation des GPU avec le GPU multi-instances (MIG) de NVIDIA sur Amazon EKS : exécution d'un plus grand nombre de pods par GPU pour des performances améliorées
- Créez et déployez un système machine learning évolutif sur Kubernetes avec Kubeflow on AWS

Exécuter des charges de travail d'encodage vidéo sur EKS

- Diffusion de contenu vidéo avec des GPU fractionnés dans des conteneurs sur Amazon EKS

Accélérer le chargement des images pour les charges de travail d'inférence

- Comment H2O.ai a optimisé et sécurisé son infrastructure d'IA/ML avec Karpenter et Bottlerocket

Surveillance des charges de travail ML

- Surveillance des charges de travail GPU sur Amazon EKS à l'aide des services open source gérés par AWS

- Activez les métriques GPU basées sur les pods dans Amazon CloudWatch

Outils Amazon EKS

Amazon EKS fournit une suite d' AWS outils qui vous aident à gérer, à dépanner et à interagir avec vos clusters EKS de manière plus efficace. Ces outils tirent parti des fonctionnalités basées sur l'IA et des interfaces standardisées pour simplifier les opérations des clusters, accélérer le dépannage et permettre des interactions en langage naturel avec vos ressources Kubernetes. Les outils disponibles incluent :

- Serveur Amazon EKS Model Context Protocol (MCP) : service entièrement géré qui permet des expériences basées sur l'IA pour la gestion des clusters EKS. Le serveur EKS MCP s'intègre aux assistants de codage AI compatibles avec MCP pour fournir une connaissance contextuelle en temps réel de vos clusters et de vos ressources Kubernetes. Utilisez-le pour gérer des clusters, déployer des applications, résoudre des problèmes et interroger la documentation EKS par le biais d'interactions en langage naturel.
- Amazon Q sur la console Amazon EKS : une intégration de résolution des problèmes basée sur l'IA, disponible directement dans la console de AWS gestion. Amazon Q analyse automatiquement les problèmes liés aux clusters et fournit les résultats des enquêtes, une analyse des causes premières et des mesures d'atténuation suggérées lorsque vous rencontrez des erreurs ou des avertissements dans la console EKS.

Ces outils fonctionnent ensemble pour fournir un support complet tout au long du cycle de vie de votre cluster EKS, depuis l'installation et la configuration initiales jusqu'aux opérations continues et au dépannage. Que vous développiez des applications, gériez une infrastructure ou résolviez des problèmes de production, ces AWS outils vous aident à travailler plus efficacement avec vos clusters EKS.

Rubriques

- Serveur Amazon EKS Model Context Protocol (MCP)
- Utiliser Amazon Q Developer sur la console Amazon EKS

Serveur Amazon EKS Model Context Protocol (MCP)

Le serveur Amazon EKS MCP est un service entièrement géré permettant des expériences basées sur l'IA pour le développement et les opérations. Le Model Context Protocol (MCP) fournit une interface standardisée qui enrichit les agents et applications d'IA grâce à une connaissance

contextuelle en temps réel de vos clusters EKS et de vos ressources Kubernetes, permettant ainsi des réponses plus précises et sensibles au contexte ainsi que des flux de travail basés sur l'IA tout au long du cycle de vie des applications, de la configuration initiale à l'optimisation de la production et au dépannage.

> **Note**
>
> Le serveur Amazon EKS MCP est en version préliminaire pour Amazon EKS et est sujet à modification.

Présentation de

Le serveur EKS MCP peut être facilement intégré à n'importe quel assistant de codage AI compatible MCP pour améliorer votre flux de travail de développement comme Kiro,Amazon Q Developer CLI ou des outils tiers tels que Cursor ou Cline. Au démarrage, le serveur EKS MCP vous guide tout au long de la création du cluster, du provisionnement automatique des prérequis et de l'application AWS des meilleures pratiques. Au cours du développement, il simplifie les opérations EKS et Kubernetes en fournissant des flux de travail de haut niveau pour le déploiement des applications et la gestion des clusters. Pour le débogage et le dépannage, le serveur accélère la résolution des problèmes grâce à des aides au dépannage intégrées et à l'accès à la base de connaissances disponibles sur la console EKS et sur vos assistants IA préférés. Ces fonctionnalités sont accessibles via des interactions en langage naturel, ce qui vous permet d'effectuer des opérations Kubernetes complexes de manière plus intuitive et plus efficace.

Le serveur EKS MCP entièrement géré est hébergé dans le AWS cloud, ce qui élimine le besoin d'installation et de maintenance locales. Il fournit des fonctionnalités professionnelles telles que des mises à jour et des correctifs automatiques, une sécurité centralisée grâce à l'intégration AWS IAM, une journalisation complète des audits via, ainsi que l'évolutivité AWS CloudTrail, la fiabilité et le support éprouvés de. AWS Le serveur EKS MCP entièrement géré et hébergé dans le AWS cloud offre les principaux avantages suivants :

- Éliminez l'installation et la maintenance. Le serveur EKS MCP étant hébergé dans le AWS cloud, vous n'avez plus besoin de gérer les mises à jour des versions ni de résoudre les problèmes liés au serveur local. Configurez simplement votre assistant AI pour qu'il se connecte au point de terminaison du serveur EKS MCP hébergé, et vous êtes prêt à commencer à travailler avec vos clusters EKS.

- Gestion centralisée des accès. Le serveur EKS MCP s'intègre à IAM, fournissant un moyen centralisé et sécurisé de contrôler l'accès au serveur. Toutes les demandes sont signées à l'aide de AWS SigV4 via un proxy léger, ce qui permet une intégration fluide avec vos AWS informations d'identification et vos politiques IAM existantes.
- Surveillance et visibilité améliorées. AWS CloudTrail l'intégration capture les appels à l'outil d'initialisation et d'accès complet effectués via le service hébergé, permettant ainsi des pistes d'audit détaillées et des rapports de conformité.
- Toujours up-to-date. Recevez automatiquement de nouveaux outils, fonctionnalités et corrections de bogues sans avoir à mettre à jour les installations locales. Le service hébergé est continuellement amélioré en fonction de vos commentaires et des AWS meilleures pratiques.

Exemples d'intégration

Le serveur EKS MCP fournit plusieurs outils que vous pouvez utiliser pour :

- Gérez votre cluster Créez, configurez et gérez des clusters EKS selon les meilleures pratiques automatisées.
- Gérez les ressources Kubernetes Déployez des applications, gérez les ressources Kuberentes et inspectez l'état du cluster.
- Résoudre les problèmes de votre cluster Diagnostiquez les problèmes à l'aide des outils de dépannage intégrés et de la base de connaissances des runbooks
- Consultez la documentation Recherchez et récupérez la documentation EKS pertinente de manière contextuelle.

Explorez vos clusters

```
Show me all EKS clusters and their status
What insights does EKS have about my production-cluster?
Show me the VPC configuration for my staging cluster
```

Consultez les ressources Kubernetes

```
List all deployments in the production namespace
Show me pods that are not in Running state
Get the logs from the api-server pod in the last 30 minutes
```

Résolution des problèmes

```
Why is my nginx-ingress-controller pod failing to start?
Search the EKS troubleshooting guide for pod networking issues
Show me events related to the failed deployment in the staging namespace
```

Création de ressources (si le mode « écriture » est activé)

```
Create a new EKS cluster named demo-cluster with VPC and Auto Mode
Deploy my containerized app from ECR to the production namespace with 3 replicas
Generate a Kubernetes deployment YAML for my Node.js app running on port 3000
```

Mise en route

Consultez the section called “Mise en route” pour démarrer.

Commencer à utiliser le serveur Amazon EKS MCP

Ce guide explique les étapes de configuration et d'utilisation du serveur EKS MCP avec vos assistants de code AI. Vous apprendrez à configurer votre environnement, à vous connecter au serveur et à commencer à gérer vos clusters EKS par le biais d'interactions en langage naturel.

> **Note**
>
> Le serveur Amazon EKS MCP est en version préliminaire pour Amazon EKS et est sujet à modification.

Conditions préalables

Avant de commencer, assurez-vous d'avoir effectué les tâches suivantes :

- Créez un AWS compte avec accès à Amazon EKS
- Installation et configuration de la AWS CLI avec des informations d'identification
- Installez Python 3.10+
- Installer uv

Configuration

1. Vérifiez les conditions préalables

```
# Check that your Python version is 3.10 or higher
python3 --version

# Check uv installation
uv --version

# Verify CLI configuration
aws configure list
```

2. Configuration des autorisations IAM

Pour vous connecter au serveur EKS MCP, votre rôle IAM doit être associé aux politiques suivantes : **`eks-mcp:InvokeMcp`**(autorisations requises pour l'initialisation et la récupération d'informations sur les outils disponibles), **`eks-mcp:CallReadOnlyTool`**(autorisations requises pour l'utilisation des outils en lecture seule) et **`eks-mcp:CallPrivilegedTool`**(autorisations requises pour l'utilisation des outils d'accès complet (écriture)). Ces `eks-mcp` autorisations sont incluses dans les politiques AWS gérées en lecture seule et en accès complet fournies ci-dessous.

- Ouvrez la console IAM.
- Dans le volet de navigation de gauche, choisissez Utilisateurs, Groupes d'utilisateurs ou Rôles en fonction de l'identité à laquelle vous souhaitez associer la politique, puis du nom de l'utilisateur, du groupe ou du rôle spécifique.
- Sélectionnez l'onglet Autorisations.
- Choisissez Joindre des politiques (ou Ajouter des autorisations si c'est la première fois).
- Dans la liste des politiques, recherchez et sélectionnez la stratégie gérée que vous souhaitez associer :
- Opérations en lecture seule : Amazon EKSMCPRead OnlyAccess
- Choisissez Joindre des politiques (ou Suivant, puis Ajouter des autorisations pour confirmer).

Cela joint la politique et les autorisations prennent effet immédiatement. Vous pouvez associer plusieurs politiques à la même identité, chacune pouvant contenir différentes autorisations. Pour en savoir plus sur ces politiques, consultez les politiques AWS gérées pour Amazon Elastic Kubernetes Service.

3. Choisissez un assistant IA

Choisissez l'un des assistants IA compatibles MCP suivants ou n'importe quel outil compatible MCP :

- Installation de la CLI Amazon Q Developer
- Installez Kiro
- Installer Cursor
- Installez l'extension de code Cline VS

Étape 1 : configurer votre assistant AI

Choisissez l'une des options suivantes pour configurer votre assistant de code AI. L'exécution de cette étape configure votre assistant de code AI pour qu'il utilise le proxy MCP pour AWS, qui est requis pour un accès sécurisé et authentifié au serveur MCP Amazon EKS. Cela implique d'ajouter ou de modifier le fichier de configuration MCP (par exemple, `~/.aws/amazonq/mcp.json` pour Amazon Q Developer CLI). Le proxy agit comme un pont côté client, gérant l'authentification AWS SigV4 à l'aide de vos informations d' AWS identification locales et permettant la découverte dynamique d'outils pour interagir avec les serveurs MCP principaux tels que le serveur AWS MCP EKS. Pour en savoir plus, consultez le proxy MCP pour le AWS référentiel.

Option A : Amazon Q Developer CLI

La CLI Q Developer fournit l'expérience la plus intégrée possible avec le serveur EKS MCP.

1. Localiser le fichier de configuration MCP

- Mac OS/Linux : `~/.aws/q/mcp.json`
- Windows : `%USERPROFILE%\.aws\q\mcp.json`

2. Ajouter une configuration de serveur MCP

Créez le fichier de configuration s'il n'existe pas. Assurez-vous de remplacer l'espace réservé à la région (`{region}`) par la région de votre choix.

Pour Mac/Linux :

```
{
  "mcpServers": {
    "eks-mcp": {
```

```
      "disabled": false,
      "type": "stdio",
      "command": "uvx",
      "args": [
        "mcp-proxy-for-aws@latest",
        "https://eks-mcp.{region}.api.aws/mcp",
        "--service",
        "eks-mcp",
        "--profile",
        "default",
        "--region",
        "{region}"
      ]
    }
  }
}
```

Pour Windows:

```
{
  "mcpServers": {
    "eks-mcp": {
      "disabled": false,
      "type": "stdio",
      "command": "uvx",
      "args": [
        "--from",
        "mcp-proxy-for-aws@latest",
        "mcp-proxy-for-aws.exe",
        "https://eks-mcp.{region}.api.aws/mcp",
        "--service",
        "eks-mcp",
        "--profile",
        "default",
        "--region",
        "{region}"
      ]
    }
  }
}
```

Remarque de sécurité : `--read-only` peut être utilisé pour autoriser uniquement les opérations d'outil en lecture seule.

3. Vérifier la configuration

Redémarrez Q Developer CLI, puis vérifiez les outils disponibles :

```
q /tools
```

Option B : Kiro IDE

Kiro est un espace de travail de codage axé sur l'IA avec support MCP intégré.

1. Ouvrez les paramètres Kiro

- Ouvrez Kiro
- Accédez à Kiro → Paramètres et recherchez « MCP Config »
- Ou appuyez sur `Cmd+Shift+P,` (Mac) ou `Ctrl+Shift+P,` (Windows/Linux) et recherchez « MCP Config »

2. Ajouter une configuration de serveur MCP

- Cliquez sur « Open Workspace MCP Config » ou « Open User MCP Config » pour modifier directement le fichier de configuration MCP.

Assurez-vous de remplacer l'espace réservé à la région (`{region}`) par la région de votre choix.

Pour Mac/Linux :

```
{
  "mcpServers": {
    "eks-mcp": {
      "disabled": false,
      "type": "stdio",
      "command": "uvx",
      "args": [
        "mcp-proxy-for-aws@latest",
        "https://eks-mcp.{region}.api.aws/mcp",
        "--service",
        "eks-mcp",
        "--profile",
        "default",
        "--region",
```

```
        "{region}"
      ]
    }
  }
}
```

Pour Windows:

```
{
  "mcpServers": {
    "eks-mcp": {
      "disabled": false,
      "type": "stdio",
      "command": "uvx",
      "args": [
        "--from",
        "mcp-proxy-for-aws@latest",
        "mcp-proxy-for-aws.exe",
        "https://eks-mcp.{region}.api.aws/mcp",
        "--service",
        "eks-mcp",
        "--profile",
        "default",
        "--region",
        "{region}"
      ]
    }
  }
}
```

Remarque de sécurité : `--read-only` peut être utilisé pour autoriser uniquement les opérations d'outil en lecture seule.

Option C : IDE du curseur

Cursor fournit un support MCP intégré avec une interface de configuration graphique.

1. Ouvrir les paramètres du curseur

- Ouvrir le curseur
- Accédez à Paramètres → Paramètres du curseur → Outils et MCP
- Ou appuyez sur `Cmd+Shift+P` (Mac)/`Ctrl+Shift+P`(Windows) et recherchez « MCP »

2. Ajouter une configuration de serveur MCP

- Cliquez sur « Nouveau serveur MCP »

Créez le fichier de configuration s'il n'existe pas. Assurez-vous de remplacer l'espace réservé à la région (`{region}`) par la région de votre choix.

Pour Mac/Linux :

```
{
  "mcpServers": {
    "eks-mcp": {
      "disabled": false,
      "type": "stdio",
      "command": "uvx",
      "args": [
        "mcp-proxy-for-aws@latest",
        "https://eks-mcp.{region}.api.aws/mcp",
        "--service",
        "eks-mcp",
        "--profile",
        "default",
        "--region",
        "{region}"
      ]
    }
  }
}
```

Pour Windows:

```
{
  "mcpServers": {
    "eks-mcp": {
      "disabled": false,
      "type": "stdio",
      "command": "uvx",
      "args": [
        "--from",
        "mcp-proxy-for-aws@latest",
        "mcp-proxy-for-aws.exe",
        "https://eks-mcp.{region}.api.aws/mcp",
```

```
        "--service",
        "eks-mcp",
        "--profile",
        "default",
        "--region",
        "{region}"
      ]
    }
  }
}
```

Remarque de sécurité : `--read-only` peut être utilisé pour autoriser uniquement les opérations d'outil en lecture seule.

3. Redémarrer le curseur

Fermez puis rouvrez le curseur pour que les modifications prennent effet.

4. Vérifier dans le chat Cursor

Ouvrez le panneau de discussion et essayez :

```
What EKS MCP tools are available?
```

Vous devriez voir la liste des outils de gestion EKS disponibles.

Option D : Cline (extension de code VS)

Cline est une extension VS Code populaire qui apporte l'assistance de l'IA directement dans votre éditeur.

1. Ouvrez les paramètres de Cline

- Ouvrez Cline
- Appuyez sur `Cmd+Shift+P` (Mac)/`Ctrl+Shift+P`(Windows) et recherchez « MCP »

2. Ajouter une configuration de serveur MCP

- Cliquez sur « Ajouter un serveur »
- Cliquez sur « Ouvrir la configuration utilisateur »

Créez le fichier de configuration s'il n'existe pas. Assurez-vous de remplacer l'espace réservé à la région (`{region}`) par la région de votre choix.

Pour Mac/Linux :

```
{
  "mcpServers": {
    "eks-mcp": {
      "disabled": false,
      "type": "stdio",
      "command": "uvx",
      "args": [
        "mcp-proxy-for-aws@latest",
        "https://eks-mcp.{region}.api.aws/mcp",
        "--service",
        "eks-mcp",
        "--profile",
        "default",
        "--region",
        "{region}"
      ]
    }
  }
}
```

Pour Windows:

```
{
  "mcpServers": {
    "eks-mcp": {
      "disabled": false,
      "type": "stdio",
      "command": "uvx",
      "args": [
        "--from",
        "mcp-proxy-for-aws@latest",
        "mcp-proxy-for-aws.exe",
        "https://eks-mcp.{region}.api.aws/mcp",
        "--service",
        "eks-mcp",
        "--profile",
        "default",
        "--region",
```

```
        "{region}"
      ]
    }
  }
}
```

Remarque de sécurité : `--read-only` peut être utilisé pour autoriser uniquement les opérations d'outil en lecture seule.

2. Recharger le code VS

Appuyez sur`Cmd+Shift+P`/`Ctrl+Shift+P`et sélectionnez « Développeur : recharger la fenêtre »

3. Vérifier la configuration

Ouvrez Cline et demandez :

```
List the available MCP tools for EKS
```

Étape 2 : (Facultatif) Créez une politique « d'écriture »

Vous pouvez éventuellement créer une politique IAM gérée par le client qui fournit un accès complet au serveur Amazon EKS MCP. Cette politique accorde des autorisations pour utiliser tous les outils du serveur EKS MCP, y compris les outils privilégiés qui peuvent impliquer des opérations d'écriture et les outils en lecture seule. Notez que les autorisations à haut risque (toutes les autorisations associées à la fonction Delete* ou à une ressource IAM illimitée) sont incluses dans cette politique, car elles sont requises pour les ressources setup/teardown du cluster dans l'outil manage_eks_stacks.

```
aws iam create-policy \
 --policy-name EKSMcpWriteManagementPolicy \
 --policy-document "{\"Version\": \"2012-10-17\", \"Statement\": [{\"Effect
\": \"Allow\", \"Action\": [\"eks:DescribeCluster\", \"eks:ListClusters\",
 \"eks:DescribeNodegroup\", \"eks:ListNodegroups\", \"eks:DescribeAddon\",
 \"eks:ListAddons\", \"eks:DescribeAccessEntry\", \"eks:ListAccessEntries\",
 \"eks:DescribeInsight\", \"eks:ListInsights\", \"eks:AccessKubernetesApi\"],
 \"Resource\": \"*\"}, {\"Effect\": \"Allow\", \"Action\": [\"eks:CreateCluster
\", \"eks:DeleteCluster\", \"eks:CreateAccessEntry\", \"eks:TagResource\"],
 \"Resource\": \"*\"}, {\"Effect\": \"Allow\", \"Action\": [\"iam:GetRole\",
 \"iam:ListRolePolicies\", \"iam:ListAttachedRolePolicies\", \"iam:GetRolePolicy\",
 \"iam:GetPolicy\", \"iam:GetPolicyVersion\"], \"Resource\": \"*\"}, {\"Effect\":
```

```
 \"Allow\", \"Action\": [\"iam:TagRole\", \"iam:CreateRole\", \"iam:AttachRolePolicy
\", \"iam:PutRolePolicy\", \"iam:DetachRolePolicy\", \"iam:DeleteRole\"],
 \"Resource\": \"*\"}, {\"Effect\": \"Allow\", \"Action\": [\"iam:PassRole\"],
 \"Resource\": \"*\", \"Condition\": {\"StringEquals\": {\"iam:PassedToService
\": [\"eks.amazonaws.com\", \"ec2.amazonaws.com\"]}}}, {\"Effect\": \"Allow\",
 \"Action\": [\"ec2:CreateVpc\", \"ec2:CreateSubnet\", \"ec2:CreateRouteTable\",
 \"ec2:CreateRoute\", \"ec2:CreateInternetGateway\", \"ec2:CreateNatGateway\",
 \"ec2:CreateSecurityGroup\", \"ec2:AttachInternetGateway\", \"ec2:AssociateRouteTable
\", \"ec2:ModifyVpcAttribute\", \"ec2:ModifySubnetAttribute\", \"ec2:AllocateAddress
\", \"ec2:CreateTags\"], \"Resource\": \"*\"}, {\"Effect\": \"Allow\", \"Action
\": [\"ec2:DeleteVpc\", \"ec2:DeleteSubnet\", \"ec2:DisassociateRouteTable\",
 \"ec2:DeleteRouteTable\", \"ec2:DeleteRoute\", \"ec2:DetachInternetGateway\",
 \"ec2:DeleteInternetGateway\", \"ec2:DeleteNatGateway\", \"ec2:ReleaseAddress\",
 \"ec2:DeleteSecurityGroup\"], \"Resource\": \"*\"}, {\"Effect\": \"Allow\", \"Action
\": [\"ec2:DescribeVpcs\", \"ec2:DescribeSubnets\", \"ec2:DescribeRouteTables\",
 \"ec2:DescribeInternetGateways\", \"ec2:DescribeNatGateways\", \"ec2:DescribeAddresses
\", \"ec2:DescribeSecurityGroups\", \"ec2:DescribeAvailabilityZones\"], \"Resource
\": \"*\"}, {\"Effect\": \"Allow\", \"Action\": [\"cloudformation:CreateStack
\", \"cloudformation:UpdateStack\", \"cloudformation:DeleteStack\",
 \"cloudformation:DescribeStacks\", \"cloudformation:TagResource\"], \"Resource
\": \"*\"}, {\"Effect\": \"Allow\", \"Action\": [\"sts:GetCallerIdentity\"],
 \"Resource\": \"*\"}, {\"Effect\": \"Allow\", \"Action\": [\"logs:StartQuery\",
 \"logs:GetQueryResults\"], \"Resource\": \"*\"}, {\"Effect\": \"Allow\", \"Action\":
 [\"cloudwatch:GetMetricData\"], \"Resource\": \"*\"}, {\"Effect\": \"Allow\", \"Action
\": [\"eks-mcp:*\"], \"Resource\": \"*\"}]}"
```

Étape 3 : vérifier votre configuration

Tester la connexion

Posez une simple question à votre assistant IA pour vérifier la connexion :

```
List all EKS clusters in my {aws} account
```

Vous devriez voir la liste de vos clusters EKS.

Étape 4 : Exécutez vos premières tâches

Exemple 1 : Explorez vos clusters

```
Show me all EKS clusters and their status
What insights does EKS have about my production-cluster?
```

```
Show me the VPC configuration for my staging cluster
```

Exemple 2 : vérifier les ressources Kubernetes

```
Get the details of all the kubernetes resources deployed in my EKS cluster
Show me pods that are not in Running state or pods with any restarts
Get the logs from the aws-node daemonset in the last 30 minutes
```

Exemple 3 : résolution des problèmes

```
Why is my nginx-ingress-controller pod failing to start?
Search the EKS troubleshooting guide for pod networking issues
Show me events related to the failed deployment in the staging namespace
```

Exemple 4 : Création de ressources (si le mode « écriture » est activé)

```
Create a new EKS cluster named demo-cluster with VPC and Auto Mode
Deploy my containerized app from ECR to the production namespace with 3 replicas
Generate a Kubernetes deployment YAML for my Node.js app running on port 3000
```

Configurations courantes

Scénario 1 : AWS profils multiples

Si vous travaillez avec plusieurs AWS comptes, créez des configurations de serveur MCP distinctes.

Pour Mac/Linux :

```
{
  "mcpServers": {
    "eks-mcp-prod": {
      "disabled": false,
      "type": "stdio",
      "command": "uvx",
      "args": [
        "mcp-proxy-for-aws@latest",
        "https://eks-mcp.{region}.api.aws/mcp",
        "--service",
        "eks-mcp",
        "--profile",
        "production",
```

```
        "--region",
        "us-west-2"
      ]
    },
    "eks-mcp-dev": {
      "disabled": false,
      "type": "stdio",
      "command": "uvx",
      "args": [
        "mcp-proxy-for-aws@latest",
        "https://eks-mcp.{region}.api.aws/mcp",
        "--service",
        "eks-mcp",
        "--profile",
        "development",
        "--region",
        "us-east-1"
      ]
    }
  }
}
```

Pour Windows:

```
{
  "mcpServers": {
    "eks-mcp-prod": {
      "disabled": false,
      "type": "stdio",
      "command": "uvx",
      "args": [
        "--from",
        "mcp-proxy-for-aws@latest",
        "mcp-proxy-for-aws.exe",
        "https://eks-mcp.{region}.api.aws/mcp",
        "--service",
        "eks-mcp",
        "--profile",
        "production",
        "--region",
        "us-west-2"
      ]
    },
```

```
    "eks-mcp-dev": {
      "disabled": false,
      "type": "stdio",
      "command": "uvx",
      "args": [
        "--from",
        "mcp-proxy-for-aws@latest",
        "mcp-proxy-for-aws.exe",
        "https://eks-mcp.{region}.api.aws/mcp",
        "--service",
        "eks-mcp",
        "--profile",
        "development",
        "--region",
        "us-east-1"
      ]
    }
  }
}
```

Scénario 2 : lecture seule pour la production

Créez une configuration en lecture seule pour les environnements de production.

Pour Mac/Linux :

```
{
  "mcpServers": {
    "eks-mcp-prod-readonly": {
      "command": "uvx",
      "args": [
        "mcp-proxy-for-aws@latest",
        "https://eks-mcp.{region}.api.aws/mcp",
        "--service",
        "eks-mcp",
        "--profile",
        "production",
        "--region",
        "us-west-2",
        "--read-only"
      ],
      "autoApprove": [
        "list_k8s_resources",
        "get_pod_logs",
```

```
        "get_k8s_events"
      ]
    }
  }
}
```

Pour Windows:

```
{
  "mcpServers": {
    "eks-mcp-prod-readonly": {
      "command": "uvx",
      "args": [
        "--from",
        "mcp-proxy-for-aws@latest",
        "mcp-proxy-for-aws.exe",
        "https://eks-mcp.{region}.api.aws/mcp",
        "--service",
        "eks-mcp",
        "--profile",
        "production",
        "--region",
        "us-west-2",
        "--read-only"
      ],
      "autoApprove": [
        "list_k8s_resources",
        "get_pod_logs",
        "get_k8s_events"
      ]
    }
  }
}
```

Scénario 3 : Développement avec accès complet

Pour les environnements de développement dotés d'un accès complet en écriture.

Pour Mac/Linux :

```
{
  "mcpServers": {
    "eks-mcp-dev-full": {
```

```
      "command": "uvx",
      "args": [
        "mcp-proxy-for-aws@latest",
        "https://eks-mcp.{region}.api.aws/mcp",
        "--service",
        "eks-mcp",
        "--profile",
        "development",
        "--region",
        "us-east-1"
      ]
    }
  }
}
```

Pour Windows:

```
{
  "mcpServers": {
    "eks-mcp-dev-full": {
      "command": "uvx",
      "args": [
        "--from",
        "mcp-proxy-for-aws@latest",
        "mcp-proxy-for-aws.exe",
        "https://eks-mcp.{region}.api.aws/mcp",
        "--service",
        "eks-mcp",
        "--profile",
        "development",
        "--region",
        "us-east-1"
      ]
    }
  }
}
```

Considérations

Sécurité

Ne transmettez pas de secrets ou d'informations sensibles par le biais de mécanismes de saisie autorisés :

- N'incluez pas de secrets ou d'informations d'identification dans les fichiers YAML appliqués avec apply_yaml.
- Ne transmettez pas d'informations sensibles directement dans l'invite au modèle.
- N'incluez pas de secrets dans les CloudFormation modèles ou les manifestes d'applications.
- Évitez d'utiliser les outils MCP pour créer des secrets Kubernetes, car cela nécessiterait de fournir les données secrètes au modèle.
- Évitez de consigner des informations sensibles dans les journaux d'applications des pods Kubernetes.

Sécurité du contenu YAML :

- N'utilisez que des fichiers YAML provenant de sources fiables.
- Le serveur s'appuie sur la validation de l'API Kubernetes pour le contenu YAML et n'effectue pas sa propre validation.
- Auditez les fichiers YAML avant de les appliquer à votre cluster.

Au lieu de transmettre des secrets via MCP :

- Utilisez AWS Secrets Manager ou Parameter Store pour stocker des informations sensibles.
- Configurez le RBAC Kubernetes approprié pour les comptes de service.
- Utilisez les rôles IAM pour les comptes de service (IRSA) pour accéder aux AWS services depuis les pods.

Rédaction de données sensibles :

- Le serveur EKS MCP expédie automatiquement les modèles courants pour les jetons de sécurité, les certificats et autres informations sensibles dans les réponses des outils.
- Les valeurs expurgées sont remplacées par `HIDDEN_FOR_SECURITY_REASONS` pour éviter d'exposer accidentellement les données au modèle.
- Cette rédaction s'applique à toutes les réponses des outils, y compris les journaux, les descriptions des ressources et les données de configuration.

Prochaine étape

Pour les options de configuration, voirthe section called "Configuration". Pour une liste complète des outils, voirthe section called "Outils".

Référence de configuration du serveur Amazon EKS MCP

Ce guide présente toutes les configurations disponibles pour l'outil mcp-proxy-for-awscôté client qui vous permet de vous connecter au serveur MCP Amazon EKS entièrement géré depuis votre IDE.

> **Note**
>
> Le serveur Amazon EKS MCP est en version préliminaire pour Amazon EKS et est sujet à modification.

Exemple

```
{
  "mcpServers": {
    "eks-mcp": {
      "disabled": false,
      "type": "stdio",
      "command": "uvx",
      "args": [
        "mcp-proxy-for-aws@latest",
        "https://eks-mcp.us-west-2.api.aws/mcp",
        "--service",
        "eks-mcp",
        "--profile",
        "default",
        "--region",
        "us-east-1",
        "--read-only"
      ]
    }
  }
}
```

Autorisations IAM

Le rôle utilisé pour se connecter au serveur MCP nécessite des **`eks-mcp:InvokeMcp`**autorisations pour l'initialisation et la récupération d'informations sur les outils disponibles. **`eks-mcp:CallReadOnlyTool`**est requis pour l'utilisation des outils en lecture seule et **`eks-mcp:CallPrivilegedTool`**est requis pour l'utilisation des outils d'accès complet (écriture).

Variables d'environnement

`AWS_PROFILE`(facultatif) nom du profil AWS d'identification à utiliser ; peut être remplacé par l'argument de `--profile` ligne de commande.

- Exemple : `export AWS_PROFILE=production`

`AWS_REGION` AWS région (optionnelle) pour la signature Sigv4 ; la valeur par défaut est `us-west-2` si elle n'est pas définie.

- Exemple : `export AWS_REGION=us-east-1`

Arguments

URL du point de terminaison MCP SigV4 (obligatoire) URL du point de terminaison MCP auquel se connecter.

- Exemple : `https://eks-mcp.us-west-2.api.aws/mcp`

`--service`nom de AWS service (facultatif) pour la signature SigV4 ; détecté automatiquement à partir du nom d'hôte du point de terminaison s'il n'est pas fourni.

- Exemple : `--service eks-mcp`

`--profile`profil AWS d'identification (facultatif) à utiliser. La valeur par défaut est la variable d'`AWS_PROFILE`environnement si elle n'est pas spécifiée.

- Exemple : `--profile production`

`--region` AWS région à utiliser. Utilise la variable d'`AWS_REGION`environnement si elle n'est pas définie, la valeur par défaut est`us-east-1`.

- Exemple : `--region us-west-2`

`--read-only`(facultatif) Désactivez les outils qui peuvent nécessiter des autorisations d'écriture (les outils qui NE nécessitent PAS d'autorisations d'écriture sont annotés avec readOnlyHint =true). Par défaut, tous les outils sont activés.

- Exemple : `--read-only`

Pour plus d'options de configuration, consultez la section Paramètres de configuration.

Outils

Pour obtenir des informations détaillées sur tous les outils disponibles, y compris les paramètres et les exemples d'utilisation, consultezthe section called "Outils".

Référence des outils du serveur Amazon EKS MCP

Le serveur expose les outils MCP suivants.

> **Note**
>
> Le serveur Amazon EKS MCP est en version préliminaire pour Amazon EKS et est sujet à modification.

Outils en lecture seule

Cette section décrit les outils en lecture seule disponibles pour le serveur EKS MCP. Notez que toutes les opérations de l'API Kubernetes en lecture seule peuvent accéder aux deux :

- Clusters privés (voir Point de terminaison privé du cluster)
- Clusters publics

search_eks_documentation

Consultez la documentation EKS pour up-to-date obtenir des informations et des conseils. Cet outil donne accès à la dernière documentation EKS, notamment aux nouvelles fonctionnalités et aux améliorations récentes dont les agents ne sont peut-être pas au courant.

Paramètres :

- requête (obligatoire) : votre question ou requête de recherche spécifique concernant la documentation, les fonctionnalités ou les meilleures pratiques d'EKS.
- limite (facultatif) : nombre maximum de résultats de documentation à renvoyer (1 à 10). Valeur par défaut : 5.

search_eks_troubleshooting_guide

Effectue une recherche dans le guide de dépannage EKS pour obtenir des informations de dépannage sur la base d'une requête. Il aide à identifier les problèmes courants et fournit step-by-step des solutions.

Paramètres :

- requête (obligatoire) : votre question ou description du problème spécifique concernant le dépannage d'EKS.

describe_eks_resource

Récupère des informations détaillées sur une ressource de cluster EKS spécifique, notamment la configuration, l'état et les métadonnées.

Paramètres :

- cluster_name (obligatoire) : nom du cluster EKS (obligatoire pour les ressources étendues au cluster).
- resource_type (obligatoire) : type de ressource EKS à décrire. Valeurs valides :
- accessentry (nécessite cluster_name et resource_name comme PrincipalArn)
- addon (nécessite cluster_name et resource_name comme nom d'addon)
- cluster (nécessite cluster_name), nodegroup (nécessite cluster_name et resource_name comme nom de groupe de nœuds).
- resource_name (facultatif) : nom de la ressource spécifique à décrire (obligatoire pour la plupart des types de ressources).

liste_eks_resources

Répertorie les ressources EKS d'un type spécifique, renvoyant un résumé de toutes les ressources du type spécifié qui sont accessibles.

Paramètres :

- resource_type (obligatoire) : type de ressource EKS à répertorier. Valeurs valides :
- accessentry (nécessite cluster_name)
- addon (nécessite cluster_name)
- cluster (aucun paramètre supplémentaire requis)
- groupe de nœuds (nécessite cluster_name).
- cluster_name (facultatif) : nom du cluster EKS (obligatoire pour les ressources étendues au cluster).

get_eks_insights

Récupère des informations sur le cluster EKS et des recommandations d'optimisation. Fournit des informations exploitables pour la sécurité, les performances et l'optimisation des coûts sur la base des AWS meilleures pratiques et de l'analyse des clusters.

Paramètres :

- cluster_name (obligatoire) : nom du cluster EKS.
- catégorie (facultatif) : Catégorie facultative par laquelle filtrer les informations (par exemple, « MISCONFIGURATION » ou « UPGRADE_READINESS »).
- insight_id (facultatif) : ID facultatif d'un aperçu spécifique pour lequel obtenir des informations détaillées.
- next_token (facultatif) : jeton facultatif pour la pagination afin d'obtenir le prochain ensemble de résultats.

get_eks_vpc_config

Récupère la configuration VPC d'un cluster EKS, y compris les sous-réseaux, les tables de routage et la connectivité réseau.

Paramètres :

- cluster_name (obligatoire) : nom du cluster EKS pour lequel vous souhaitez obtenir la configuration VPC.
- vpc_id (facultatif) : ID du VPC spécifique à interroger (facultatif, il utilisera le VPC du cluster s'il n'est pas spécifié).

get_k8s_events

Récupère les événements Kubernetes liés à des ressources spécifiques pour le dépannage et la surveillance.

Paramètres :

- cluster_name (obligatoire) : nom du cluster EKS où se trouve la ressource.
- type (obligatoire) : type de l'objet impliqué (par exemple, « Pod », « Deployment », « Service »). Doit correspondre exactement au type de ressource.
- name (obligatoire) : nom de l'objet impliqué pour lequel vous souhaitez obtenir des événements.
- namespace (facultatif) : espace de noms de l'objet impliqué. Nécessaire pour les ressources avec espace de noms (comme les pods, les déploiements). Non requis pour les ressources étendues au cluster (comme les nœuds,). PersistentVolumes

get_pod_logs

Récupère les journaux des modules d'un cluster EKS avec des options de filtrage.

Paramètres :

- cluster_name (obligatoire) : nom du cluster EKS sur lequel le pod est exécuté.
- espace de noms (obligatoire) : espace de noms Kubernetes où se trouve le pod.
- pod_name (obligatoire) : nom du pod à partir duquel récupérer les journaux.
- container_name (facultatif) : nom du conteneur spécifique à partir duquel obtenir les journaux. Obligatoire uniquement si le pod contient plusieurs contenants.
- limit_bytes (facultatif) : nombre maximum d'octets à renvoyer. Par défaut : 10 Ko (10 240 octets).
- previous (facultatif) : renvoie les journaux des conteneurs précédemment fermés (la valeur par défaut est false). Utile pour obtenir les journaux des pods qui redémarrent.

- since_seconds (facultatif) : renvoie uniquement les journaux dont la durée est supérieure à ce nombre de secondes. Utile pour obtenir des journaux récents sans récupérer l'historique complet.
- tail_lines (facultatif) : nombre de lignes à renvoyer depuis la fin des journaux. Par défaut : 100.

liste_api_versions

Répertorie toutes les versions d'API disponibles dans le cluster Kubernetes spécifié.

Paramètres :

- cluster_name (obligatoire) : nom du cluster EKS.

liste_k8s_resources

Répertorie les ressources Kubernetes d'un type spécifique dans un cluster EKS.

Paramètres :

- cluster_name (obligatoire) : nom du cluster EKS où se trouvent les ressources.
- kind (obligatoire) : type de ressources Kubernetes à répertorier (par exemple, « Pod », « Service », « Deployment »). Utilisez l'`list_api_versions`outil pour trouver les types de ressources disponibles.
- api_version (obligatoire) : version API des ressources Kubernetes (par exemple, « v1 », « 1 »). apps/v1', 'networking.k8s.io/v Utilisez l'`list_api_versions`outil pour trouver les versions d'API disponibles.
- field_selector (facultatif) : sélecteur de champs pour filtrer les ressources (par exemple, 'metadata.name=my-pod, status.phase=Running'). Utilise la même syntaxe que l'indicateur --field-selector de kubectl.
- label_selector (facultatif) : sélecteur d'étiquettes pour filtrer les ressources (par exemple, 'app=nginx, tier=frontend'). Utilise la même syntaxe que le drapeau de kubectl. `--selector`
- namespace (facultatif) : espace de noms des ressources Kubernetes à répertorier. Si elles ne sont pas fournies, les ressources seront répertoriées dans tous les espaces de noms (pour les ressources avec espace de noms).

read_k8s_resource

Récupère des informations détaillées sur une ressource Kubernetes spécifique dans un cluster EKS.

Paramètres :

- api_version (obligatoire) : version de l'API de la ressource Kubernetes (par exemple, « v1", « 1"). apps/v1", "networking.k8s.io/v
- cluster_name (obligatoire) : nom du cluster EKS où se trouve la ressource.
- kind (obligatoire) : type de ressource Kubernetes (par exemple, « Pod », « Service », « Deployment »).
- name (obligatoire) : nom de la ressource Kubernetes à lire.
- namespace (facultatif) : espace de noms de la ressource Kubernetes. Obligatoire pour les ressources avec espace de noms. Non requis pour les ressources étendues au cluster (comme les nœuds,). PersistentVolumes

generate_app_manifest

Génère des manifestes de déploiement et de service Kubernetes standardisés pour les applications conteneurisées.

Paramètres :

- app_name (obligatoire) : nom de l'application. Utilisé pour les noms de déploiement et de service, ainsi que pour les étiquettes.
- image_uri (obligatoire) : URI de l'image ECR complète avec balise (par exemple, 123456789012.dkr.ecr.region.amazonaws.com/repo:tag). Doit inclure le chemin complet du référentiel et le tag.
- load_balancer_scheme (facultatif) : schéma d'équilibreur de charge. AWS Valeurs valides :
- « interne » (VPC privé uniquement)
- « connecté à Internet » (accès public).
- Par défaut : « interne ».
- cpu (facultatif) : demande de processeur pour chaque conteneur (par exemple, « 100 m » pour 0,1 cœur de processeur, « 500 m » pour un demi-cœur). Par défaut : « 100 m ».
- mémoire (facultatif) : demande de mémoire pour chaque conteneur (par exemple, « 128 Mi » pour 128 Mo, « 1 Gi » pour 1 GiB). Par défaut : « 128 Mi »
- espace de noms (facultatif) : espace de noms Kubernetes dans lequel déployer l'application. Par défaut : « par défaut ».

- port (facultatif) : port du conteneur sur lequel l'application écoute. Par défaut: 80
- répliques (facultatif) : nombre de répliques à déployer. Par défaut: 2

get_cloudwatch_logs

CloudWatch Interroge les journaux avec filtrage basé sur les paramètres d'entrée et prend en charge les groupes de journaux standard utilisés pour l'observabilité du cluster EKS.

Paramètres :

- cluster_name (obligatoire) : nom du cluster EKS où se trouve la ressource. Utilisé pour construire le nom du groupe de CloudWatch journaux.
- resource_type (obligatoire) : type de ressource à rechercher dans les journaux. Valeurs valides : +"pod », « node », « container », « cluster ». Cela détermine la manière dont les journaux sont filtrés.
- log_type (obligatoire) : type de journal à interroger. Valeurs valides :
- « application » : Container/application journaux
- « host » : journaux système au niveau du nœud
- « performance » : journaux des indicateurs de performance
- « plan de contrôle » : journaux du plan de contrôle EKS
- your-log-group-name« » : Fournissez directement un nom de groupe de CloudWatch journaux personnalisé.
- resource_name (facultatif) : nom de la ressource à rechercher dans les messages du journal (par exemple, nom du pod, nom du nœud, nom du conteneur). Utilisé pour filtrer les journaux pour une ressource spécifique.
- minutes (facultatif) : nombre de minutes nécessaires pour consulter les journaux. Par défaut : 15. Ignoré si start_time est fourni. Utilisez des valeurs plus petites pour les numéros récents, des valeurs plus élevées pour l'analyse historique.
- start_time (facultatif) : heure de début au format ISO (par exemple, « 2023-01-01T 00:00:00 Z »). S'il est fourni, remplace le paramètre minutes.
- end_time (facultatif) : heure de fin au format ISO (par exemple, « 2023-01-01T 01:00:00 Z »). Si ce n'est pas le cas, la valeur par défaut est l'heure actuelle.
- champs (facultatif) : champs personnalisés à inclure dans les résultats de la requête (par défaut, « @timestamp, @message »). Utilisez la syntaxe des champs CloudWatch Logs Insights.

- filter_pattern (facultatif) : modèle de filtre CloudWatch Logs supplémentaire à appliquer. Utilise la syntaxe de CloudWatch Logs Insights (par exemple, « ERROR », « field=value »).
- limite (facultatif) : nombre maximum d'entrées de journal à renvoyer. Utilisez des valeurs inférieures (10 à 50) pour des requêtes plus rapides, des valeurs plus élevées (100 à 1 000) pour des résultats plus complets. Des valeurs élevées peuvent avoir un impact sur les performances.

get_cloudwatch_metrics

Récupère CloudWatch les métriques et les points de données pour la surveillance du cluster EKS et l'analyse des performances. Gère les métriques de Container Insights, les métriques personnalisées et les périodes et dimensions configurables.

Paramètres :

- cluster_name (obligatoire) : nom du cluster EKS pour lequel obtenir des métriques.
- dimensions (obligatoire) : dimensions à utiliser pour la requête CloudWatch métrique sous forme de chaîne JSON. Doit inclure les dimensions appropriées pour le type de ressource et la métrique (par exemple, '{» ClusterName « : « my-cluster », "PodName« : « my-pod », « Namespace » : « default"}').
- metric_name (obligatoire) : nom de la métrique à récupérer. Exemples courants :
- `cpu_usage_total`: Utilisation totale du processeur
- `memory_rss`: utilisation de la mémoire par Resident Set Size
- `network_rx_bytes`: octets réseau reçus
- `network_tx_bytes`: octets réseau transmis
- namespace (obligatoire) : espace de CloudWatch noms dans lequel la métrique est stockée. Valeurs communes :
- `"ContainerInsights"`: Pour les métriques relatives aux conteneurs
- `"AWS/EC2"`: EC2 Par exemple, les métriques
- `"AWS/EKS"`: pour les métriques du plan de contrôle EKS
- minutes (facultatif) : nombre de minutes nécessaires pour consulter les statistiques. Par défaut : 15. Ignoré si start_time est fourni.
- start_time (facultatif) : heure de début au format ISO (par exemple, « 2023-01-01T 00:00:00 Z »). S'il est fourni, remplace le paramètre minutes.

- end_time (facultatif) : heure de fin au format ISO (par exemple, « 2023-01-01T 01:00:00 Z »). Si ce n'est pas le cas, la valeur par défaut est l'heure actuelle.
- limite (facultatif) : nombre maximum de points de données à renvoyer. Des valeurs plus élevées (100-1000) fournissent des données plus détaillées mais peuvent avoir un impact sur les performances. Par défaut : 50.
- période (facultatif) : période en secondes pour les points de données métriques. Par défaut : 60 (1 minute). Des valeurs plus faibles (1 à 60) fournissent une résolution plus élevée mais peuvent être moins disponibles.
- stat (facultatif) : statistique à utiliser pour l'agrégation des métriques. Par défaut : « Moyen ». Valeurs valides :
- `Average`: valeur moyenne au cours de la période
- `Sum:`Valeur totale au cours de la période
- `Maximum`: valeur la plus élevée de la période
- `Minimum`: valeur la plus faible de la période
- `SampleCount`: Nombre d'échantillons au cours de la période.

get_eks_metrics_guidance

Obtenez CloudWatch des conseils sur les métriques pour des types de ressources spécifiques dans les clusters EKS. Utile pour l'agent lorsqu'il détermine les dimensions correctes à utiliser avec l'outil get_cloudwatch_metrics.

Paramètres :

- resource_type (obligatoire) : type de ressource pour laquelle obtenir des métriques (cluster, nœud, pod, espace de noms, service).

get_policies_for_role

Récupère toutes les politiques associées à un rôle IAM spécifié, y compris la politique d'acceptation des rôles, les politiques gérées et les politiques intégrées.

Paramètres :

- role_name (obligatoire) : nom du rôle IAM pour lequel les politiques doivent être obtenues. Le rôle doit exister dans votre AWS compte.

Outils d'accès complet (écriture)

Cette section décrit les outils en lecture seule disponibles pour le serveur EKS MCP. Notez que (à partir d' « aujourd'hui ») toutes les opérations d'écriture de l'API Kubernetes ne peuvent accéder qu'à :

- Clusters publics (endpointPublicAccess=vrai)

manage_k8s_resource

Gère une seule ressource Kubernetes avec des opérations d'écriture (création, mise à jour, correction ou suppression).

Paramètres :

- opération (obligatoire) : opération à effectuer sur la ressource. Valeurs valides :
- `create:`Création d'une nouvelle ressource
- `replace`: Remplacer une ressource existante
- `patch`: Mettre à jour des champs spécifiques d'une ressource existante
- `delete`: Supprimer une ressource existante
- Remarque : utilisez read_k8s_resource pour lire les ressources et list_k8s_resources pour répertorier plusieurs ressources.
- cluster_name (obligatoire) : nom du cluster EKS où se trouve ou sera créée la ressource.
- kind (obligatoire) : type de ressource Kubernetes (par exemple, « Pod », « Service », « Deployment »).
- api_version (obligatoire) : version de l'API de la ressource Kubernetes (par exemple, « v1", « 1"). apps/v1", "networking.k8s.io/v
- body (facultatif) : définition de la ressource sous forme de chaîne JSON. Nécessaire pour les opérations de création, de remplacement et de correction. Pour créer et remplacer, il doit s'agir d'une définition de ressource complète. Pour le correctif, celui-ci ne doit contenir que les champs à mettre à jour.
- name (facultatif) : nom de la ressource Kubernetes. Obligatoire pour toutes les opérations à l'exception de la création (où elle peut être spécifiée dans le corps).
- namespace (facultatif) : espace de noms de la ressource Kubernetes. Obligatoire pour les ressources avec espace de noms. Non requis pour les ressources étendues au cluster (comme les nœuds,). PersistentVolumes

apply_yaml

Applique les manifestes YAML Kubernetes à un cluster EKS.

Paramètres :

- cluster_name (obligatoire) : nom du cluster EKS dans lequel les ressources seront créées ou mises à jour.
- espace de noms (obligatoire) : espace de noms Kubernetes auquel appliquer les ressources. Sera utilisé pour les ressources avec espacement de noms qui ne spécifient pas d'espace de noms.
- yaml_content (obligatoire) : contenu YAML à appliquer au cluster. Peut contenir plusieurs documents séparés par un « ---».
- force (facultatif) : s'il faut mettre à jour les ressources si elles existent déjà (similaire à kubectl apply). Définissez cette valeur sur false pour créer uniquement de nouvelles ressources.

manage_eks_stacks

Gère les CloudFormation piles EKS avec des opérations de génération de modèles, de déploiement, de description et de suppression des clusters EKS et de leur infrastructure sous-jacente. La création d'un cluster prend généralement 15 à 20 minutes. Pour les opérations de déploiement et de suppression, la pile doit avoir été créée par cet outil (c'est-à-dire étiquetée avec CreatedBy =EksMcpServer).

Paramètres :

- cluster_name (obligatoire) : nom du cluster EKS (pour les opérations de génération, de déploiement, de description et de suppression). Ce nom sera utilisé pour dériver le nom de la CloudFormation pile et sera intégré aux ressources du cluster.
- opération (obligatoire) : opération à effectuer. Valeurs valides :
- `generate`: générer un CloudFormation modèle
- `deploy`: Déployer une CloudFormation pile (nécessite`template_content`)
- `describe`: Describe/read une CloudFormation pile (en lecture seule)
- `delete`: Supprimer une CloudFormation pile
- template_content (facultatif) : contenu du CloudFormation modèle (pour les opérations de déploiement). Il doit s'agir du contenu complet du modèle YAML ou JSON. Supporte à la fois les ressources uniques et le contenu YAML multidocument séparés par « ---».

ajouter_inline_policy

Ajoute une nouvelle politique intégrée à un rôle IAM.

Paramètres :

- autorisations (obligatoire) : autorisations à inclure dans la politique sous forme de chaînes JSON représentant les déclarations de politique IAM. Il peut s'agir d'une chaîne JSON unique ou d'un tableau de chaînes JSON.
- policy_name (obligatoire) : nom de la politique en ligne à créer. Doit être unique au sein du rôle.
- role_name (obligatoire) : nom du rôle IAM auquel ajouter la politique. Le rôle doit exister.

Utiliser Amazon Q Developer sur la console Amazon EKS

Amazon Elastic Kubernetes Service (EKS) s'intègre à Amazon Q pour fournir un dépannage basé sur l'IA directement dans la console de gestion. AWS Cette intégration vous permet d'étudier et de résoudre rapidement les problèmes de cluster, de plan de contrôle, de nœud et de charge de travail avec l'aide de l'IA d'Amazon Q.

Comment ça marche

La console Amazon EKS affiche les boutons Inspect with Amazon Q de manière contextuelle, ainsi que les erreurs ou les problèmes rencontrés sur l'ensemble de la console. Lorsque vous cliquez sur ce bouton, Amazon Q analyse automatiquement le problème et ouvre un panneau de discussion sur le côté droit de la console avec les résultats de l'enquête, l'analyse des causes profondes et les mesures d'atténuation suggérées.

L'intégration apparaît aux emplacements suivants dans la console Amazon EKS :

- État du cluster : étudiez les problèmes de santé du cluster et les messages d'état dans l'onglet État du cluster du tableau de bord d'observabilité
- Plan de contrôle - Résoudre les erreurs et les avertissements relatifs aux composants du plan de contrôle dans le cadre de la surveillance du plan de contrôle dans le tableau de bord d'observabilité
- Informations sur les mises à niveau : analysez les obstacles potentiels aux mises à niveau et les problèmes de compatibilité dans le tableau de bord des informations sur les mises à niveau dans l'observabilité
- État des nœuds : étudiez les problèmes au niveau des nœuds affectant la capacité du cluster dans Problèmes de santé des nœuds dans le tableau de bord d'observabilité

- Charges de travail : analysez les événements Kubernetes sur les pods indiquant des défaillances ou des problèmes

Utilisation d'Amazon Q pour le dépannage

Pour étudier un problème lié à Amazon Q

1. Ouvrez la console Amazon EKS.
2. Choisissez le nom du cluster à étudier.
3. Lorsque vous recevez un message d'erreur ou un indicateur de problème, recherchez le bouton Inspecter avec Amazon Q. Le bouton apparaît de manière contextuelle à côté du problème ou dans la vue détaillée de l'état de l'erreur.
4. Choisissez Inspect with Amazon Q.
5. Amazon Q examine automatiquement le problème et affiche l'analyse dans un panneau de discussion sur le côté droit de la console.
6. Passez en revue les résultats de l'enquête, y compris l'analyse des causes profondes et les mesures d'atténuation suggérées.
7. Vous pouvez poursuivre la conversation en posant des questions de suivi à Amazon Q à propos du problème.

Remarque L'intégration Amazon Q n'apparaît qu'en cas d'erreur, d'avertissement ou de problème nécessitant une enquête. Cela n'apparaît pas lorsque les ressources sont saines.

Considérations

Tenez compte des points suivants lorsque vous utilisez Amazon Q avec Amazon EKS :

- Opérations en lecture seule : l'intégration Amazon Q effectue uniquement des opérations de lecture sur les ressources de votre cluster. Il n'effectue aucune action de mutation ou d'écriture dans la configuration ou les charges de travail de votre cluster.
- Traitement entre régions : Amazon Q peut traiter les données entre les AWS régions afin de fournir une analyse basée sur l'IA. Pour plus d'informations sur le traitement interrégional, consultez la section Traitement interrégional dans le manuel Amazon Q Developer User Guide.
- AWS Console de gestion uniquement : cette intégration est uniquement disponible via la console AWS de gestion. Il n'est pas disponible via la AWS CLI ou AWS APIs l'infrastructure en tant qu'outils de code.

En savoir plus

Pour plus d'informations sur l'utilisation d'Amazon Q, consultez Discuter avec Amazon Q dans le manuel Amazon Q Developer User Guide.

Gestion des versions sur Amazon EKS

Cette section est conçue pour vous aider à comprendre le fonctionnement de la gestion des versions Kubernetes dans Amazon Elastic Kubernetes Service (EKS). Vous y trouverez des informations détaillées sur les versions prises en charge, nos politiques de dépréciation et de support, les processus de mise à niveau et les considérations clés pour la gestion des versions de votre cluster.

Rubriques

- Comprendre le cycle de vie des versions de Kubernetes sur EKS
- Afficher les versions de plateforme Amazon EKS pour chaque version de Kubernetes

Comprendre le cycle de vie des versions de Kubernetes sur EKS

> **Tip**
>
> Inscrivez-vous aux prochains ateliers Amazon EKS.

Kubernetes évolue rapidement avec de nouvelles fonctionnalités, des mises à jour de conception et des corrections de bogues. La communauté publie de nouvelles versions mineures de Kubernetes (telles que `1.33`) en moyenne une fois tous les quatre mois. Amazon EKS suit le cycle de publication en amont et d'obsolescence pour les versions mineures. À mesure que de nouvelles versions Kubernetes deviennent disponibles dans Amazon EKS, nous vous recommandons de mettre à jour de manière proactive vos clusters pour utiliser la dernière version disponible.

Une version mineure bénéficie d'un support standard dans Amazon EKS pendant les 14 premiers mois suivant sa publication. Une fois la date de fin du support standard passée, la version bénéficie d'un support étendu pendant les 12 mois suivants. Le support étendu vous permet de conserver une version spécifique de Kubernetes plus longtemps, moyennant un coût supplémentaire par heure de cluster. Si vous n'avez pas mis à jour votre cluster avant la fin de la période de support étendu, celui-ci est automatiquement mis à niveau vers la plus ancienne version étendue actuellement prise en charge.

Le support étendu est activé par défaut. Pour le désactiver, consultez Désactiver le support étendu EKS.

Nous vous recommandons de créer votre cluster avec la dernière version disponible Kubernetes prise en charge par Amazon EKS. Si votre application nécessite une version spécifique de Kubernetes, vous pouvez en sélectionner une plus ancienne. Vous pouvez créer de nouveaux clusters Amazon EKS sur n'importe quelle version proposée dans le cadre du support standard ou étendu.

Versions disponibles bénéficiant du support standard

Les versions Kubernetes suivantes sont actuellement disponibles dans le cadre du support standard Amazon EKS :

- `1.35`
- `1.34`
- `1.33`
- `1.32`

Pour connaître les modifications importantes à prendre en compte pour chaque version dans le cadre du support standard, consultez Support standard des versions Kubernetes.

Versions disponibles bénéficiant du support étendu

Les versions Kubernetes suivantes sont actuellement disponibles dans le cadre du support étendu Amazon EKS :

- `1.31`
- `1.30`
- `1.29`

Pour connaître les modifications importantes à prendre en compte pour chaque version dans le cadre du support étendu, consultez Support étendu des versions Kubernetes.

Calendrier de publication Amazon EKS Kubernetes

Le tableau suivant présente les dates de publication et de support importantes à prendre en compte pour chaque version de Kubernetes. La facturation du support étendu commence au début de la

journée où la version atteint la fin du support standard, dans le fuseau horaire UTC+0. Les dates indiquées dans le tableau suivant utilisent le fuseau horaire UTC+0.

> **Note**
>
> Les dates avec seulement un mois et une année sont approximatives et sont mises à jour avec une date exacte quand elles sont connues.

Pour recevoir des notifications concernant toutes les modifications apportées aux fichiers sources de cette page de documentation spécifique, vous pouvez vous abonner à l'URL suivante à l'aide d'un lecteur RSS :

```
https://github.com/awsdocs/amazon-eks-user-guide/commits/mainline/latest/ug/versioning/kubernetes-versions.adoc.atom
```

Version de Kubernetes	Version en amont	Version d'Amazon EKS	Date de fin du support standard	Date de fin du support étendu
1.35	17 décembre 2025	27 janvier 2026	27 mars 2027	27 mars 2028
1.34	27 août 2025	2 octobre 2025	2 décembre 2026	2 décembre 2027
1.33	23 avril 2025	29 mai 2025	29 juillet 2026	29 juillet 2027
1.32	11 décembre 2024	23 janvier 2025	23 mars 2026	23 mars 2027
1.31	13 août 2024	26 septembre 2024	26 novembre 2025	26 novembre 2026
1.30	17 avril 2024	23 mai 2024	23 juillet 2025	23 juillet 2026
1.29	13 décembre 2023	23 janvier 2024	23 mars 2025	23 mars 2026

Obtenir des informations de version avec la AWS CLI

Vous pouvez utiliser la AWS CLI pour obtenir des informations sur les versions de Kubernetes disponibles sur EKS, telles que la date de fin du Support standard.

Pour récupérer des informations sur les versions de Kubernetes disponibles sur EKS à l'aide de la CLI AWS

1. Ouvrez votre terminal.
2. Assurez-vous que la AWS CLI est installée et configurée. Pour plus d'informations, consultez Installation ou mise à jour vers la dernière version de la CLI.
3. Exécutez la commande suivante :

```
aws eks describe-cluster-versions
```

4. La commande renverra une sortie JSON avec des détails sur les versions de cluster disponibles. Voici un exemple de sortie :

```
{
    "clusterVersions": [
        {
            "clusterVersion": "1.31",
            "clusterType": "eks",
            "defaultPlatformVersion": "eks.21",
            "defaultVersion": true,
            "releaseDate": "2024-09-25T17:00:00-07:00",
            "endOfStandardSupportDate": "2025-11-25T16:00:00-08:00",
            "endOfExtendedSupportDate": "2026-11-25T16:00:00-08:00",
            "status": "STANDARD_SUPPORT",
            "kubernetesPatchVersion": "1.31.3"
        }
    ]
}
```

La sortie fournit les informations suivantes pour chaque version de cluster :

- `clusterVersion` : la version Kubernetes du cluster EKS
- `clusterType` : le type de cluster (par exemple, « eks »)
- `defaultPlatformVersion` : la version par défaut de la plateforme EKS

- `defaultVersion` : s'il s'agit de la version par défaut
- `releaseDate` : la date à laquelle cette version a été publiée
- `endOfStandardSupportDate` : la date de fin du support standard
- `endOfExtendedSupportDate` : la date de fin du support étendu
- `status` : état actuel du support de la version, tel que `STANDARD_SUPPORT` ou `EXTENDED_SUPPORT`
- `kubernetesPatchVersion` : version spécifique du correctif Kubernetes

Version d'Amazon EKS FAQs

Combien de versions Kubernetes sont disponibles dans le cadre du support standard ?

> Conformément au support de la communauté Kubernetes pour les versions Kubernetes, Amazon EKS s'engage à offrir un support pour au moins trois versions Kubernetes à tout moment. Nous annoncerons la date de fin du support standard d'une version mineure donnée de Kubernetes au moins 60 jours à l'avance. En raison du processus de qualification et de publication d'Amazon EKS pour les nouvelles versions de Kubernetes, la date de fin du support standard d'une version de Kubernetes sur Amazon EKS sera postérieure à la date à laquelle le projet Kubernetes cessera de prendre en charge la version en amont.

Pendant combien de temps une version de Kubernetes bénéficie-t-elle du support standard d'Amazon EKS ?

> Une version de Kubernetes bénéficie du support standard pendant 14 mois après sa première mise à disposition sur Amazon EKS. Cela vaut même si Kubernetes en amont ne prend plus en charge une version disponible sur Amazon EKS. Nous rétroportons les correctifs de sécurité applicables aux versions de Kubernetes prises en charge sur Amazon EKS.

Suis-je averti lorsque le support standard d'une version Kubernetes sur Amazon EKS prend fin ?

> Oui. Si l'un des clusters de votre compte exécute la version dont le support approche de la fin, Amazon EKS envoie une notification via le AWS Health Dashboard environ 12 mois après la sortie de la version Kubernetes sur Amazon EKS. L'avis indique la date de fin de la prise en charge. Le délai est d'au moins 60 jours à compter de la date de l'avis.

Quelles sont les fonctionnalités Kubernetes prises en charge par Amazon EKS ?

> Amazon EKS prend en charge toutes les fonctionnalités disponibles pour tous (GA) de l'API Kubernetes. Les nouvelles versions bêta APIs ne sont pas activées dans les clusters par défaut.

Cependant, la version bêta existante APIs et les nouvelles versions de la version bêta existante APIs continuent d'être activées par défaut. Les fonctionnalités alpha ne sont pas prises en charge.

Les groupes de nœuds gérés par Amazon EKS sont-ils automatiquement mis à jour en même temps que la version du plan de contrôle du cluster ?

Non. Un groupe de nœuds géré crée des instances Amazon EC2 dans votre compte. Ces instances ne sont pas automatiquement mises à niveau lorsque vous ou Amazon EKS mettez à jour votre plan de contrôle. Pour de plus amples informations, veuillez consulter the section called "Mettre à jour". Nous vous recommandons de conserver la même version Kubernetes sur votre plan de contrôle et vos nœuds.

Les groupes de nœuds autogérés sont-ils automatiquement mis à jour en même temps que la version du plan de contrôle du cluster ?

Non. Un groupe de nœuds autogérés inclut des instances Amazon EC2 dans votre compte. Ces instances ne sont pas mises à niveau automatiquement lorsque la version du plan de contrôle est mise à jour par vous ou par Amazon EKS en votre nom. Un groupe de nœuds autogérés n'a aucune indication dans la console qu'il doit être mis à jour. Vous pouvez afficher la version `kubelet` installée sur un nœud en sélectionnant le nœud dans la boîte de dialogue Nœuds dans la Présentation de votre cluster pour déterminer quels nœuds doivent être mis à jour. Vous devez mettre à jour les nœuds manuellement. Pour de plus amples informations, veuillez consulter the section called "Méthodes de mise à jour".

Le projet Kubernetes teste la compatibilité entre le plan de contrôle et les nœuds pour un maximum de trois versions mineures. Par exemple, les nœuds `1.30` continuent à fonctionner tant qu'ils sont orchestrés par un plan de contrôle `1.33`. Cependant, il n'est pas recommandé d'exécuter un cluster avec des nœuds qui ont trois versions mineures de retard par rapport au plan de contrôle. Pour plus d'informations, consultez Version Kubernetes et politique de prise en charge d'asymétrie de version dans la documentation Kubernetes. Nous vous recommandons de conserver la même version Kubernetes sur votre plan de contrôle et vos nœuds.

Les pods exécutés sur Fargate sont-ils automatiquement mis à niveau avec une mise à niveau automatique de la version du plan de contrôle du cluster ?

Non. Nous recommandons fortement d'exécuter les pods Fargate dans le cadre d'un contrôleur de réplication, tel qu'un déploiement Kubernetes. Procédez ensuite à un redémarrage progressif de tous les pods Fargate. La nouvelle version du pod Fargate est déployée avec une version `kubelet` identique à celle de votre plan de contrôle de cluster mis à jour. Pour plus d'informations, consultez Déploiements dans la documentation Kubernetes.

> **Important**
>
> Si vous mettez à jour le plan de contrôle, vous devez également mettre à jour les nœuds Fargate vous-même. Pour mettre à jour les nœuds Fargate, supprimez le pod Fargate représenté par le nœud et redéployez le pod. Le nouveau pod est déployé avec une version `kubelet` identique à celle de votre cluster.

Quelles versions de Kubernetes sont prises en charge pour les nœuds hybrides ?

Les nœuds hybrides Amazon EKS prennent en charge les mêmes versions de Kubernetes que les clusters Amazon EKS avec d'autres types de nœuds de calcul, y compris la prise en charge des versions standard et étendues de Kubernetes. Les nœuds hybrides ne sont pas automatiquement mis à niveau lorsque vous mettez à niveau votre version de plan de contrôle et vous êtes responsable de la mise à niveau de vos nœuds hybrides. Pour de plus amples informations, veuillez consulter the section called "Mettre à niveau les nœuds hybrides".

Support étendu d'Amazon EKS FAQs

Les termes « support standard » et « support étendu » sont nouveaux pour moi. Que signifient ces termes ?

Le support standard d'une version Kubernetes dans Amazon EKS commence lorsqu'Amazon EKS publie une version Kubernetes et prend fin 14 mois après la date de publication. Le support étendu pour une version Kubernetes commence immédiatement après la fin du support standard et prend fin 12 mois plus tard. Par exemple, le support standard pour la version `1.23` dans Amazon EKS a pris fin le 11 octobre 2023. Le support étendu pour la version `1.23` a commencé le 12 octobre 2023 et a pris fin le 11 octobre 2024.

Que dois-je faire pour bénéficier du support étendu pour les clusters Amazon EKS ?

Vous devrez activer le support étendu (voir Support étendu EKS) pour votre cluster en modifiant la politique de mise à niveau du cluster sur EXTENDED. Par défaut, pour tous les clusters nouveaux et existants, la politique de mise à niveau est définie sur EXTENDED, sauf indication contraire. Consultez Politique de mise à niveau des clusters pour connaître la politique de mise à niveau de votre cluster. Le support standard commencera lorsqu'Amazon EKS publiera une version de Kubernetes et prendra fin 14 mois après la date de publication. Le support étendu pour une

version Kubernetes commence immédiatement après la fin du support standard et prend fin 12 mois plus tard.

Pour quelles versions de Kubernetes puis-je bénéficier du support étendu ?

Vous pouvez exécuter des clusters sur n'importe quelle version jusqu'à 12 mois après la fin du support standard pour cette version. Cela signifie que chaque version sera prise en charge pendant 26 mois dans Amazon EKS (14 mois de support standard plus 12 mois de support étendu).

Que faire si je ne veux pas bénéficier du support étendu ?

Si vous ne voulez pas être automatiquement inscrit au support étendu, vous pouvez mettre à niveau votre cluster vers une version Kubernetes bénéficiant du support standard Amazon EKS. Pour désactiver le support étendu, consultez Désactiver le support étendu EKS. Remarque : si vous désactivez le support étendu, Amazon EKS mettra automatiquement à niveau votre cluster à la fin du support standard.

Que se passera-t-il à la fin des 12 mois de support étendu ?

Amazon EKS mettra automatiquement à niveau vers la version suivante les clusters fonctionnant sur une version Kubernetes dont le cycle de vie de 26 mois (14 mois de support standard plus 12 mois de support étendu) est terminé. La mise à niveau automatique ne concerne que le plan de contrôle Kubernetes. Si vous disposez de nœuds du mode automatique EKS, ceux-ci peuvent être mis à jour automatiquement. Les nœuds autogérés et les groupes de nœuds gérés EKS resteront sur la version précédente.

Passé la date de fin de la période de support étendu, vous ne pouvez plus créer de nouveaux clusters Amazon EKS avec la version non prise en charge. Les plans de contrôle existants sont automatiquement mis à jour par Amazon EKS vers la version prise en charge la plus récente, par le biais d'un processus de déploiement progressif après la date de fin de la prise en charge. Après la mise à jour automatique du plan de contrôle, assurez-vous de mettre manuellement à jour les modules complémentaires des clusters et les nœuds Amazon EC2. Pour de plus amples informations, veuillez consulter the section called “Mises à jour de la version Kubernetes”.

À quel moment précis mon plan de contrôle sera-t-il mis à jour automatiquement après la fin de la période de support étendu ?

Amazon EKS n’est pas en mesure de fournir de délais précis. Les mises à jour automatiques peuvent intervenir à tout moment après la date de fin de la période de support étendu. Vous ne recevrez aucune notification avant la mise à jour. Nous vous recommandons de mettre à jour, de manière proactive, votre plan de contrôle sans vous fier au processus de mise à jour automatique

Amazon EKS. Pour de plus amples informations, veuillez consulter the section called "Mises à jour de la version Kubernetes".

Puis-je laisser mon plan de contrôle sur une version Kubernetes indéfiniment ?

Non La sécurité du cloud AWS est la priorité absolue. Au-delà d'un certain délai (généralement un an), la communauté Kubernetes cesse de publier des correctifs pour les vulnérabilités et expositions courantes (CVE) et décourage la soumission de CVE pour les versions non prises en charge. Cela signifie que les vulnérabilités spécifiques à une ancienne version de Kubernetes pourraient ne même pas être signalées. Les clusters sont dès lors exposés sans préavis et aucune option de remédiation en cas de vulnérabilité n'est possible. Dans ce contexte, Amazon EKS n'autorise pas les plans de contrôle à rester sur une version qui a atteint la fin du support étendu.

Y a-t-il un coût supplémentaire pour bénéficier du support étendu ?

Oui, il y a un coût supplémentaire pour les clusters Amazon EKS fonctionnant sous support étendu. Pour en savoir plus sur les tarifs, consultez le support étendu d'Amazon EKS pour la tarification des versions Kubernetes sur le AWS blog ou sur notre page de tarification.

Que comprend le support étendu ?

Les clusters Amazon EKS sous support étendu bénéficient de l'application de correctifs de sécurité continus pour le plan de contrôle Kubernetes. De plus, Amazon EKS publiera des correctifs pour les modules complémentaires CNI Amazon VPC, `kube-proxy` et CoreDNS pour les versions bénéficiant d'un support étendu. Amazon EKS publiera également des correctifs pour les versions AWS publiées d'Amazon EKS optimisées AMIs pour Amazon Linux, Bottlerocket et Windows, ainsi que pour les nœuds Amazon EKS Fargate pour ces versions. Tous les clusters concernés par le Support étendu continueront d'avoir accès au support technique auprès de AWS.

Existe-t-il des limitations concernant l'application de correctifs pour les composants non Kubernetes dans le cadre d'un support étendu ?

Bien que le support étendu couvre tous les composants spécifiques de Kubernetes AWS, il ne fournira à tout moment qu'un support pour AWS Amazon EKS publié et optimisé pour AMIs Amazon Linux, Bottlerocket et Windows. Cela signifie que vous aurez potentiellement de nouveaux composants (système d'exploitation ou noyau, par exemple) sur votre AMI optimisée pour Amazon EKS lorsque vous utiliserez le support étendu. Par exemple, une fois qu'Amazon Linux 2 aura atteint la fin de son cycle de vie en 2025, Amazon Linux optimisé pour Amazon EKS AMIs sera développé à l'aide d'un système d'exploitation Amazon Linux plus récent. Amazon EKS

annoncera et documentera les divergences importantes en matière de cycle de vie du support, telles que celle-ci, pour chaque version de Kubernetes.

Puis-je créer de nouveaux clusters à l'aide d'une version bénéficiant du support étendu ?

Oui.

Examiner les notes de mise à jour pour les versions Kubernetes bénéficiant d'une assistance standard

> **Tip**
>
> Inscrivez-vous aux prochains ateliers Amazon EKS.

Cette rubrique présente les modifications importantes à prendre en compte pour chaque version Kubernetes bénéficiant d'une assistance standard. Lors de la mise à niveau, examinez attentivement les modifications apportées entre l'ancienne et la nouvelle version de votre cluster.

Kubernetes 1.35

Kubernetes `1.35` est désormais disponible dans Amazon EKS. Pour plus d'informations sur Kubernetes `1.35`, consultez l'annonce officielle de publication.

> **Important**
>
> - Support de Cgroup v1 supprimé : Kubernetes 1.35 déconseille le support de cgroup v1, ce qui signifie que le kubelet refusera de démarrer par défaut sur les nœuds utilisant cgroup v1.
>   - AL2023: AL2023 utilise cgroup v2 par défaut et s'aligne sur le comportement en amont de Kubernetes.
>   - Action requise : Les clients qui ont configuré manuellement AL2023 pour utiliser cgroup v1 doivent soit migrer vers cgroups v2, soit configurer `failCgroupV1: false` manuellement la configuration kubelet.
>   - Bottlerocket : Bottlerocket 1.35 utilise cgroup v2 par défaut, mais définit la configuration kubelet, `failCgroupV1: false` en maintenant la rétrocompatibilité.
>   - Fargate : Fargate continue d'utiliser cgroup v1.

- Containerd 1.x Fin du support : Kubernetes 1.35 est la dernière version compatible avec containerd 1.x. Vous devez passer à containerd 2.0 ou à une version ultérieure avant de passer à la prochaine version de Kubernetes.
- En mars 2026, le projet Kubernetes en amont supprimera Ingress NGINX, un composant d'infrastructure essentiel pour de nombreux environnements Kubernetes.
  - Action requise : les clients d'EKS doivent évaluer s'ils font confiance à Ingress NGINX et commencer à planifier la migration vers des alternatives telles que l'API Gateway ou des contrôleurs Ingress tiers, car aucune autre version ne sera publiée pour les corrections de bogues, les correctifs de sécurité ou les mises à jour après le retrait. Les déploiements existants continueront de fonctionner, mais le fait de continuer à utiliser Ingress NGINX après le retrait expose votre environnement aux risques de sécurité, car aucune des alternatives disponibles n'est un remplacement direct et nécessitera du temps de planification et d'ingénierie. Pour plus d'informations sur cette annonce de Kubernetes, consultez la déclaration officielle des comités de pilotage et de réponse à la sécurité de Kubernetes Ingress NGINX.

- Mises à jour des ressources des modules sur place (stables) : les mises à jour des ressources des modules sur place permettent aux utilisateurs d'ajuster les ressources du processeur et de la mémoire sans redémarrer les pods ou les conteneurs. Auparavant, de telles modifications nécessitaient de recréer des pods, ce qui pouvait perturber les charges de travail, en particulier pour les applications dynamiques ou par lots. La nouvelle fonctionnalité sur place permet une mise à l'échelle verticale plus fluide et sans interruption, améliore l'efficacité et peut également simplifier le développement.
  - Pour plus d'informations, consultez Kubernetes 1.35 : In-Place Pod Resize Graduates to Stable sur le blog de Kubernetes.
- PreferSameNode Distribution du trafic (stable) : le `trafficDistribution` champ Services a été mis à jour pour permettre un contrôle plus explicite du routage du trafic. Une nouvelle option a été introduite pour permettre aux services de hiérarchiser strictement les points de terminaison sur le nœud local lorsqu'ils sont disponibles, sinon de revenir aux points de terminaison distants. `PreferSameNode` Cette modification rend l'API plus explicite quant à la préférence du trafic au sein du nœud actuel.
- StatefulSet MaxUnavailable (Bêta) : Cette fonctionnalité permet les mises à jour parallèles du pod en configurant `maxUnavailable` (par exemple, 3 ou 10 %), ce qui permet aux applications dynamiques telles que les clusters de bases de données de se mettre à jour jusqu'à 60 % plus

rapidement que les one-at-a-time mises à jour séquentielles, réduisant ainsi considérablement les fenêtres de maintenance.

- Support pour Windows Server 2025 : EKS 1.35 ajoute le support pour Windows Server 2025.
- Suppression du drapeau Kubelet : Le `--pod-infra-container-image` drapeau a été supprimé de Kubelet. Les utilisateurs de l'AMI personnalisée doivent supprimer cet indicateur de la configuration de kubelet avant de passer à la version 1.35.
- Avis de dépréciation - Mode IPVS : le mode IPVS dans kube-proxy est obsolète et sera supprimé dans Kubernetes 1.36.

Pour le journal complet des `1.35` modifications de Kubernetes, voir -1.35.md https://github.com/kubernetes/ kubernetes/blob/master/CHANGELOG/CHANGELOG

Kubernetes 1.34

Kubernetes `1.34` est désormais disponible dans Amazon EKS. Pour plus d'informations sur Kubernetes `1.34`, consultez l'annonce officielle de publication.

⚠ Important

- Containerd a été mis à jour vers la version 2.1 dans la version 1.34 pour le lancement.
  - Si vous rencontrez des problèmes après la mise à niveau, consultez les notes de mise à jour de containerd 2.1.
- AWS ne publie pas d'AMI Amazon Linux 2 optimisée pour EKS pour Kubernetes 1.34.
  - AWS vous encourage à migrer vers Amazon Linux 2023. Découvrez comment the section called "Passez à AL2023".
  - Pour de plus amples informations, veuillez consulter the section called "Obsolescence de l'AMI Amazon Linux 2".
- AppArmor est obsolète dans Kubernetes 1.34.
  - Nous vous recommandons de migrer vers d'autres solutions de sécurité des conteneurs telles que seccomp ou Pod Security Standards.
- VolumeAttributesClass (VAC) passe à GA dans Kubernetes 1.34, en migrant de l'API bêta () vers l'API `storage.k8s.io/v1beta1` stable (). `storage.k8s.io/v1`
  - Si vous utilisez le pilote EBS CSI avec des conteneurs annexes AWS gérés (depuis CSI Components sur la galerie ECR), la modification du volume continuera de fonctionner parfaitement sur les clusters EKS 1.31-1.33. AWS patchera les sidecars pour prendre en

charge la version bêta de VAC APIs jusqu'à la fin du support standard d'EKS 1.33 (29 juillet 2026).

- Si vous gérez vous-même vos conteneurs de sidecars CSI, vous devrez peut-être les connecter aux anciennes versions des clusters antérieurs à la version 1.34 pour conserver les fonctionnalités du VAC.
- Pour utiliser les VolumeAttributesClass fonctionnalités GA (telles que l'annulation des modifications), effectuez une mise à niveau vers EKS 1.34 ou version ultérieure.

- Le signataire externe de JWT pour les jetons de compte de service est promu en version bêta. Lorsque vous utilisez des signataires externes, l'indicateur -- service-account-extend-token -expiration n'est plus totalement respecté. Le serveur API impose l'expiration minimale entre l'extension souhaitée (1 an) et la limite du signataire externe (24 heures).
  - Nous vous recommandons d'utiliser des jetons de compte de service liés, qui sont automatiquement montés et pivotés par Kubernetes.

- Base (GA) d'allocation dynamique des ressources APIs (DRA) : L'allocation dynamique des ressources est devenue stable, ce qui permet une gestion efficace du matériel spécialisé par le GPUs biais d'interfaces d'allocation standardisées, simplifiant ainsi la gestion des ressources pour les accélérateurs matériels et améliorant l'utilisation des ressources spécialisées.
- ServiceAccount Jetons projetés pour Kubelet (version bêta) : cette amélioration améliore la sécurité en utilisant des informations d'identification de courte durée pour les extractions d'images de conteneurs au lieu de secrets de longue durée, réduisant ainsi le risque d'exposition des informations d'identification et renforçant le niveau de sécurité global de vos clusters.
- Demandes et limites de ressources au niveau des pods (version bêta) : cette fonctionnalité simplifie la gestion des ressources en permettant le partage de pools de ressources pour les pods à conteneurs multiples, ce qui permet une allocation et une utilisation plus efficaces des ressources pour les applications complexes comportant plusieurs conteneurs.
- Nombre allocable de nœuds CSI mutable (version bêta) : le portail des `MutableCSINodeAllocatableCount` fonctionnalités est activé par défaut dans EKS 1.34, ce qui rend l'attribut du nombre CSINode maximal de volumes attachables mutable et introduit un mécanisme permettant de le mettre à jour de manière dynamique en fonction de la configuration utilisateur au niveau du pilote CSI. Ces mises à jour peuvent être déclenchées soit par des intervalles périodiques, soit par la détection de défaillances, ce qui améliore la fiabilité de la planification dynamique des pods en corrigeant les incohérences entre la capacité d'attachement signalée et réelle sur les nœuds.

  - Pour plus d'informations, consultez Kubernetes v1.34 : nombre de nœuds CSI mutables sur le blog de Kubernetes.
- Avis d'obsolescence - Configuration du pilote cgroup : la configuration manuelle du pilote cgroup est déconseillée au profit de la détection automatique.
  - Impact sur le client : si vous définissez actuellement l'`--cgroup-driver`indicateur manuellement dans votre configuration kubelet, vous devez vous préparer à supprimer cette configuration.
  - Action requise : prévoyez de mettre à jour les scripts d'amorçage des nœuds et les configurations personnalisées des AMI afin de supprimer les paramètres manuels du pilote cgroup avant que la fonctionnalité ne soit supprimée dans une future version de Kubernetes.
  - Pour plus d'informations, consultez la documentation du pilote cgroup.

Pour le journal complet des `1.34` modifications de Kubernetes, voir -1.34.md https://github.com/kubernetes/ kubernetes/blob/master/CHANGELOG/CHANGELOG

Kubernetes 1.33

Kubernetes `1.33` est désormais disponible dans Amazon EKS. Pour plus d'informations sur Kubernetes `1.33`, consultez l'annonce officielle de publication.

> **Important**
>
> - L'API Kubernetes beta d'allocation dynamique des ressources est activée.
>   - Cette API bêta améliore l'expérience de planification et de surveillance des charges de travail nécessitant des ressources telles que GPUs.
>   - L'API bêta est définie par la communauté Kubernetes et est susceptible d'évoluer dans les futures versions de Kubernetes.
>   - Lisez attentivement les étapes relatives aux fonctionnalités dans la documentation de Kubernetes pour comprendre les implications de l'utilisation de la version bêta. APIs
> - AWS ne publie pas d'AMI Amazon Linux 2 optimisée pour EKS pour Kubernetes 1.33.
>   - AWS vous encourage à migrer vers Amazon Linux 2023. Découvrez comment the section called "Passez à AL2023".
>   - Pour de plus amples informations, veuillez consulter the section called "Obsolescence de l'AMI Amazon Linux 2".

- Redimensionnement des ressources des pods sur place (bêta) : le redimensionnement des ressources sur place a été promu au statut bêta, ce qui permet de mettre à jour de manière dynamique les ressources CPU et mémoire des pods existants sans redémarrage. Cela permet une mise à l'échelle verticale des charges de travail avec état sans durée d'indisponibilité et des ajustements transparents des ressources en fonction des modèles de trafic.
- Conteneurs sidecar désormais stables : les conteneurs sidecar sont désormais stables. Ils sont implémentés en tant que conteneurs d'initialisation spéciaux avec `restartPolicy: Always` qui démarrent avant les conteneurs d'application, s'exécutent tout au long du cycle de vie du pod et prennent en charge les sondes pour la signalisation de l'état opérationnel.
  - Pour plus d'informations, consultez Conteneurs sidecar dans la documentation Kubernetes.
- Obsolète de l'API Endpoints : L'API Endpoints est désormais officiellement obsolète et renvoie des avertissements en cas d'accès. Migrez les charges de travail et les scripts pour utiliser plutôt l' EndpointSlices API, qui prend en charge des fonctionnalités modernes telles que le réseau à double pile et en gère plusieurs par service. EndpointSlices
  - Pour plus d'informations, consultez Kubernetes v1.33 : Continuing the transition from Endpoints to EndpointSlice the Kubernetes blog.
- Prise en charge d'Elastic Fabric Adapter : le groupe de sécurité par défaut pour les clusters Amazon EKS prend désormais en charge le trafic Elastic Fabric Adapter (EFA). Le groupe de sécurité par défaut dispose d'une nouvelle règle sortante qui autorise le trafic EFA vers la même destination que le groupe de sécurité. Cela permet le trafic EFA au sein du cluster.
  - Pour plus d'informations, consultez Elastic Fabric Adapter pour AI/ML les charges de travail HPC sur Amazon EC2 dans le guide de l'utilisateur d'Amazon Elastic Compute Cloud.

Pour le journal complet des `1.33` modifications de Kubernetes, voir -1.33.md https://github.com/kubernetes/ kubernetes/blob/master/CHANGELOG/CHANGELOG

Kubernetes 1.32

Kubernetes `1.32` est désormais disponible dans Amazon EKS. Pour plus d'informations sur Kubernetes `1.32`, consultez l'annonce officielle de publication.

> **Important**
>
> - La version `flowcontrol.apiserver.k8s.io/v1beta3` API de FlowSchema et PriorityLevelConfiguration a été supprimée dans la version`1.32`. Si vous les utilisez APIs,

vous devez mettre à jour vos configurations pour utiliser la dernière version prise en charge avant de procéder à la mise à niveau.

- ServiceAccount `metadata.annotations[kubernetes.io/enforce-mountable-secrets]`a été obsolète dans sa version `1.32` et sera supprimée dans une future version mineure de Kubernetes. Il est recommandé d'utiliser des espaces de noms distincts pour isoler l'accès aux secrets montés.
- La version Kubernetes `1.32` est la dernière version pour laquelle Amazon EKS publiera Amazon Linux 2 (). AL2 AMIs `1.33`À partir de la version, Amazon EKS continuera à publier des versions basées sur Amazon Linux 2023 (AL2023) et Bottlerocket. AMIs

- La fonctionnalité Memory Manager est désormais disponible pour tous (GA) dans la version `1.32` de Kubernetes. Cette amélioration offre une allocation de mémoire plus efficace et plus prévisible pour les applications conteneurisées, ce qui est particulièrement avantageux pour les charges de travail ayant des exigences spécifiques en matière de mémoire.
- PersistentVolumeClaims (PVCs) créé à StatefulSets ce jour inclut une fonctionnalité de nettoyage automatique. Lorsqu'ils ne PVCs sont plus nécessaires, ils sont automatiquement supprimés tout en maintenant la persistance des données pendant les StatefulSet mises à jour et les opérations de maintenance des nœuds. Cette fonctionnalité simplifie la gestion du stockage et permet d'éviter les orphelins PVCs dans votre cluster.
- La fonctionnalité Custom Resource Field Selector a été introduite, permettant aux développeurs d'ajouter des sélecteurs de champs aux ressources personnalisées. Cette fonctionnalité offre les mêmes capacités de filtrage que celles disponibles pour les objets Kubernetes intégrés aux ressources personnalisées, ce qui permet un filtrage plus précis et plus efficace des ressources et favorise de meilleures pratiques de conception d'API.

Pour le journal des `1.32` modifications complet de Kubernetes, voir -1.32.md https://github.com/kubernetes/ kubernetes/blob/master/CHANGELOG/CHANGELOG

Modifications de l'authentification anonyme

À partir d'Amazon EKS `1.32`, l'authentification anonyme est limitée aux points de terminaison de surveillance de l'état du serveur API suivants :

- `/healthz`
- `/livez`

- `/readyz`

Les demandes adressées à tout autre point de terminaison utilisant l'utilisateur `system:unauthenticated` recevront une réponse HTTP `401 Unauthorized`. Cette amélioration de la sécurité permet d'éviter tout accès involontaire au cluster qui pourrait se produire en raison de politiques RBAC mal configurées.

> **Note**
>
> Le rôle RBAC `public-info-viewer` continue de s'appliquer aux points de terminaison de surveillance de l'état répertoriés ci-dessus.

Obsolescence de l'AMI Amazon Linux 2

La version Kubernetes `1.32` est la dernière version publiée par Amazon EKS. AL2 AMIs `1.33`À partir de la version, Amazon EKS continuera à publier AL2023 et sera basé sur Bottlerocket. AMIs Pour de plus amples informations, veuillez consulter the section called "AL2 Obsolète de l'AMI".

Examiner les notes de mise à jour pour les versions de Kubernetes bénéficiant d'un support étendu

Amazon EKS prend en charge les versions de Kubernetes plus longtemps qu'elles ne sont prises en charge en amont, avec un support standard pour les versions mineures de Kubernetes pendant 14 mois à compter de leur publication dans Amazon EKS, et un support étendu pour les versions mineures de Kubernetes pendant 12 mois supplémentaires (26 mois au total par version).

Cette rubrique présente les modifications importantes à prendre en compte pour chaque version de Kubernetes bénéficiant d'un support étendu. Lors de la mise à niveau, examinez attentivement les modifications apportées entre l'ancienne et la nouvelle version de votre cluster.

Kubernetes 1.31

Kubernetes `1.31` est désormais disponible dans Amazon EKS. Pour plus d'informations sur Kubernetes `1.31`, consultez l'annonce officielle de publication.

> **Important**
>
> - L'indicateur kubelet `--keep-terminated-pod-volumes`, obsolète depuis 2017, a été supprimé dans le cadre de la version `1.31` publiée. Cette modification a un impact sur la manière dont les volumes de pods résiliés sont gérés par le kubelet. Si vous utilisez cet indicateur dans vos configurations de nœuds, vous devez mettre à jour vos scripts de démarrage et vos modèles de lancement afin de le supprimer avant de mettre à niveau.

- La fonctionnalité bêta `VolumeAttributesClass` et la ressource API sont activées dans Amazon EKS version `1.31`. Cette fonctionnalité permet aux opérateurs de clusters de modifier les propriétés mutables des volumes persistants (PVs) gérés par des pilotes CSI compatibles, notamment le pilote Amazon EBS CSI. Pour tirer parti de cette fonctionnalité, veuillez vous assurer que votre pilote CSI prend en charge la fonctionnalité `VolumeAttributesClass` (pour le pilote CSI Amazon EBS, veuillez mettre à niveau vers la version `1.35.0` ou ultérieure afin d'activer automatiquement la fonctionnalité). Vous pourrez créer `VolumeAttributesClass` des objets pour définir les attributs de volume souhaités, tels que le type de volume et le débit, et les associer à vos demandes de volume persistantes (PVCs). Pour plus d'informations, consultez la documentation officielle de Kubernetes ainsi que la documentation de votre pilote CSI.
  - Pour plus d'informations sur le pilote CSI Amazon EBS, consultez the section called "Amazon EBS".
- Le support de Kubernetes pour est AppArmordevenu stable et est désormais généralement disponible pour un usage public. Cette fonctionnalité vous permet de protéger vos conteneurs AppArmor en définissant le `appArmorProfile.type` champ dans le conteneur`securityContext`. Avant la version de Kubernetes`1.30`, il AppArmor était contrôlé par des annotations. À partir de la version `1.30`, il est contrôlé à l'aide de champs. Pour tirer parti de cette fonctionnalité, nous vous recommandons de migrer depuis les annotations et d'utiliser le champ `appArmorProfile.type` afin de garantir la compatibilité de vos charges de travail.
- La fonctionnalité de transition de PersistentVolume dernière phase est devenue stable et est désormais généralement disponible pour une utilisation publique dans la version Kubernetes. `1.31` Cette fonctionnalité introduit un nouveau champ, dans le `.status.lastTransitionTime` PersistentVolumeStatus, qui fournit un horodatage indiquant la date à laquelle une PersistentVolume dernière phase est passée à une phase différente. Cette amélioration permet un meilleur suivi et une meilleure gestion PersistentVolumes, en particulier dans les scénarios où il est important de comprendre le cycle de vie des volumes.

Pour le journal complet des `1.31` modifications de Kubernetes, voir -1.31.md https://github.com/kubernetes/ kubernetes/blob/master/CHANGELOG/CHANGELOG

Kubernetes 1.30

Kubernetes `1.30` est désormais disponible dans Amazon EKS. Pour plus d'informations sur Kubernetes `1.30`, consultez l'annonce officielle de publication.

- À partir de la version Amazon EKS `1.30` ou d'une version plus récente, tous les groupes de nœuds gérés nouvellement créés utiliseront automatiquement par défaut Amazon Linux 2023 (AL2023) comme système d'exploitation des nœuds. Pour plus d'informations sur la spécification du système d'exploitation d'un groupe de nœuds gérés, consultez the section called "Créer".
- Avec Amazon EKS `1.30`, l'étiquette `topology.k8s.aws/zone-id` est ajoutée aux composants master. Vous pouvez utiliser la zone de disponibilité IDs (AZ IDs) pour déterminer l'emplacement des ressources d'un compte par rapport aux ressources d'un autre compte. Pour plus d'informations, consultez la section Zone de disponibilité IDs de vos AWS ressources dans le Guide de l'utilisateur de la AWS RAM.
- À partir de la version `1.30`, Amazon EKS n'inclut plus l'annotation `default` sur la ressource `gp2 StorageClass` appliquée aux clusters nouvellement créés. Cela n'a aucune incidence si vous référencez cette classe de stockage par son nom. Vous devez prendre des mesures si vous comptiez sur la présence d'une `StorageClass` par défaut dans le cluster. Vous devez référencer la `StorageClass` par le nom `gp2`. Vous pouvez également déployer la classe de stockage par défaut recommandée par Amazon EBS en définissant le paramètre `defaultStorageClass.enabled` sur true lors de l'installation de la version `1.31.0` ou ultérieure du `aws-ebs-csi-driver add-on`.
- La politique IAM minimale requise pour le rôle IAM du cluster Amazon EKS a été modifiée. L'action `ec2:DescribeAvailabilityZones` est requise. Pour de plus amples informations, veuillez consulter the section called "Rôle IAM du cluster".

Pour le journal complet des `1.30` modifications de Kubernetes, consultez -1.30.md. https://github.com/kubernetes/ kubernetes/blob/master/CHANGELOG/CHANGELOG

Kubernetes 1.29

Kubernetes `1.29` est désormais disponible dans Amazon EKS. Pour plus d'informations sur Kubernetes `1.29`, consultez l'annonce officielle de publication.

> **Important**
>
> - Les versions API obsolètes `flowcontrol.apiserver.k8s.io/v1beta2` de `FlowSchema` et `PriorityLevelConfiguration` ne sont plus prises en charge dans la version `1.29` de Kubernetes. Si vous disposez de manifestes ou de logiciels clients qui utilisent le groupe d'API bêta obsolète, il est recommandé de les modifier avant de mettre à niveau vers la version `1.29`.

- Le champ `.status.kubeProxyVersion` pour les objets nœud est désormais obsolète, et le projet Kubernetes propose de le supprimer dans une prochaine version. Le champ obsolète n'est pas précis et a toujours été géré par `kubelet`, qui ne connaît pas réellement la version de `kube-proxy`, ni même si `kube-proxy` est en cours d'exécution. Si vous avez utilisé ce champ dans un logiciel client, veuillez cesser de le faire, car les informations ne sont pas fiables et le champ est désormais obsolète.
- Dans Kubernetes `1.29`, afin de réduire la surface d'attaque potentielle, la fonctionnalité `LegacyServiceAccountTokenCleanUp` marque les jetons hérités générés automatiquement et basés sur des secrets comme non valides s'ils n'ont pas été utilisés depuis longtemps (1 an par défaut), et les supprime automatiquement si aucune tentative d'utilisation n'est effectuée pendant une longue période après leur marquage comme non valides (1 année supplémentaire par défaut). Pour identifier ces jetons, vous pouvez exécuter :

```
kubectl get cm kube-apiserver-legacy-service-account-token-tracking -n kube-system
```

Pour le journal des `1.29` modifications complet de Kubernetes, consultez -1.29.md# 1280. https:// github.com/kubernetes/ kubernetes/blob/master/CHANGELOG/CHANGELOG changelog-since-v

Afficher la période de support actuelle du cluster

La section Période de support du cluster de la console AWS indique si votre cluster bénéficie actuellement d'un support standard ou étendu. Si la période de prise en charge de votre cluster est Support étendu, vous êtes facturé pour le support étendu EKS.

Pour plus d'informations sur le support standard et étendu, consultez Comprendre le cycle de vie des versions Kubernetes sur EKS.

1. Accédez à la page Clusters dans la section EKS de la console AWS. Vérifiez que la console est configurée sur la même région AWS que le cluster que vous voulez examiner.
2. Vérifiez la colonne Période de support. Si la valeur est Support standard jusqu'au…, vous n'êtes actuellement pas facturé pour le support étendu. Vous êtes dans la période de support standard. Si la valeur est Support étendu…, ce cluster est actuellement facturé pour le support étendu.

> **Note**
>
> La Période de support ne peut pas être récupérée avec l'API ou l'interface CLI AWS.

Consulter la politique actuelle de mise à niveau du cluster

La politique de mise à niveau du cluster détermine ce qu'il advient de votre cluster lorsqu'il quitte la période de support standard. Si votre politique de mise à niveau est `EXTENDED`, le cluster ne sera pas automatiquement mis à niveau et entrera en support étendu. Si votre politique de mise à niveau est `STANDARD`, il sera automatiquement mis à niveau.

Les contrôles Amazon EKS pour la politique de version Kubernetes vous permettent de choisir le comportement de fin de support standard pour vos clusters EKS. Grâce à ces contrôles, vous pouvez décider quels clusters doivent entrer en support étendu et quels clusters doivent être automatiquement mis à niveau à la fin du support standard pour une version Kubernetes.

Une version mineure bénéficie d'un support standard dans Amazon EKS pendant les 14 premiers mois suivant sa publication. Une fois la date de fin du support standard passée, la version bénéficie d'un support étendu pendant les 12 mois suivants. Le support étendu vous permet de conserver une version spécifique de Kubernetes plus longtemps, moyennant un coût supplémentaire par heure de cluster. Vous pouvez activer ou désactiver le support étendu pour un cluster EKS. Si vous désactivez le support étendu, AWS mettra automatiquement à niveau votre cluster vers la version suivante à la fin du support standard. Si vous activez le support étendu, vous pouvez conserver la version actuelle moyennant un coût supplémentaire pendant une période limitée. Prévoyez de mettre à niveau régulièrement votre cluster Kubernetes, même si vous utilisez le support étendu.

Vous pouvez définir la politique de version pour les clusters nouveaux et existants à l'aide de la propriété `supportType`. Deux options sont disponibles pour définir la politique de prise en charge des versions :

- **`STANDARD`** : votre cluster EKS est éligible à une mise à niveau automatique à la fin du support standard. Ce paramètre n'entraîne pas de frais de support étendu, mais votre cluster EKS sera automatiquement mis à niveau vers la prochaine version Kubernetes prise en charge dans le cadre du support standard.
- **`EXTENDED`** : votre cluster EKS bénéficiera d'un support étendu une fois que la version Kubernetes aura atteint la fin du support standard. Ce paramètre entraîne des frais de support étendu. Vous pouvez mettre à niveau votre cluster vers une version Kubernetes prise en charge standard afin de ne plus avoir à payer de frais de support étendu. Les clusters fonctionnant sous support étendu pourront bénéficier d'une mise à niveau automatique à la fin du support étendu.

Le support étendu est activé par défaut pour les nouveaux clusters et les clusters existants. Vous pouvez vérifier si le support étendu est activé pour un cluster dans la AWS Management Console ou à l'aide de l'AWS CLI.

> **Important**
>
> Si vous souhaitez que votre cluster conserve sa version actuelle de Kubernetes afin de bénéficier de la période de support étendue, vous devez activer la politique de mise à niveau du support étendu avant la fin de la période de support standard.

Vous ne pouvez définir la politique de prise en charge des versions pour vos clusters que lorsqu'ils fonctionnent sous une version de Kubernetes bénéficiant d'un support standard. Une fois que la version entre en support étendu, vous ne pourrez plus modifier ce paramètre tant que vous n'utiliserez pas une version bénéficiant d'un support standard.

Par exemple, si vous avez défini votre politique de prise en charge des versions sur `standard`, vous ne pourrez plus modifier ce paramètre une fois que la version de Kubernetes exécutée sur votre cluster aura atteint la fin du support standard. Si vous avez défini votre politique de prise en charge des versions comme `extended`, vous ne pourrez pas modifier ce paramètre une fois que la version Kubernetes exécutée sur votre cluster aura atteint la fin de la prise en charge standard. Pour modifier le paramètre de politique de prise en charge des versions, votre cluster doit être exécuté sur une version Kubernetes prise en charge standard.

Afficher la politique de mise à niveau du cluster (console AWS)

1. Accédez à la page Clusters dans la section EKS de la console AWS. Vérifiez que la console est configurée sur la même région AWS que le cluster que vous voulez examiner.

2. Consultez la colonne Politique de mise à niveau. Si la valeur est Support standard, votre cluster ne bénéficiera pas du support étendu. Si la valeur est Support étendu, votre cluster bénéficiera du support étendu.

Afficher la politique de mise à niveau du cluster (AWS CLI)

1. Vérifiez que l'AWS CLI est installée et que vous êtes connecté. Découvrez comment mettre à jour et installer l'interface AWS CLI.
2. Déterminez le nom de votre cluster EKS. Définissez la CLI sur la même région AWS que votre cluster EKS.
3. Exécutez la commande suivante :

```
aws eks describe-cluster \
--name <cluster-name> \
--query "cluster.upgradePolicy.supportType"
```

4. Si la valeur est `STANDARD`, votre cluster ne bénéficiera pas du support étendu. Si la valeur est `EXTENDED`, votre cluster bénéficiera du support étendu.

Ajoutez de la flexibilité à la planification des mises à niveau des versions de Kubernetes en activant la prise en charge étendue d'EKS

Cette rubrique décrit comment définir la stratégie de mise à niveau d'un cluster EKS afin d'activer le support étendu. La politique de mise à niveau d'un cluster EKS détermine ce qui se passe lorsqu'un cluster atteint la fin de la période de support standard. Si une politique de mise à niveau de cluster dispose d'une assistance prolongée activée, elle entrera dans la période d'assistance prolongée à la fin de la période d'assistance standard. Le cluster ne sera pas automatiquement mis à niveau à la fin de la période de support standard.

Les clusters qui se trouvent actuellement en période de support étendu entraînent des coûts plus élevés. Si un cluster dispose uniquement d'une politique de mise à niveau configurée pour activer le support étendu et se trouve par ailleurs dans la période de support standard, il engendre des coûts standard.

Si vous créez un cluster dans la console AWS, la politique de mise à niveau sera configurée pour désactiver le support étendu. Si vous créez un cluster d'une autre manière, la politique de mise à

niveau sera définie pour permettre un support étendu. Par exemple, les clusters créés avec l'API AWS bénéficient d'une prise en charge étendue.

Pour plus d'informations sur les politiques de mise à niveau, consultez la section Politique de mise à niveau du cluster.

> **Important**
>
> Si vous souhaitez que votre cluster conserve sa version actuelle de Kubernetes afin de bénéficier de la période de support étendue, vous devez activer la politique de mise à niveau du support étendu avant la fin de la période de support standard.
> Si vous n'activez pas le support étendu, votre cluster sera automatiquement mis à niveau.

Activer le support étendu EKS (console AWS)

1. Accédez à votre cluster EKS dans la console AWS. Sélectionnez l'onglet Aperçu sur la page Informations sur le cluster.
2. Dans la section Paramètres de la version Kubernetes, sélectionnez Gérer.
3. Sélectionnez Support étendu, puis Enregistrer les modifications.

Activer le support étendu EKS (CLI AWS)

1. Vérifiez que l'AWS CLI est installée et que vous êtes connecté. Découvrez comment mettre à jour et installer l'interface CLI AWS.
2. Déterminez le nom de votre cluster EKS.
3. Exécutez la commande suivante :

```
aws eks update-cluster-config \
--name <cluster-name> \
--upgrade-policy supportType=EXTENDED
```

Empêchez l'augmentation des coûts des clusters en désactivant le support étendu d'EKS

Cette rubrique décrit comment définir la politique de mise à niveau d'un cluster EKS pour désactiver le support étendu. La politique de mise à niveau d'un cluster EKS détermine ce qui se passe lorsqu'un cluster atteint la fin de la période de support standard. Si une politique de mise à niveau de cluster a désactivé la prise en charge étendue, elle sera automatiquement mise à niveau vers la version suivante de Kubernetes.

Pour plus d'informations sur les politiques de mise à niveau, consultez la section Politique de mise à niveau du cluster.

> **Important**
>
> Vous ne pouvez pas désactiver le support étendu une fois que votre cluster y est entré.
> Vous ne pouvez désactiver le support étendu pour les clusters que dans le cadre du support standard.
> AWS recommande de mettre à niveau votre cluster vers une version pendant la période de support standard.

Désactiver le support étendu EKS (console AWS)

1. Accédez à votre cluster EKS dans la console AWS. Sélectionnez l'onglet Aperçu sur la page Informations sur le cluster.
2. Dans la section de configuration de la version Kubernetes, sélectionnez Gérer.
3. Sélectionnez Assistance standard, puis Enregistrer les modifications.

Désactiver le support étendu EKS (CLI AWS)

1. Vérifiez que l'AWS CLI est installée et que vous êtes connecté. Découvrez comment mettre à jour et installer l'interface CLI AWS.
2. Déterminez le nom de votre cluster EKS.
3. Exécutez la commande suivante :

```
aws eks update-cluster-config \
```

```
--name <cluster-name> \
--upgrade-policy supportType=STANDARD
```

Afficher les versions de plateforme Amazon EKS pour chaque version de Kubernetes

Les versions de la plateforme Amazon EKS représentent les fonctionnalités du plan de contrôle du cluster Amazon EKS, telles que les indicateurs de serveur d'API Kubernetes qui sont activés, ainsi que la version de correctif Kubernetes actuelle. Une ou plusieurs versions de plateforme Amazon EKS sont associées à chaque version mineure de Kubernetes. Les versions de plateforme pour les différentes versions mineures de Kubernetes sont indépendantes. Vous pouvez récupérer la version actuelle de la plate-forme de votre cluster à l'aide de la AWS CLI ou AWS Management Console. Si vous avez un cluster local sur AWS Outposts, consultez the section called “Versions de la plateforme EKS” plutôt cette rubrique.

Lorsqu'une nouvelle version mineure de Kubernetes est disponible dans Amazon EKS, telle que la version 1.33, la version initiale de la plateforme Amazon EKS pour cette version mineure de Kubernetes commence par `eks.1`. Cependant, Amazon EKS publie régulièrement de nouvelles versions de plateforme pour activer de nouveaux paramètres de plan de contrôle Kubernetes et fournir des correctifs de sécurité.

Lorsque de nouvelles versions de plateforme Amazon EKS deviennent disponibles pour une version mineure :

- Le numéro de version de plateforme Amazon EKS est incrémenté (`eks.<n+1>`).
- Amazon EKS met automatiquement à niveau tous les clusters existants vers la dernière version de plateforme Amazon EKS pour les versions mineures Kubernetes correspondantes. Les mises à niveau automatiques de versions de plateforme Amazon EKS existantes sont déployées de façon incrémentielle. Le processus de déploiement peut prendre du temps. Si vous avez besoin immédiatement des fonctions de la dernière version de plateforme Amazon EKS, vous devez créer un nouveau cluster Amazon EKS.

  Si votre cluster a plus de deux versions de plateforme de retard par rapport à la version actuelle, il est possible qu'Amazon EKS n'ait pas pu mettre à jour automatiquement votre cluster. Pour plus de détails sur la cause de ce problème, consultez the section called “La version de la plateforme Amazon EKS est inférieure de plus de deux versions à la version actuelle de la plateforme”.

- Amazon EKS peut publier une nouvelle AMI de nœud avec une version de correctif correspondante. Cependant, toutes les versions de correctif sont compatibles entre le plan de contrôle et le nœud EKS AMIs pour une version mineure de Kubernetes donnée.

Les nouvelles versions de plateforme Amazon EKS n'introduisent pas de changements majeurs et ne provoquent pas d'interruptions de service.

Les clusters sont toujours créés avec la dernière version de plateforme Amazon EKS disponible (`eks.<n>`) pour la version Kubernetes spécifiée. Si vous mettez à jour votre cluster vers une nouvelle version mineure de Kubernetes, votre cluster reçoit la version de plateforme Amazon EKS actuelle pour la version mineure Kubernetes vers laquelle vous avez effectué la mise à jour.

Les versions actuelles et récentes de plateforme Amazon EKS sont décrites dans les tableaux ci-dessous.

> **Note**
>
> AWS a récemment désactivé certaines versions de la plateforme publiées en juin 2024. Les versions de plateforme présentaient des problèmes de stabilité. Aucune action n'est nécessaire.

Pour recevoir des notifications concernant toutes les modifications apportées aux fichiers sources de cette page de documentation spécifique, vous pouvez vous abonner à l'URL suivante à l'aide d'un lecteur RSS :

```
https://github.com/awsdocs/amazon-eks-user-guide/commits/mainline/latest/ug/versioning/platform-versions.adoc.atom
```

Version de Kubernetes **`1.35`**

Les contrôleurs d'admission suivants sont activés pour toutes les plateformes version `1.35` :`NodeRestriction`,`ExtendedResourceToleration`,`NamespaceLifecycle`,`LimitRang`

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
1.35.0	eks.3	Version initiale de Kubernetes 1.35 pour EKS. Pour de plus amples informations, veuillez consulter the section called "Kubernetes 1.35".	27 janvier 2026

Version de Kubernetes **1.34**

Les contrôleurs d'admission suivants sont activés pour toutes les plateformes version 1.34 :NodeRestriction,ExtendedResourceToleration,NamespaceLifecycle,LimitRang

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
1.34.3	eks.13	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	28 janvier 2026
1.34.3	eks.12	Nouvelle version de la plate-forme avec des correctifs de sécurité et des améliorations. La version 1.34 eks.11 a été supprimée en interne et n'a jamais été publiée.	16 janvier 2026

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
1.34.2	eks.10	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	19 décembre 2025
1.34.1	eks.9	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	18 novembre 2025
1.34.1	eks.8	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	17 novembre 2025
1.34.1	eks.7	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	6 novembre 2025
1.34.1	eks.6	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	30 octobre 2025
1.34.1	eks.5	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	15 octobre 2025

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
1.34.1	eks.4	Version initiale de Kubernetes 1.34 pour EKS. Pour de plus amples informations, veuillez consulter the section called "Kubernetes 1.34".	2 octobre 2025

Version de Kubernetes **1.33**

Les contrôleurs d'admission suivants sont activés pour toutes les plateformes version 1.33 :NodeRestriction,ExtendedResourceToleration,NamespaceLifecycle,LimitRang

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
1.33.7	eks.27	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	28 janvier 2026
1.33.7	eks.26	Nouvelle version de la plate-forme avec des correctifs de sécurité et des améliorations. La version 1.33 eks.25 a été supprimée en interne et n'a jamais été publiée.	16 janvier 2026
1.33.6	eks.24	Nouvelle version de la plateforme	19 décembre 2025

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
		avec des corrections de sécurité et des améliorations.	
1.33.5	eks.23	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	18 novembre 2025
1.33.5	eks.22	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	17 novembre 2025
1.33.5	eks.21	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	6 novembre 2025
1.33.5	eks.20	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	30 octobre 2025
1.33.5	eks.19	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	28 octobre 2025

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
1.33.5	eks.18	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	15 octobre 2025
1.33.5	eks.17	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	15 octobre 2025
1.33.5	eks.16	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	2 octobre 2025
1.33.4	eks.15	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	29 septembre 2025
1.33.4	eks.14	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	29 septembre 2025
1.33.4	eks.13	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	16 septembre 2025

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
1.33.4	eks.12	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	5 septembre 2025
1.33.3	eks.11	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	25 août 2025
1.33.3	eks.10	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	17 août 2025
1.33.2	eks.9	Nouvelle version de plateforme avec corrections de sécurité et améliorations. Les versions 1.33 eks.7 et 1.33 eks.8 ont été supprimées en interne et n'ont jamais été publiées.	30 juillet 2025
1.33.1	eks.6	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	26 juin 2025

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
1.33.1	eks.5	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	11 juin 2025
1.33.1	eks.4	Version initiale de Kubernetes 1.33 pour EKS. Pour de plus amples informations, veuillez consulter the section called "Kubernetes 1.33".	28 mai 2025

Version de Kubernetes **1.32**

Les contrôleurs d'admission suivants sont activés pour toutes les plateformes version 1.32 :NodeRestriction,ExtendedResourceToleration,NamespaceLifecycle,LimitRang

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
1.32.11	eks.34	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	28 janvier 2026
1.32.11	eks.33	Nouvelle version de la plate-forme avec des correctifs de sécurité et des améliorations. La version 1.32 eks.32	16 janvier 2026

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
		a été supprimée en interne et n'a jamais été publiée.	
1.32.10	eks.31	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	19 décembre 2025
1.32.9	eks.30	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	18 novembre 2025
1.32.9	eks.29	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	17 novembre 2025
1.32.9	eks.28	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	6 novembre 2025
1.32.9	eks.27	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	30 octobre 2025

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
1.32.9	eks.26	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	28 octobre 2025
1.32.9	eks.25	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	15 octobre 2025
1.32.9	eks.24	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	15 octobre 2025
1.32.9	eks.23	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	2 octobre 2025
1.32.8	eks.22	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	29 septembre 2025
1.32.8	eks.21	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	29 septembre 2025

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
1.32.8	eks.20	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	16 septembre 2025
1.32.8	eks.19	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	5 septembre 2025
1.32.7	eks.18	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	25 août 2025
1.32.7	eks.17	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	17 août 2025
1.32.6	eks.16	Nouvelle version de plateforme avec corrections de sécurité et améliorations. Les versions 1.32 eks.14 et 1.32 eks.15 ont été supprimées en interne et n'ont jamais été publiées.	30 juillet 2025

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
1.32.5	eks.13	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	26 juin 2025
1.32.5	eks.12	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	11 juin 2025
1.32.5	eks.11	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	30 mai 2025
1.32.3	eks.10	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	16 mai 2025
1.32.3	eks.9	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	29 avril 2025
1.32.3	eks.8	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	18 avril 2025

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
1.32.3	eks.7	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	18 avril 2025
1.32.3	eks.6	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	2 avril 2025
1.32.2	eks.5	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	17 mars 2025
1.32.2	eks.4	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	4 mars 2025
1.32.1	eks.3	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	24 février 2025

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
1.32.0	eks.2	Version initiale de Kubernetes 1.32 pour EKS. Pour de plus amples informations, veuillez consulter the section called "Kubernetes 1.32".	Janvier 2025

Version de Kubernetes **1.31**

Les contrôleurs d'admission suivants sont activés pour toutes les plateformes de version 1.31 : NodeRestriction, ExtendedResourceToleration, NamespaceLifecycle, LimitRanger, ServiceAccount, TaintNodesByCondition, PodSecurity, Priority, DefaultTolerationSeconds, DefaultStorageClass, StorageObjectInUseProtection, PersistentVolumeClaimResize, RuntimeClass, CertificateApproval, CertificateSigning, CertificateSubjectRestriction, DefaultIngressClass, MutatingAdmissionWebhook, ValidatingAdmissionWebhook, ResourceQuota, ObjectCount.

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
1.31.14	eks.50	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	28 janvier 2026
1.31.14	eks.49	Nouvelle version de la plate-forme avec des correctifs de sécurité et des améliorations. La	16 janvier 2026

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
		version 1.31 eks.48 a été supprimée en interne et n'a jamais été publiée.	
1.31.14	eks.47	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	19 décembre 2025
1.31.13	eks.46	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	18 novembre 2025
1.31.13	eks.45	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	17 novembre 2025
1.31.13	eks.44	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	6 novembre 2025
1.31.13	eks.43	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	30 octobre 2025

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
1.31.13	eks.42	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	28 octobre 2025
1.31.13	eks.41	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	15 octobre 2025
1.31.13	eks.40	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	15 octobre 2025
1.31.13	eks.39	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	2 octobre 2025
1.31.12	eks.38	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	29 septembre 2025
1.31.12	eks.37	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	29 septembre 2025

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
1.31.12	eks.36	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	16 septembre 2025
1.31.12	eks.35	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	5 septembre 2025
1.31.11	eks.34	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	25 août 2025
1.31.11	eks.33	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	17 août 2025
1.31.10	eks.32	Nouvelle version de plateforme avec corrections de sécurité et améliorations. Les versions 1.31 eks.30 et 1.31 eks.31 ont été supprimées en interne et n'ont jamais été publiées.	30 juillet 2025

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
1.31.9	eks.29	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	26 juin 2025
1.31.9	eks.28	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	11 juin 2025
1.31.9	eks.27	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	30 mai 2025
1.31.7	eks.26	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	16 mai 2025
1.31.7	eks.25	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	29 avril 2025
1.31.7	eks.24	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	18 avril 2025

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
1.31.7	eks.23	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	18 avril 2025
1.31.7	eks.22	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	2 avril 2025
1.31.6	eks.21	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	17 mars 2025
1.31.6	eks.20	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	4 mars 2025
1.31.5	eks.19	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	24 février 2025
1.31.5	eks.18	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	24 février 2025

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
1.31.4	eks.17	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	17 janvier 2025
1.31.2	eks.12	Nouvelle version de plateforme avec prise en charge des nœuds hybrides Amazon EKS et améliorations de l'observabilité du plan de contrôle. Veuillez consulter the section called "Nœuds hybrides" et Amazon EKS améliore l'observabilité des performances, respectivement.	15 novembre 2024
1.31.1	eks.6	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	21 octobre 2024
1.31.0	eks.4	Version initiale de Kubernetes 1.31 pour EKS. Pour de plus amples informations, veuillez consulter the section called "Kubernetes 1.31".	26 septembre 2024

Version de Kubernetes **1.30**

Les contrôleurs d'admission suivants sont activés pour toutes les plateformes version 1.30 :NodeRestriction,ExtendedResourceToleration,NamespaceLifecycle,LimitRang

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
1.30.14	eks.58	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	28 janvier 2026
1.30.14	eks.57	Nouvelle version de la plate-forme avec des correctifs de sécurité et des améliorations. La version 1.30 eks.56 a été supprimée en interne et n'a jamais été publiée.	16 janvier 2026
1.30.14	eks.55	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	19 décembre 2025
1.30.14	eks.54	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	18 novembre 2025
1.30.14	eks.53	Nouvelle version de la plateforme	17 novembre 2025

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
		avec des corrections de sécurité et des améliorations.	
1.30.14	eks.52	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	6 novembre 2025
1.30.14	eks.51	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	30 octobre 2025
1.30.14	eks.50	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	28 octobre 2025
1.30.14	eks.49	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	15 octobre 2025
1.30.14	eks.48	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	15 octobre 2025

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
1.30.14	eks.47	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	2 octobre 2025
1.30.14	eks.46	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	29 septembre 2025
1.30.14	eks.45	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	29 septembre 2025
1.30.14	eks.44	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	16 septembre 2025
1.30.14	eks.43	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	5 septembre 2025
1.30.14	eks.42	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	25 août 2025

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
1.30.14	eks.41	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	17 août 2025
1.30.14	eks.40	Nouvelle version de plateforme avec corrections de sécurité et améliorations. Les versions 1.30 eks.38 et 1.30 eks.39 ont été supprimées en interne et n'ont jamais été publiées.	30 juillet 2025
1.30.13	eks.37	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	26 juin 2025
1.30.13	eks.36	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	11 juin 2025
1.30.13	eks.35	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	30 mai 2025

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
1.30.11	eks.34	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	16 mai 2025
1.30.11	eks.33	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	29 avril 2025
1.30.11	eks.32	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	18 avril 2025
1.30.11	eks.31	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	18 avril 2025
1.30.11	eks.30	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	2 avril 2025
1.30.10	eks.29	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	17 mars 2025

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
1.30.10	eks.28	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	4 mars 2025
1.30.9	eks.27	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	24 février 2025
1.30.8	eks.25	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	17 janvier 2025
1.30.8	eks.24	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	3 janvier 2025
1.30.7	eks.23	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	13 décembre 2024
1.30.6	eks.22	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	13 décembre 2024

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
1.30.6	eks.21	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	13 décembre 2024
1.30.6	eks.20	Nouvelle version de plateforme avec prise en charge des nœuds hybrides Amazon EKS et améliorations de l'observabilité du plan de contrôle. Veuillez consulter the section called "Nœuds hybrides" et Amazon EKS améliore l'observabilité des performances, respectivement.	15 novembre 2024
1.30.5	eks.12	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	21 octobre 2024
1.30.4	eks.8	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	3 septembre 2024

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
1.30.3	eks.7	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	28 août 2024
1.30.3	eks.6	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	9 août 2024
1.30.2	eks.5	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	2 juillet 2024
1.30.0	eks.2	Version initiale de Kubernetes 1.30 pour EKS. Pour de plus amples informations, veuillez consulter the section called "Kubernetes 1.30".	23 mai 2024

Version de Kubernetes **1.29**

Les contrôleurs d'admission suivants sont activés pour toutes les plateformes version 1.29 :NodeRestriction,ExtendedResourceToleration,NamespaceLifecycle,LimitRang

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
1.29.15	eks.61	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	28 janvier 2026
1.29.15	eks.60	Nouvelle version de la plate-forme avec des correctifs de sécurité et des améliorations. La version 1.29 eks.59 a été supprimée en interne et n'a jamais été publiée.	16 janvier 2026
1.29.15	eks.58	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	19 décembre 2025
1.29.15	eks.57	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	18 novembre 2025
1.29.15	eks.56	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	17 novembre 2025

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
1.29.15	eks.55	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	6 novembre 2025
1.29.15	eks.54	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	30 octobre 2025
1.29.15	eks.53	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	28 octobre 2025
1.29.15	eks.52	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	15 octobre 2025
1.29.15	eks.51	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	15 octobre 2025
1.29.15	eks.50	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	2 octobre 2025

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
1.29.15	eks.49	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	29 septembre 2025
1.29.15	eks.48	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	29 septembre 2025
1.29.15	eks.47	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	16 septembre 2025
1.29.15	eks.46	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	5 septembre 2025
1.29.15	eks.45	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	25 août 2025
1.29.15	eks.44	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	17 août 2025

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
1.29.15	eks.43	Nouvelle version de plateforme avec corrections de sécurité et améliorations. Les versions 1.29 eks.41 et 1.29 eks.42 ont été supprimées en interne et n'ont jamais été publiées.	30 juillet 2025
1.29.15	eks.40	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	26 juin 2025
1.29.15	eks.39	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	11 juin 2025
1.29.15	eks.38	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	30 mai 2025
1.29.15	eks.37	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	16 mai 2025

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
1.29.15	eks.36	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	29 avril 2025
1.29.15	eks.35	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	18 avril 2025
1.29.15	eks.34	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	18 avril 2025
1.29.15	eks.33	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	2 avril 2025
1.29.14	eks.32	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	17 mars 2025
1.29.14	eks.31	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	4 mars 2025

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
1.29.13	eks.30	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	24 février 2025
1.29.13	eks.29	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	24 février 2025
1.29.12	eks.28	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	20 janvier 2025
1.29.12	eks.27	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	3 janvier 2025
1.29.11	eks.26	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	13 décembre 2024
1.29.10	eks.25	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	13 décembre 2024

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
1.29.10	eks.24	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	13 décembre 2024
1.29.10	eks.23	Nouvelle version de plateforme avec prise en charge des nœuds hybrides Amazon EKS et améliorations de l'observabilité du plan de contrôle. Veuillez consulter the section called "Nœuds hybrides" et Amazon EKS améliore l'observabilité des performances, respectivement.	15 novembre 2024
1.29.9	eks.17	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	21 octobre 2024
1.29.8	eks.13	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	3 septembre 2024

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
1.29.7	eks.12	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	28 août 2024
1.29.7	eks.11	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	9 août 2024
1.29.6	eks.10	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	2 juillet 2024
1.29.4	eks.7	Nouvelle version de plateforme avec mise à l'échelle automatique CoreDNS, correctifs de sécurité et améliorations. Pour plus d'informations sur la mise à l'échelle automatique CoreDNS, consultez the section called "Mise à l'échelle pour un trafic élevé".	16 mai 2024

Version de Kubernetes	Version de la plateforme EKS	Notes de mise à jour	Date de publication
1.29.3	eks.6	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	18 avril 2024
1.29.1	eks.5	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	29 mars 2024
1.29.1	eks.4	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	20 mars 2024
1.29.1	eks.3	Nouvelle version de la plateforme avec des corrections de sécurité et des améliorations.	12 mars 2024
1.29.0	eks.1	Version initiale de Kubernetes 1.29 pour EKS. Pour de plus amples informations, veuillez consulter the section called "Kubernetes 1.29".	23 janvier 2024

Obtenir la version actuelle de la plateforme

1. Ouvrez la console Amazon EKS.
2. Dans le panneau de navigation, choisissez Clusters.
3. Dans la liste des clusters, sélectionnez le Nom du cluster dont vous voulez vérifier la version de plateforme.
4. Choisissez l'onglet Overview (Présentation).
5. La Version de plateforme est disponible dans la section Détails.
6. Déterminez le Nom du cluster dont vous voulez vérifier la version de plateforme.
7. Exécutez la commande suivante :

```
aws eks describe-cluster --name my-cluster --query cluster.platformVersion
```

L'exemple qui suit illustre un résultat.

```
"eks.10"
```

Changer la version de plateforme

Vous ne pouvez pas modifier la version de plateforme d'un cluster EKS. Lorsque de nouvelles versions de plateforme Amazon EKS sont disponibles pour une version de Kubernetes, EKS met automatiquement à niveau tous les clusters existants vers la dernière version de plateforme Amazon EKS pour leur version correspondante de Kubernetes. Les mises à niveau automatiques de versions de plateforme Amazon EKS existantes sont déployées de façon incrémentielle. Vous ne pouvez pas utiliser la AWS console ou la CLI pour modifier la version de la plateforme.

Si vous mettez à niveau votre version de Kubernetes, votre cluster passera à la version de plateforme la plus récente pour la version de Kubernetes.

Résolution des problèmes liés aux clusters et aux nœuds Amazon EKS

Ce chapitre traite de certaines erreurs courantes que vous pouvez rencontrer lorsque vous utilisez Amazon EKS, ainsi que des solutions. Si vous avez besoin de dépanner des zones spécifiques d'Amazon EKS, consultez les rubriques the section called "Résolution des problèmes", the section called "Dépannage d'EKS Connector" et Dépannage d'ADOT à l'aide de modules complémentaires EKS.

Pour plus d'informations sur la résolution des problèmes, consultez Contenu du centre de connaissances sur Amazon Elastic Kubernetes Service sur AWS re:Post.

Capacité insuffisante

Si vous recevez l'erreur suivante lorsque vous essayez de créer un cluster Amazon EKS, cela signifie que l'une des zones de disponibilité que vous avez spécifiées ne dispose pas d'une capacité suffisante pour prendre en charge un cluster.

```
Cannot create cluster 'example-cluster' because region-1d, the targeted Availability Zone, does not currently have sufficient capacity to support the cluster. Retry and choose from these Availability Zones: region-1a, region-1b, region-1c
```

Essayez à nouveau de créer votre cluster avec des sous-réseaux de votre VPC de cluster hébergés dans les zones de disponibilité renvoyés par ce message d'erreur.

Il existe des zones de disponibilité dans lesquelles un cluster ne peut pas résider. Veuillez comparer les zones de disponibilité dans lesquelles se trouvent vos sous-réseaux avec la liste des zones de disponibilité dans les Exigences et considérations relatives aux sous-réseaux.

Les nœuds ne parviennent pas à joindre le cluster

Quelques raisons courantes empêchent les nœuds de joindre le cluster :

- Si les nœuds sont des nœuds gérés, Amazon EKS ajoute des entrées à la `ConfigMap` `aws-auth` lorsque vous créez le groupe de nœuds. Si l'entrée a été supprimée ou modifiée, vous devez l'ajouter de nouveau. Pour plus d'informations, entrez `eksctl create iamidentitymapping`

`--help` dans votre terminal. Vous pouvez afficher vos entrées actuelles de `ConfigMap aws-auth` en remplaçant *`my-cluster`* dans la commande suivante par le nom de votre cluster et en exécutant la commande modifiée : `eksctl get iamidentitymapping --cluster` *`my-cluster`* . L'ARN du rôle que vous spécifiez ne peut pas inclure de chemin autre que `/`. Par exemple, si le nom de votre rôle est `development/apps/my-role`, vous devez le remplacer par `my-role` lorsque vous spécifiez l'ARN du rôle. Assurez-vous que vous spécifiez l'ARN pour le rôle IAM du nœud (et non l'ARN du profil d'instance).

Si les nœuds sont autogérés et que vous n'avez pas créé d'entrées d'accès pour l'ARN du rôle IAM du nœud, exécutez les mêmes commandes que celles répertoriées pour les nœuds gérés. Si vous avez créé une entrée d'accès pour l'ARN du rôle IAM de votre nœud, il se peut qu'elle ne soit pas correctement configurée dans l'entrée d'accès. Assurez-vous que l'ARN du rôle IAM du nœud (et non l'ARN du profil d'instance) est spécifié comme ARN principal dans votre entrée de `ConfigMap aws-auth` ou dans votre entrée d'accès. Pour plus d'informations sur les entrées d'accès, consultez the section called "Entrées d'accès".

- ClusterName dans votre modèle AWS CloudFormation de nœud ne correspond pas exactement au nom du cluster auquel vous voulez que vos nœuds se joignent. Si vous transmettez une valeur incorrecte à ce champ, le fichier `/var/lib/kubelet/kubeconfig` du nœud sera mal configuré et les nœuds ne rejoindront pas le cluster.
- Le nœud n'est pas étiqueté comme appartenant au cluster. Vos nœuds doivent être balisés avec la balise suivante, où *`my-cluster`* est remplacé par le nom de votre cluster.

Clé	Valeur
`kubernetes.io/cluster/` *`my-cluster`*	`owned`

- Les nœuds peuvent ne pas être en mesure d'accéder au cluster à l'aide d'une adresse IP publique. Assurez-vous qu'une adresse IP publique est attribuée aux nœuds déployés dans les sous-réseaux publics. Si ce n'est pas le cas, vous pouvez associer une adresse IP Elastic à un nœud après son lancement. Pour plus d'informations, consultez Association d'une adresse IP élastique à une instance en cours d'exécution ou à une interface réseau. Si le sous-réseau public n'est pas défini pour attribuer automatiquement des adresses IP publiques aux instances qui y sont déployées, nous vous recommandons d'activer ce paramètre. Pour plus d'informations, consultez Modification de l'attribut d'adressage IPv4 public de votre sous-réseau. Si le nœud est déployé sur un sous-réseau privé, ce sous-réseau doit disposer d'une route vers une passerelle NAT à laquelle une adresse IP publique est attribuée.

- Le point de terminaison STS AWS pour la région AWS dans laquelle vous déployez les nœuds n'est pas activé pour votre compte. Pour activer la région, consultez Activation et désactivation de STS AWS dans une région AWS.
- Le nœud ne dispose pas d'une entrée DNS privée, ce qui entraîne l'apparition d'une erreur « `kubelet` » dans le journal `node "" not found`. Assurez-vous que le VPC sur lequel le nœud est créé a des valeurs définies pour `domain-name` et `domain-name-servers` comme `Options` dans un `DHCP options set`. Les valeurs par défaut sont `domain-name:<region>.compute.internal` et `domain-name-servers:AmazonProvidedDNS`. Pour en savoir plus, consultez Jeux d'options DHCP dans le Guide de l'utilisateur Amazon VPC.
- Si les nœuds du groupe de nœuds gérés ne se connectent pas au cluster dans les 15 minutes, un problème de santé « NodeCreationFailure » sera émis et le statut de la console sera défini sur « `Create failed` ». Pour les AMI Windows dont le temps de lancement est lent, ce problème peut être résolu à l'aide du fast launch.

Pour identifier et résoudre les problèmes courants qui empêchent les composants master de rejoindre un cluster, vous pouvez utiliser le runbook`AWSSupport-TroubleshootEKSWorkerNode`. Pour plus d'informations, consultez `AWSSupport-TroubleshootEKSWorkerNode` dans Référence du dossier d'exploitation AWS Systems Manager Automation.

Accès non autorisé ou refusé (`kubectl`)

Si vous recevez l'une des erreurs suivantes lors de l'exécution des commandes `kubectl`, cela signifie que `kubectl` n'est pas correctement configuré pour Amazon EKS ou que les informations d'identification du principal IAM (rôle ou utilisateur) que vous utilisez ne correspondent pas à un nom d'utilisateur Kubernetes disposant des autorisations suffisantes pour les objets Kubernetes sur votre cluster Amazon EKS.

- `could not get token: AccessDenied: Access denied`
- `error: You must be logged in to the server (Unauthorized)`
- `error: the server doesn't have a resource type "svc"`

Cela peut être dû à l'une des raisons suivantes :

- Le cluster a été créé avec les informations d'identification d'un principal IAM et `kubectl` est configuré pour utiliser les informations d'identification d'un autre principal IAM. Pour résoudre ce problème, mettez à jour votre fichier `kube config` afin d'utiliser les informations d'identification

à l'origine de la création du cluster. Pour de plus amples informations, consultez the section called "Accès au cluster avec kubectl".

- Si votre cluster répond aux exigences minimales de la plateforme dans la section « Conditions préalables » de la section Accorder aux utilisateurs IAM l'accès à Kubernetes avec des entrées d'accès EKS, aucune entrée d'accès n'existe avec votre principal IAM. Si elle existe, elle ne dispose pas des noms de groupe Kubernetes nécessaires qui lui sont définis ou n'est pas associée à la stratégie d'accès appropriée. Pour de plus amples informations, consultez the section called "Entrées d'accès".
- Si votre cluster ne répond pas aux exigences minimales de la plateforme dans Accorder aux utilisateurs IAM l'accès à Kubernetes avec des entrées d'accès EKS, aucune entrée avec votre principal IAM n'existe dans le `aws-auth` `ConfigMap`. Si elle existe, elle n'est pas mappée aux noms de groupes Kubernetes liés à un `Role` ou un `ClusterRole` Kubernetes avec les autorisations nécessaires. Pour plus d'informations sur les objets d'autorisation basée sur les rôles (RBAC) Kubernetes, consultez Utilisation de l'autorisation RBAC dans la documentation Kubernetes. Vous pouvez afficher vos entrées actuelles de `ConfigMap` `aws-auth` en remplaçant *`my-cluster`* dans la commande suivante par le nom de votre cluster et en exécutant la commande modifiée : `eksctl get iamidentitymapping --cluster` *`my-cluster`* . Si une entrée pour l'ARN de votre principal IAM ne se trouve pas dans le `ConfigMap`, saisissez `eksctl create iamidentitymapping --help` dans votre terminal pour savoir comment en créer une.

Si vous installez et configurez l'AWS CLI, vous pouvez configurer les informations d'identification IAM que vous utilisez. Pour plus d'informations, consultez la section Configuration de l'AWS CLI du Guide de l'utilisateur de l'interface de la ligne de commande AWS. Vous pouvez également configurer `kubectl` pour utiliser un rôle IAM, si vous assumez un rôle IAM pour accéder aux objets Kubernetes sur votre cluster. Pour de plus amples informations, consultez the section called "Accès au cluster avec kubectl".

`hostname doesn't match`

La version Python de votre système doit être `2.7.9` ou ultérieure. Sinon, vous recevrez des erreurs `hostname doesn't match` avec les appels AWS CLI à Amazon EKS. Pour plus d'informations, consultez Que sont les erreurs « nom d'hôte ne correspond pas » ? dans les Questions fréquentes sur les demandes Python.

getsockopt: no route to host

Docker s'exécute dans la plage d'adresses CIDR `172.17.0.0/16` dans les clusters Amazon EKS. Nous vous recommandons de ne pas faire chevaucher cette plage par les sous-réseaux VPC de votre cluster. Sinon, vous recevrez l'erreur suivante :

```
Error: : error upgrading connection: error dialing backend: dial tcp
 172.17.<nn>.<nn>:10250: getsockopt: no route to host
```

Instances failed to join the Kubernetes cluster

Si vous recevez l'erreur `Instances failed to join the Kubernetes cluster` dans la AWS Management Console, assurez-vous que l'accès au point de terminaison privé du cluster est activé ou que vous avez correctement configuré les blocs d'adresse CIDR pour l'accès au point de terminaison public. Pour de plus amples informations, consultez the section called "Accès au point de terminaison de cluster".

Codes d'erreurs liées aux groupes de nœuds gérés

Si votre groupe de nœuds gérés rencontre un problème d'intégrité matérielle, Amazon EKS renvoie un code d'erreur pour vous aider à diagnostiquer le problème. Ces surveillances de l'état ne détectent pas les problèmes logiciels, car elles sont basées sur les surveillances de l'état Amazon EC2. La liste suivante décrit les codes d'erreur.

AccessDenied

> Amazon EKS ou un ou plusieurs de vos nœuds gérés ne parviennent pas à s'authentifier ou à s'autoriser auprès du serveur API de votre cluster Kubernetes. Pour de plus amples informations sur la résolution d'une cause courante, consultez the section called "Correction d'une cause courante d'erreurs `AccessDenied` pour les groupes de nœuds gérés". Les AMI Windows privées peuvent également provoquer ce code d'erreur, accompagné du message d'erreur `Not authorized for images`. Pour de plus amples informations, consultez the section called "`Not authorized for images`".

AmiIdNotFound

> Nous n'avons pas trouvé l'ID AMI associé à votre modèle de lancement. Assurez-vous que l'AMI existe et est partagée avec votre compte.

AutoScalingGroupNotFound

Nous n’avons pas trouvé le groupe Auto Scaling associé au groupe de nœuds gérés. Vous pouvez peut-être recréer un groupe Auto Scaling avec les mêmes paramètres pour effectuer une récupération.

ClusterUnreachable

Amazon EKS ou un ou plusieurs de vos nœuds gérés ne parviennent pas à communiquer avec votre serveur API du cluster Kubernetes. Cela peut se produire s'il y a des interruptions de réseau ou si les serveurs d'API temporisent le traitement des demandes.

Ec2SecurityGroupNotFound

Nous n’avons pas trouvé le groupe de sécurité du cluster. Vous devez recréer votre cluster.

Ec2SecurityGroupDeletionFailure

Nous n'avons pas pu supprimer le groupe de sécurité d'accès à distance pour votre groupe de nœuds gérés. Supprimez toutes les dépendances du groupe de sécurité.

Ec2LaunchTemplateNotFound

Nous n’avons pas trouvé le modèle de lancement Amazon EC2 pour votre groupe de nœuds gérés. Vous devez recréer votre groupe de nœuds pour effectuer une récupération.

Ec2LaunchTemplateVersionMismatch

La version du modèle de lancement Amazon EC2 pour votre groupe de nœuds gérés ne correspond pas à la version créée par Amazon EKS. Vous pouvez peut-être revenir à la version Amazon EKS créée pour effectuer une récupération.

IamInstanceProfileNotFound

Nous n’avons pas trouvé le profil d’instance IAM pour votre groupe de nœuds gérés. Vous pouvez peut-être recréer un profil d'instance avec les mêmes paramètres pour effectuer une récupération.

IamNodeRoleNotFound

Nous n’avons pas trouvé le rôle IAM pour votre groupe de nœuds gérés. Vous pouvez peut-être recréer un rôle IAM avec les mêmes paramètres pour effectuer une récupération.

AsgInstanceLaunchFailures

Votre groupe Auto Scaling rencontre des problèmes lors d'une tentative de lancement d'instances.

NodeCreationFailure

Vos instances lancées ne peuvent pas s'enregistrer auprès de votre cluster Amazon EKS. Les causes courantes de cet échec sont des autorisations de rôle IAM de nœud insuffisantes ou l'absence d'un accès Internet sortant pour les nœuds. Vos nœuds doivent répondre à l'une des exigences suivantes :

- Accès à Internet à l'aide d'une adresse IP publique. Le groupe de sécurité associé au sous-réseau dans lequel se trouve le nœud doit autoriser la communication. Pour plus d'informations, consultez the section called "Exigences et considérations requises pour les sous-réseaux" et the section called "Exigences de groupe de sécurité".
- Vos nœuds et votre VPC doivent répondre aux exigences décrites dans Déployer des clusters privés avec un accès Internet limité.

InstanceLimitExceeded

Votre compte AWS ne peut pas lancer d'autres instances du type d'instance spécifié. Vous pouvez demander une augmentation de la limite d'instances Amazon EC2 pour effectuer une récupération.

InsufficientFreeAddresses

Un ou plusieurs sous-réseaux associés à votre groupe de nœuds gérés ne disposent pas d'adresses IP suffisantes pour les nouveaux nœuds.

InternalFailure

Ces erreurs sont généralement dues à un problème côté serveur Amazon EKS.

Correction d'une cause courante d'erreurs **`AccessDenied`** pour les groupes de nœuds gérés

La cause la plus courante d'erreurs `AccessDenied` lors de la réalisation d'opérations sur des groupes de nœuds gérés est l'absence de `eks:node-manager` `ClusterRole` ou `ClusterRoleBinding`. Amazon EKS configure ces ressources dans votre cluster dans le cadre de l'onboarding avec les groupes de nœuds gérés, et celles-ci sont nécessaires pour gérer les groupes de nœuds.

La `ClusterRole` peut changer au fil du temps, mais elle doit ressembler à l'exemple suivant :

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
```

```
metadata:
  name: eks:node-manager
rules:
- apiGroups:
  - ''
  resources:
  - pods
  verbs:
  - get
  - list
  - watch
  - delete
- apiGroups:
  - ''
  resources:
  - nodes
  verbs:
  - get
  - list
  - watch
  - patch
- apiGroups:
  - ''
  resources:
  - pods/eviction
  verbs:
  - create
```

La `ClusterRoleBinding` peut changer au fil du temps, mais elle doit ressembler à l'exemple suivant :

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: eks:node-manager
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: eks:node-manager
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: User
  name: eks:node-manager
```

Vérifiez que la `eks:node-manager` `ClusterRole` existe.

```
kubectl describe clusterrole eks:node-manager
```

Si elle est présente, comparez la sortie à l'exemple de la `ClusterRole`précédente.

Vérifiez que la `eks:node-manager` `ClusterRoleBinding` existe.

```
kubectl describe clusterrolebinding eks:node-manager
```

Si elle est présente, comparez la sortie à l'exemple de la `ClusterRoleBinding`précédente.

Si vous avez identifié un `ClusterRole` ou un `ClusterRoleBinding` manquant ou défectueux comme cause d'une erreur `AcessDenied` lors de la demande d'opérations sur le groupe de nœuds gérés, vous pouvez les restaurer. Enregistrez le contenu suivant dans un fichier nommé *`eks-node-manager-role.yaml`*.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: eks:node-manager
rules:
- apiGroups:
  - ''
  resources:
  - pods
  verbs:
  - get
  - list
  - watch
  - delete
- apiGroups:
  - ''
  resources:
  - nodes
  verbs:
  - get
  - list
  - watch
  - patch
- apiGroups:
  - ''
```

```
  resources:
  - pods/eviction
  verbs:
  - create
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: eks:node-manager
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: eks:node-manager
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: User
  name: eks:node-manager
```

Appliquez le fichier.

```
kubectl apply -f eks-node-manager-role.yaml
```

Réessayez l'opération de groupe de nœuds pour voir si cela a résolu votre problème.

`Not authorized for images`

Une cause potentielle du message d'erreur `Not authorized for images` est l'utilisation d'une AMI Windows Amazon EKS privée pour lancer des groupes de nœuds gérés Windows. Après avoir publié de nouvelles AMI Windows, AWS rend privées les AMI datant de plus de 4 mois, ce qui les rend inaccessibles. Si votre groupe de nœuds gérés utilise une AMI Windows privée, envisagez de mettre à jour votre groupe de nœuds gérés Windows. Bien que nous ne puissions garantir l'accès aux AMI qui ont été rendues privées, vous pouvez en faire la demande en soumettant un ticket au service d'assistance AWS. Pour plus d'informations, consultez Correctifs dans le Guide de l'utilisateur Amazon EC2.

Le nœud est dans l'état **`NotReady`**

Si votre nœud passe à l'état `NotReady`, cela indique probablement que le nœud est en mauvais état et indisponible pour planifier de nouveaux pods. Cela peut se produire pour diverses raisons,

par exemple si le nœud ne dispose pas de ressources suffisantes pour le processeur, la mémoire ou l'espace disque disponible.

Pour les AMI Windows optimisées pour Amazon EKS, aucune réservation de ressources de calcul n'est spécifiée par défaut dans la configuration `kubelet`. Afin d'éviter les problèmes de ressources, vous pouvez réserver des ressources de calcul pour les processus système en fournissant au `kubelet` les valeurs de configuration pour kube-reserved et/ou system-reserved. Pour ce faire, utilisez le paramètre de ligne de commande `-KubeletExtraArgs` dans le script d'initialisation. Pour plus d'informations, consultez Réserver des ressources de calcul pour les démons système dans la documentation Kubernetes et the section called "Paramètres de configuration du script d'amorçage" dans ce guide d'utilisation.

Collecteur de journaux EKS

Pour résoudre les problèmes liés aux nœuds Amazon EKS, un script pré-créé est disponible sur les nœuds à l'emplacement `/etc/eks/log-collector-script/eks-log-collector.sh`. Vous pouvez utiliser le script pour collecter des journaux de diagnostic pour les cas de support et le dépannage général.

Utilisez la commande suivante pour exécuter le script sur votre nœud :

```
sudo bash /etc/eks/log-collector-script/eks-log-collector.sh
```

> **Note**
>
> Si le script n'est pas présent à cet emplacement. Vous pouvez manuellement télécharger et exécuter le script à l'aide de la commande suivante :
>
> ```
> curl -O https://amazon-eks.s3.amazonaws.com/support/log-collector-script/linux/
> eks-log-collector.sh
> sudo bash eks-log-collector.sh
> ```

Le script collecte les informations de diagnostic suivantes.

```
$ sudo bash /etc/eks/log-collector-script/eks-log-collector.sh
```

```
      This is version 0.7.8. New versions can be found at https://github.com/awslabs/
amazon-eks-ami/blob/main/log-collector-script/

Trying to collect common operating system logs...
Trying to collect kernel logs...
Trying to collect mount points and volume information...
...
...

 Done... your bundled logs are located in /var/log/eks_i-EXAMPLE_2025-03-25_0000-
UTC_0.7.8.tar.gz
```

Les informations de diagnostic sont collectées et stockés dans :

```
/var/log/eks_i-EXAMPLE_2025-03-25_0000-UTC_0.7.8.tar.gz
```

Pour récupérer le lot de journaux pour les nœuds Bottlerocket, veuillez vous reporter à la section Journal Bottlerocket pour plus de détails.

Le réseau d'exécution du conteneur n'est pas prêt

Vous pouvez recevoir une erreur `Container runtime network not ready` et des erreurs d'autorisation similaires aux suivantes :

```
4191 kubelet.go:2130] Container runtime network not ready: NetworkReady=false
 reason:NetworkPluginNotReady message:docker: network plugin is not ready: cni config
 uninitialized
4191 reflector.go:205] k8s.io/kubernetes/pkg/kubelet/kubelet.go:452: Failed to list
 *v1.Service: Unauthorized
4191 kubelet_node_status.go:106] Unable to register node
 "ip-10-40-175-122.ec2.internal" with API server: Unauthorized
4191 reflector.go:205] k8s.io/kubernetes/pkg/kubelet/kubelet.go:452: Failed to list
 *v1.Service: Unauthorized
```

Les raisons possibles sont les suivantes :

1. Soit vous ne disposez pas d'un `aws-auth` `ConfigMap` sur votre cluster, soit celui-ci ne contient pas les entrées pour le rôle IAM avec lequel vous avez configuré vos nœuds.

   Pour résoudre le problème, affichez les entrées existantes de votre `ConfigMap` en remplaçant *`my-cluster`* dans la commande suivante par le nom de votre cluster et en exécutant la

commande modifiée : `eksctl get iamidentitymapping --cluster` *`my-cluster`* . Si vous recevez un message d'erreur de la commande, cela peut être dû au fait que votre cluster ne dispose pas d'un `aws-auth ConfigMap`. La commande suivante ajoute une entrée à la `ConfigMap`. Si le `ConfigMap` n'existe pas, la commande le crée également. Remplacez *`111122223333`* par l'ID de compte AWS du rôle IAM et *`myAmazonEKSNodeRole`* par le nom du rôle de votre nœud.

```
eksctl create iamidentitymapping --cluster my-cluster \
    --arn arn:aws:iam::111122223333:role/myAmazonEKSNodeRole --group
 system:bootstrappers,system:nodes \
    --username system:node:{{EC2PrivateDNSName}}
```

L'ARN du rôle que vous spécifiez ne peut pas inclure de chemin autre que `/`. Par exemple, si le nom de votre rôle est `development/apps/my-role`, vous devez le remplacer par `my-role` lorsque vous spécifiez l'ARN du rôle. Assurez-vous que vous spécifiez l'ARN pour le rôle IAM du nœud (et non l'ARN du profil d'instance).

2. Vos nœuds autogérés se trouvent dans un cluster dont la version de plateforme correspond à la version minimale indiquée dans les conditions préalables de la rubrique Accorder aux utilisateurs IAM l'accès à Kubernetes avec des entrées d'accès EKS, mais aucune entrée n'est répertoriée dans le `aws-auth ConfigMap` (voir élément précédent) pour le rôle IAM du nœud ou aucune entrée d'accès n'existe pour le rôle. Pour résoudre le problème, affichez vos entrées existantes en remplaçant *`my-cluster`* dans la commande suivante par le nom de votre cluster et en exécutant la commande modifiée : `aws eks list-access-entries --cluster-name` *`my-cluster`* . La commande suivante ajoute une entrée d'accès pour le rôle IAM du nœud. Remplacez *`111122223333`* par l'ID de compte AWS du rôle IAM et *`myAmazonEKSNodeRole`* par le nom du rôle de votre nœud. Si vous disposez d'un nœud Windows, remplacez *`EC2_LINUX`* par `EC2_Windows`. Assurez-vous que vous spécifiez l'ARN pour le rôle IAM du nœud (et non l'ARN du profil d'instance).

```
aws eks create-access-entry --cluster-name my-cluster --principal-arn
 arn:aws:iam::111122223333:role/myAmazonEKSNodeRole --type EC2_LINUX
```

Délai d'expiration de la liaison TLS

Lorsqu'un nœud est incapable d'établir une connexion avec le point de terminaison du serveur d'API public, vous pouvez voir une erreur similaire à l'erreur suivante.

```
server.go:233] failed to run Kubelet: could not init cloud provider "aws": error
 finding instance i-1111f2222f333e44c: "error listing AWS instances: \"RequestError:
 send request failed\\ncaused by: Post  net/http: TLS handshake timeout\""
```

Le processus `kubelet` se reproduira continuellement et testera le point de terminaison du serveur d'API. L'erreur peut également se produire temporairement au cours de toute procédure qui effectue une mise à jour continue du cluster dans le plan de contrôle, telle qu'une modification de configuration ou une mise à jour de version.

Pour résoudre le problème, vérifiez la table de routage et les groupes de sécurité pour vous assurer que le trafic provenant des nœuds peut atteindre le point de terminaison public.

InvalidClientTokenId

Si vous utilisez des rôles IAM pour les comptes de service d'un pod ou d'un DaemonSet déployé sur un cluster dans une région AWS en Chine et que vous n'avez pas défini la variable d'environnement `AWS_DEFAULT_REGION` dans la spécification, le pod ou le DaemonSet peut recevoir l'erreur suivante :

```
An error occurred (InvalidClientTokenId) when calling the GetCallerIdentity operation:
 The security token included in the request is invalid
```

Pour résoudre le problème, vous devez ajouter la variable d'environnement `AWS_DEFAULT_REGION` à la spécification de votre pod ou DaemonSet, comme indiqué dans l'exemple de spécification de pod suivant.

```
apiVersion: v1
kind: Pod
metadata:
  name: envar-demo
  labels:
    purpose: demonstrate-envars
spec:
  containers:
  - name: envar-demo-container
    image: gcr.io/google-samples/node-hello:1.0
    env:
    - name: AWS_DEFAULT_REGION
      value: "region-code"
```

Les groupes de nœuds doivent correspondre à la version Kubernetes avant de mettre à niveau le plan de contrôle

Avant de mettre à niveau un plan de contrôle vers une nouvelle version Kubernetes, la version mineure des nœuds gérés et Fargate de votre cluster doit être identique à la version actuelle de votre plan de contrôle. L'API `update-cluster-version` Amazon EKS rejette les demandes tant que vous n'avez pas mis à niveau tous les nœuds gérés par Amazon EKS vers la version actuelle du cluster. Amazon EKS fournit des API pour mettre à niveau les nœuds gérés. Pour plus d'informations sur la mise à niveau de la version Kubernetes d'un groupe de nœuds gérés, consultez the section called "Mettre à jour". Pour mettre à niveau la version d'un nœud Fargate, supprimez le pod représenté par le nœud et redéployez-le après avoir mis à niveau votre plan de contrôle. Pour de plus amples informations, consultez the section called "Mises à jour de la version Kubernetes".

Lors du lancement de nombreux nœuds, des erreurs **`Too Many Requests`** se produisent

Si vous lancez plusieurs nœuds simultanément, vous pouvez voir un message d'erreur dans les journaux d'exécution des données utilisateur Amazon EC2 indiquant `Too Many Requests`. Cela peut se produire parce que le plan de contrôle est surchargé d'appels `describeCluster`. Cette surcharge se traduit par une limitation, des nœuds qui ne parviennent pas à exécuter le script d'amorçage et des nœuds qui ne parviennent pas à rejoindre le cluster.

Assurez-vous que les arguments `--apiserver-endpoint`, `--b64-cluster-ca` et `--dns-cluster-ip` sont transmis au script de démarrage du nœud. Lorsque vous incluez ces arguments, le script d'initialisation n'a pas besoin d'effectuer d'appel `describeCluster`, ce qui permet d'éviter la surcharge du plan de contrôle. Pour de plus amples informations, consultez the section called "Fournissez des données utilisateur pour transmettre des arguments au `bootstrap.sh` fichier inclus dans une Linux/Bottlerocket AMI optimisée pour Amazon EKS".

Réponse d'erreur non autorisée HTTP 401 sur les requêtes du serveur API Kubernetes

Ces erreurs s'affichent si le jeton de compte de service d'un pod a expiré sur un cluster.

Le serveur API Kubernetes de votre cluster Amazon EKS rejette les demandes dont les jetons datent de plus de 90 jours. Dans les versions précédentes de Kubernetes, les jetons n'avaient pas de

période d'expiration. Cela signifie que les clients qui s'appuient sur ces jetons doivent les actualiser dans l'heure. Pour éviter que le serveur API Kubernetes ne rejette votre demande en raison d'un jeton non valide, la version du kit SDK client Kubernetes utilisée par votre charge de travail doit être identique ou postérieure aux versions suivantes :

- Go version `0.15.7` et ultérieure
- Python version `12.0.0` et ultérieure
- Java version `9.0.0` et ultérieure
- JavaScript version `0.10.3` et ultérieure
- Branche `master` Ruby
- Haskell version `0.3.0.0`
- Version C# `7.0.5` et ultérieure

Vous pouvez identifier tous les pod existants dans votre cluster qui utilisent des jetons périmés. Pour de plus amples informations, consultez the section called "Jetons de compte de service".

La version de la plateforme Amazon EKS est inférieure de plus de deux versions à la version actuelle de la plateforme

Cela peut se produire lorsque Amazon EKS n'est pas en mesure de mettre à jour automatiquement la platform-version de votre cluster. Bien qu'il existe de nombreuses causes à cela, certaines des causes les plus courantes suivent. Si l'un de ces problèmes s'applique à votre cluster, celui-ci peut continuer à fonctionner, mais sa version de plateforme ne sera pas mise à jour par Amazon EKS.

Problème

Le Rôle IAM du cluster a été supprimé : ce rôle a été spécifié lors de la création du cluster. Vous pouvez voir quel rôle a été spécifié à l'aide de la commande suivante. Remplacez *my-cluster* par le nom de votre cluster.

```
aws eks describe-cluster --name my-cluster --query cluster.roleArn --output text | cut
 -d / -f 2
```

L'exemple qui suit illustre un résultat.

```
eksClusterRole
```

Solution

Création d'un nouveau rôle IAM du cluster portant le même nom.

Problème

Un sous-réseau spécifié lors de la création du cluster a été supprimé : les sous-réseaux à utiliser avec le cluster ont été spécifiés lors de la création du cluster. Vous pouvez voir quels sous-réseaux ont été spécifiés à l'aide de la commande suivante. Remplacez *my-cluster* par le nom de votre cluster.

```
aws eks describe-cluster --name my-cluster --query cluster.resourcesVpcConfig.subnetIds
```

L'exemple qui suit illustre un résultat.

```
[
"subnet-EXAMPLE1",
"subnet-EXAMPLE2"
]
```

Solution

Vérifiez si les ID de sous-réseau existent dans votre compte.

```
vpc_id=$(aws eks describe-cluster --name my-cluster --query
 cluster.resourcesVpcConfig.vpcId --output text)
aws ec2 describe-subnets --filters "Name=vpc-id,Values=$vpc_id" --query
 "Subnets[*].SubnetId"
```

L'exemple qui suit illustre un résultat.

```
[
"subnet-EXAMPLE3",
"subnet-EXAMPLE4"
]
```

Si les ID de sous-réseau renvoyés dans la sortie ne correspondent pas aux ID de sous-réseau spécifiés lors de la création du cluster, vous devez modifier les sous-réseaux utilisés par le cluster si vous voulez qu'Amazon EKS mette à jour le cluster. En effet, si vous avez spécifié plus de deux sous-réseaux lors de la création de votre cluster, Amazon EKS sélectionne de manière aléatoire

les sous-réseaux que vous avez spécifiés pour y créer de nouvelles interfaces réseau Elastic. Ces interfaces réseau permettent au plan de contrôle de communiquer avec vos nœuds. Amazon EKS ne mettra pas à jour le cluster si le sous-réseau qu'il sélectionne n'existe pas. Vous n'avez aucun contrôle sur les sous-réseaux que vous avez spécifiés lors de la création du cluster dans lesquels Amazon EKS choisit de créer une nouvelle interface réseau.

Lorsque vous lancez une mise à jour de la version Kubernetes pour votre cluster, la mise à jour peut échouer pour la même raison.

Problème

Un groupe de sécurité spécifié lors de la création du cluster a été supprimé : si vous avez spécifié des groupes de sécurité lors de la création du cluster, vous pouvez voir leurs ID à l'aide de la commande suivante. Remplacez *my-cluster* par le nom de votre cluster.

```
aws eks describe-cluster --name my-cluster --query
 cluster.resourcesVpcConfig.securityGroupIds
```

L'exemple qui suit illustre un résultat.

```
[
    "sg-EXAMPLE1"
]
```

Si `[]` est renvoyé, cela signifie qu'aucun groupe de sécurité n'a été spécifié lors de la création du cluster et que l'absence de groupe de sécurité n'est pas le problème. Si des groupes de sécurité sont renvoyés, vérifiez qu'ils existent dans votre compte.

Solution

Vérifiez si ces groupes de sécurité existent dans votre compte.

```
vpc_id=$(aws eks describe-cluster --name my-cluster --query
 cluster.resourcesVpcConfig.vpcId --output text)
aws ec2 describe-security-groups --filters "Name=vpc-id,Values=$vpc_id" --query
 "SecurityGroups[*].GroupId"
```

L'exemple qui suit illustre un résultat.

```
[
"sg-EXAMPLE2"
```

```
]
```

Si les ID de groupe de sécurité renvoyés dans la sortie ne correspondent pas aux ID de groupe de sécurité qui ont été spécifiés lors de la création du cluster, et si vous voulez qu'Amazon EKS mette à jour le cluster, vous devez modifier les groupes de sécurité utilisés par le cluster. Amazon EKS ne mettra pas à jour un cluster si les ID de groupe de sécurité spécifiés lors de la création du cluster n'existent pas.

Lorsque vous lancez une mise à jour de la version Kubernetes pour votre cluster, la mise à jour peut échouer pour la même raison.

- Vous ne disposez pas d'au moins six (bien que nous recommandions 16) adresses IP disponibles dans chacun des sous-réseaux que vous avez spécifiés lors de la création de votre cluster. Si vous ne disposez pas d'un nombre suffisant d'adresses IP disponibles dans le sous-réseau, vous devez soit libérer des adresses IP dans le sous-réseau, soit modifier les sous-réseaux utilisés par le cluster afin d'utiliser des sous-réseaux disposant d'un nombre suffisant d'adresses IP disponibles.
- Vous avez activé le chiffrement des secrets lors de la création de votre cluster et la clé AWS KMS que vous avez spécifiée a été supprimée. Si vous souhaitez qu'Amazon EKS mette à jour le cluster, vous devez créer un nouveau cluster

Foire aux questions sur l'intégrité du cluster et codes d'erreur avec chemins de résolution

Amazon EKS détecte les problèmes liés à vos clusters EKS et à l'infrastructure du cluster, puis les stocke dans l'objet health de votre ressource de cluster EKS. Vous pouvez détecter et résoudre les problèmes de cluster plus rapidement à l'aide des informations relatives à l'état du cluster. Cela vous permet de créer des environnements d'applications plus sécurisés et à jour. De plus, il peut vous être impossible de mettre à niveau vers des versions plus récentes de Kubernetes ou d'Amazon EKS d'installer des mises à jour de sécurité sur un cluster dégradé en raison de problèmes liés à l'infrastructure nécessaire ou à la configuration du cluster. Amazon EKS peut mettre 3 heures pour détecter les problèmes ou détecter qu'un problème est résolu.

La santé d'un cluster Amazon EKS est une responsabilité partagée entre Amazon EKS et ses utilisateurs. Vous êtes responsable de l'infrastructure préalable des rôles IAM et des sous-réseaux Amazon VPC, ainsi que des autres infrastructures nécessaires, qui doivent être fournies à l'avance. Amazon EKS détecte les modifications apportées à la configuration de cette infrastructure et du cluster.

Pour accéder à l'état de santé de votre cluster dans la console Amazon EKS, recherchez un tableau intitulé Problèmes de santé dans l'onglet Problèmes de santé du cluster du tableau de bord d'observabilité accessible depuis la page de détails du cluster Amazon EKS. Ces données seront également disponibles en appelant l'action `DescribeCluster` dans l'API EKS, par exemple à partir de l'interface de ligne de commande AWS.

Pourquoi utiliser cette fonctionnalité ?

Vous pourrez bénéficier d'une visibilité accrue sur l'état de votre cluster Amazon EKS, diagnostiquer et résoudre rapidement les problèmes, sans avoir à passer du temps à déboguer ou à ouvrir des cas de support AWS. Par exemple : si vous avez accidentellement supprimé un sous-réseau pour le cluster Amazon EKS, Amazon EKS ne pourra pas créer d'interfaces réseau intercompte et les commandes AWS CLI Kubernetes telles que `kubectl` exec ou `kubectl` logs. L'erreur suivante s'affiche : `Error from server: error dialing backend: remote error: tls: internal error.` Le problème d'état Amazon EKS indique : `subnet-da60e280 was deleted: could not create network interface`.

Comment cette fonctionnalité est-elle liée ou fonctionne-t-elle avec d'autres services AWS ?

Les rôles IAM et les sous-réseaux Amazon VPC sont deux exemples d'infrastructure préalable avec laquelle l'état du cluster détecte les problèmes. Cette fonctionnalité renvoie des informations détaillées si ces ressources ne sont pas correctement configurées.

Un cluster présentant des problèmes de santé entraîne-t-il des frais ?

Oui, chaque cluster Amazon EKS est facturé au tarif standard d'Amazon EKS. La fonctionnalité liée à l'état du cluster est disponible sans frais supplémentaires.

Cette fonctionnalité fonctionne-t-elle avec les clusters Amazon EKS sur AWS Outposts ?

Oui, les problèmes de cluster sont détectés pour les clusters EKS dans le cloud AWS, y compris les clusters étendus sur AWS Outposts et les clusters locaux sur AWS Outposts. La santé du cluster ne détecte pas les problèmes avec Amazon EKS Anywhere ou Amazon EKS Distro (EKS-D).

Puis-je être averti lorsque de nouveaux problèmes sont détectés ?

Oui. AWS envoie un courriel et une notification sur le tableau de bord d'intégrité personnel lorsque de nouveaux problèmes d'intégrité du cluster sont détectés.

La console m'avertit-elle des problèmes d'intégrité ?

Oui, tout cluster présentant des problèmes d'état présentera une bannière en haut de la console.

Les deux premières colonnes sont celles qui sont nécessaires pour les valeurs de réponse de l'API. Le troisième champ de l'objet Health ClusterIssue est resourceIds, dont le retour dépend du type de problème.

Code	Message	ResourceIds	Le cluster est-il récupérable ?
SUBNET_NOT_FOUND	Nous n'avons pas pu trouver un ou plusieurs sous-réseaux actuellement associés à votre cluster. Veuillez appeler l'API Amazon EKS update-cluster-config pour mettre à jour les sous-réseaux.	ID de sous-réseaux	Oui
SECURITY_GROUP_NOT_FOUND	Nous n'avons pas pu trouver un ou plusieurs groupes de sécurité actuellement associés à votre cluster. Appelez l'API update-cluster-config d'Amazon EKS pour mettre à jour les groupes de sécurité	ID de groupe de sécurité	Oui
IP_NOT_AVAILABLE	Un ou plusieurs sous-réseaux associés à votre cluster ne disposent pas d'un nombre suffisant d'adresses IP pour qu'Amazon EKS puisse effectuer des	ID de sous-réseaux	Oui

Code	Message	ResourceIds	Le cluster est-il récupérable ?
	opérations de gestion de cluster. Libérez des adresses dans le ou les sous-réseaux ou associez différents sous-réseaux à votre cluster à l'aide de l'API update-cluster-config d'Amazon EKS.		
VPC_NOT_FOUND	Nous n’avons pas pu trouver le VPC associé à votre cluster. Vous devez supprimer et recréer votre cluster.	ID du VPC	Non
ASSUME_ROLE_ACCESS_DENIED	Votre cluster n'utilise pas le rôle lié au service Amazon EKS. Nous n’avons pas pu assumer le rôle associé à votre cluster pour effectuer les opérations de gestion Amazon EKS requises. Vérifiez que le rôle existe et qu'il dispose de la politique de confiance requise.	Rôle IAM du cluster	Oui

Code	Message	ResourceIds	Le cluster est-il récupérable ?
PERMISSIO N_ACCESS_DENIED	Votre cluster n'utilise pas le rôle lié au service Amazon EKS. Le rôle associé à votre cluster n'accorde pas les autorisations suffisant es à Amazon EKS pour effectuer les opérations de gestion requises. Vérifiez les politiques associées au rôle de cluster et si des politiques de refus distinctes sont appliquées.	Rôle IAM du cluster	Oui
ASSUME_RO LE_ACCESS _DENIED_U SING_SLR	Nous n’avons pas pu assumer le rôle lié à un service de gestion du cluster Amazon EKS. Vérifiez que le rôle existe et qu'il dispose de la politique de confiance requise.	Rôle lié au service Amazon EKS	Oui

Code	Message	ResourceIds	Le cluster est-il récupérable ?
PERMISSIO N_ACCESS_ DENIED_USING_SLR	Le rôle lié au service de gestion de cluster Amazon EKS n'accorde pas les autorisations suffisant es à Amazon EKS pour effectuer les opérations de gestion requises. Vérifiez les politiques associées au rôle de cluster et si des politiques de refus distinctes sont appliquées.	Rôle lié au service Amazon EKS	Oui
OPT_IN_REQUIRED	Votre compte ne dispose pas d'un abonnement au service Amazon EC2. Mettez à jour les abonnements de votre compte sur la page des paramètres de votre compte.	N/A	Oui
STS_REGIO NAL_ENDPO INT_DISABLED	Le point de terminais on régional STS est désactivé. Activez le point de terminaison pour qu'Amazon EKS effectue les opération s de gestion de cluster requises.	N/A	Oui

Code	Message	ResourceIds	Le cluster est-il récupérable ?
KMS_KEY_DISABLED	La clé AWS KMS associée à votre cluster est désactivée. Réactivez la clé pour récupérer votre cluster.	L'ARN de la clé KMS	Oui
KMS_KEY_NOT_FOUND	Nous n'avons pas pu trouver la clé AWS KMS associée à votre cluster. Vous devez supprimer et recréer le cluster.	L'ARN de la clé KMS	Non
KMS_GRANT_REVOKED	Les autorisations pour la clé AWS KMS associée à votre cluster sont révoquées. Vous devez supprimer et recréer le cluster.	L'ARN de la clé KMS	Non

Étendre les capacités d'Amazon EKS avec des projets open source

Ces projets open source étendent les fonctionnalités des clusters Kubernetes exécutés sur ou en dehors de ceux-ci AWS, y compris les clusters gérés par Amazon EKS.

Prise en charge des logiciels déployés sur EKS

Lorsque vous consultez la documentation Amazon EKS, vous trouverez des références à divers outils et logiciels open source dans nos procédures et exemples. Ces outils comprennent le Kubernetes Metrics Server et le Cert Manager.

Veuillez noter que tout logiciel tiers ou open source que vous choisissez de déployer ne relève pas du champ d'application de vos Contrats de AWS Support. L'un des avantages de Kubernetes est sa communauté open source active. Nous vous recommandons de travailler directement avec les communautés open source et les responsables de projets concernés afin de mettre en place des canaux de support appropriés pour ces composants. Pour plus d'informations, consultez les projets gradués et en incubation associés à la Cloud Native Computing Foundation (CNCF).

L'écosystème Kubernetes comprend de nombreux projets et composants qui bénéficient de différents niveaux de support communautaire, de temps de réponse et de cas d'utilisation prévus. Lorsque vous implémentez ces technologies avec EKS, assurez-vous de bien comprendre la matrice de support de chaque composant.

AWS gère les composants open source que nous intégrons dans le plan de contrôle EKS. Cela inclut notre pipeline de sécurité complet couvrant la vérification des builds, l'analyse des vulnérabilités, les tests de validation et la gestion des correctifs pour toutes les images de conteneurs et tous les binaires que nous distribuons. Par exemple, AWS est responsable du serveur d'API Kubernetes. Le serveur API Kubernetes est couvert par le contrat de niveau de service Amazon EKS. Vous pouvez utiliser votre plan de support Amazon Web Services pour résoudre les problèmes liés au serveur API Kubernetes ou obtenir des conseils généraux.

Vous devez examiner attentivement le support proposé pour les différents modules complémentaires Amazon EKS. AWS les modules complémentaires sont les seuls types de modules complémentaires Amazon EKS entièrement pris en charge par AWS. AWS Les modules complémentaires Marketplace sont principalement pris en charge par AWS les partenaires. Les modules complémentaires

communautaires bénéficient d'une assistance de base sur le cycle de vie de AWS. Pour plus d'informations, consultez Assistance pour les modules complémentaires.

Chaque module complémentaire EKS, quel que soit son type, bénéficie d'un support de base sur le cycle de vie de la part d'EKS, y compris les modules complémentaires Marketplace. L'assistance de base tout au long du cycle de vie comprend l'installation et la désinstallation du module complémentaire. Pour plus d'informations sur les types de modules complémentaires Amazon EKS disponibles et les niveaux d'assistance associés, consultez Portée de l'assistance pour les modules complémentaires Amazon EKS. Pour voir les modules complémentaires entièrement pris en charge par AWS, consultez les modules complémentaires Amazon Web Services.

- Pour plus d'informations sur nos pratiques de sécurité et les limites de l'assistance, consultez Sécurité dans Amazon EKS.
- Pour plus d'informations sur les modules complémentaires destinés à la communauté et au AWS marché disponibles via les modules complémentaires Amazon EKS, consultez la section Support des modules complémentaires EKS.

Outils de gestion

Outils de gestion connexes pour les clusters Amazon EKS et Kubernetes.

eksctl

`eksctl` est un outil de CLI pour créer des clusters sur Amazon EKS.

- URL du projet
- Documentation du projet
- AWS blog open source : eksctl : cluster Amazon EKS avec une seule commande

AWS Contrôleurs pour Kubernetes

Avec AWS Controllers for Kubernetes, vous pouvez créer et gérer des AWS ressources directement à partir de votre cluster Kubernetes.

Disponible dans la version EKS Capabilities.

- URL du projet
- AWS blog open source : l'opérateur AWS de service pour Kubernetes est désormais disponible

kro (Kube Resource Orchestrator)

kro vous permet de créer des Kubernetes personnalisés APIs qui composent plusieurs ressources en abstractions de niveau supérieur. Les équipes chargées de la plateforme peuvent définir des modèles réutilisables à l'aide de barrières de sécurité, tandis que les équipes chargées des applications utilisent des méthodes simples et de haut niveau APIs pour approvisionner et gérer les ressources.

Disponible dans la version EKS Capabilities.

- URL du projet
- Documentation du projet

Argo CD

Argo CD est un outil de diffusion GitOps continue déclaratif pour Kubernetes. Il surveille en permanence vos référentiels Git et synchronise automatiquement les modifications apportées à vos clusters.

Disponible dans la version EKS Capabilities.

- URL du projet
- Documentation du projet

Flux CD

Flux est un outil que vous pouvez utiliser pour gérer la configuration de votre cluster à l'aide de Git. Il utilise un opérateur dans le cluster pour déclencher des déploiements à l'intérieur de Kubernetes. Pour plus d'informations sur les opérateurs, consultez OperatorHub.io on GitHub.

- URL du projet
- Documentation du projet

CDK pour Kubernetes

Avec le CDK pour Kubernetes (cdk8s), vous pouvez définir des applications et des composants Kubernetes en utilisant des langages de programmation familiers. Les applications cdk8s se

synthétisent en manifestes Kubernetes standard, qui peuvent être appliqués à n'importe quel cluster Kubernetes.

- URL du projet
- Documentation du projet
- AWS blog sur les conteneurs : Présentation de cdk8s+ : piloté par l'intention pour les objets Kubernetes APIs

Réseaux

Projets de mise en réseau connexes pour les clusters Amazon EKS et Kubernetes.

Plugin CNI Amazon VPC pour Kubernetes

Amazon EKS prend en charge la mise en réseau VPC via le plugin CNI Amazon VPC pour Kubernetes. Le plug-in attribue une adresse IP de votre VPC à chaque pod.

- URL du projet
- Documentation du projet

AWS Contrôleur Load Balancer pour Kubernetes

Le AWS Load Balancer Controller permet de gérer les AWS Elastic Load Balancers pour un cluster Kubernetes. Il satisfait les ressources Kubernetes Ingress en provisionnant des équilibreurs de charge d'application. AWS Il satisfait les ressources du service Kubernetes en fournissant des équilibreurs de charge réseau. AWS

- URL du projet
- Documentation du projet

ExternalDNS

ExternalDNS synchronise les services Kubernetes exposés et les entrées avec les fournisseurs DNS tels qu'Amazon Route 53 et Service Discovery. AWS

- URL du projet

- Documentation du projet

Machine learning

Projets de machine learning connexes pour les clusters Amazon EKS et Kubernetes.

Kubeflow

Une boîte à outils de machine learning pour Kubernetes.

- URL du projet
- Documentation du projet
- AWS blog open source : Kubeflow sur Amazon EKS

Auto Scaling

Projets de scalabilité automatique connexes pour les clusters Amazon EKS et Kubernetes.

Cluster Autoscaler

Cluster Autoscaler est un outil qui ajuste automatiquement la taille du cluster Kubernetes en fonction de la sollicitation de l'UC et de la mémoire.

- URL du projet
- Documentation du projet
- Atelier Amazon EKS : Cluster Autoscaler

Karpenter

Karpenter est un auto-scaler de nœuds Kubernetes créé pour offrir flexibilité, performances et simplicité.

- URL du projet
- Documentation du projet
- Atelier Amazon EKS : Karpenter

Escalator

Escalator est un outil de scalabilité automatique horizontal optimisé de lot ou de tâche pour Kubernetes.

- URL du projet
- Documentation du projet

Contrôle

Projets de surveillance connexes pour les clusters Amazon EKS et Kubernetes.

Prometheus

Prometheus est une boîte à outils de surveillance de systèmes et d'alerte open source.

- URL du projet
- Documentation du projet
- Atelier Amazon EKS : https://eksworkshop.com/intermediate/240_monitoring/

Intégration continue/déploiement continu

CI/CD Projets impératifs connexes pour les clusters Amazon EKS et Kubernetes.

Jenkins X

Solution CI/CD pour les applications cloud modernes sur les clusters Amazon EKS et Kubernetes.

- URL du projet
- Documentation du projet

Découvrez les nouvelles fonctionnalités et la feuille de route d’Amazon EKS

Vous pouvez en savoir plus sur les nouvelles fonctionnalités d’Amazon EKS en faisant défiler le flux Nouveautés sur la page Nouveautés AWS. Vous pouvez également consulter la feuille de route sur GitHub, qui vous permet de connaître les fonctions et les priorités à venir afin que vous puissiez planifier la façon dont vous souhaitez utiliser Amazon EKS à l'avenir. Vous pouvez nous fournir des commentaires directs sur les priorités de la feuille de route.

Historique du document

Le tableau suivant décrit certaines des principales mises à jour et les nouvelles fonctions pour le Guide de l'utilisateur Amazon EKS. Pour recevoir des notifications lorsque ce tableau reçoit une nouvelle entrée, vous pouvez vous abonner à l'URL suivante à l'aide d'un lecteur RSS :

```
https://docs.aws.amazon.com/eks/latest/userguide/doc-history.rss
```

Modification	Description	Date
AWS mises à jour des politiques gérées	Ajouté `autoscaling:PutWarmPool` `autoscaling:DeleteWarmPool` , et `autoscaling:DescribeWarmPool` autorisations relatives à la `AWSServiceRoleForAmazonEKSNodegroup` politique. Cela permet aux groupes de nœuds gérés par Amazon EKS de gérer les ressources du warm pool ASG sous-jacentes tout au long du cycle de vie des groupes de nœuds.	17 février 2026
AWS mises à jour des politiques gérées	Suppression de l'exigence de préfixe « eks » dans le nom du profil d'instance cible pour l'`iam:GetInstanceProfile` autorisation dans`AmazonEKSServiceRolePolicy` . Cela permet au mode automatique d'Amazon EKS de valider et d'utiliser des	2 février 2026

	profils d'instance personnal isés NodeClasses sans avoir besoin du préfixe de dénomination « eks ».	
Version de Kubernetes `1.35`	Ajout de la prise en charge de la version Kubernetes `1.35` pour les nouveaux clusters et les mises à niveau de version.	27 janvier 2026
AWS mises à jour des politiques gérées	Ajout de `ec2:LockSnapshot` l'autorisation permettant `AmazonEBSCSIDriverPolicy` au pilote EBS CSI de verrouiller directement les instantanés EBS.	15 janvier 2026
Fonctionnalités d'Amazon EKS	Amazon EKS fournit désormais des fonctionnalités de cluster entièrement gérées avec les fonctionnalités EKS, avec prise en charge du déploiement continu déclaratif avec prise en charge d'Argo CD, gestion des AWS ressources avec prise en charge des AWS contrôleurs pour Kubernetes, et composition et orchestration des ressources Kubernetes avec prise en charge de Kube Resource Orchestrator.	30 novembre 2025

Nouvelle politique AWS gérée	Amazon EKS a publié une nouvelle politique gérée `AmazonEKSMCPReadOnlyAccess` pour activer les outils en lecture seule sur le serveur Amazon EKS MCP à des fins d'observabilité et de résolution des problèmes . Pour plus d'informations, consultez les mises à jour des politiques AWS gérées par Amazon EKS.	21 novembre 2025
Observabilité du réseau	Amazon EKS fournit désormais une meilleure observabilité du réseau de conteneurs grâce à la surveillance des performances et à la visibilité du trafic de charge de travail.	19 novembre 2025
AWS mises à jour des politiques gérées	Ajout de `ec2:CopyVolumes` l'autorisation `AmazonEBSCSIDriverPolicy` permettant au pilote EBS CSI de copier directement les volumes EBS.	17 novembre 2025

AWS mises à jour des politiques gérées	Ajouté `ec2:DescribeRouteTables` et `ec2:DescribeNetworkAcls` autorisations pour`AmazonEKSServiceRolePolicy` . Cela permet à Amazon EKS de détecter les problèmes de configuration liés aux tables de routage VPC et au réseau ACLs pour les nœuds hybrides dans le cadre de l'analyse du cluster.	22 octobre 2025
Mise à jour des rôles liés au service	`ssmmessages:OpenDataChannel` Autorisation ajoutée pour `AmazonEKSConnectorServiceRolePolicy`	15 octobre 2025
Version de Kubernetes `1.34`	Ajout de la prise en charge de la version Kubernetes `1.34` pour les nouveaux clusters et les mises à niveau de version.	2 octobre 2025
Réparation automatique des nœuds configurables	Amazon EKS offre désormais un contrôle plus précis sur le comportement de réparation automatique des nœuds.	22 septembre 2025
Actualisation des informations sur le cluster	Vous pouvez désormais actualiser manuellement les informations sur les clusters.	27 août 2025

AWS mises à jour des politiques gérées	Autorisation ajoutée à `AmazonEKSServiceRolePolicy` . Ce rôle peut associer une nouvelle politique d'accès `AmazonEKSEventPolicy` . Autorisations restreintes pour `ec2:DeleteLaunchTemplate` et `ec2:TerminateInstances` .	26 août 2025
Prévention du député confus entre services	Ajout d'un sujet avec un exemple de politique de confiance que vous pouvez appliquer pour la prévention des conflits de délégation entre services. Amazon EKS accepte les conditions `aws:SourceArn` et `aws:SourceAccount` de la politique de confiance d'un rôle de cluster EKS.	19 août 2025
Mise à jour de la version de plateforme Amazon EKS	Il s'agit d'une nouvelle version de plateforme qui comporte des correctifs et des améliorations de sécurité. Cela inclut les nouvelles versions de correctif de Kubernetes `1.33.2`, `1.32.6`, `1.31.10` et `1.30.14`.	30 juillet 2025

Fonctionnalité VPC CNI Multi-NIC pour les pods multi-homed	Amazon EKS ajoute des pods multi-homed au VPC CNI. Vous pouvez désormais configurer une charge de travail et les adresses IP attribuées par le CNI VPC à partir de chaque carte réseau de l'instance EC2 vers chaque pod. L'application peut établir des connexions simultanées afin d'utiliser la bande passante de chaque carte réseau. Chaque interface réseau est configurée dans le même sous-réseau et les mêmes groupes de sécurité que le nœud. Auparavant, vous deviez utiliser Multus CNI pour en exécuter plusieurs autres afin de créer des pods multi-hôtes. CNIs La documentation et les étapes à suivre pour ce faire ont été déplacées vershttps://docs.aws.amazon.com/eks/latest/userguide/pod-multus.html.	15 juillet 2025
Mise à jour du contenu relatif au dépannage du CNI VPC	Élargissement de la page de dépannage pour la politique réseau Kubernetes dans le VPC CNI. Ajout des CRDs autorisations RBAC et de 12 problèmes connus.	30 juin 2025

Mise à jour de la version de plateforme Amazon EKS	Il s'agit d'une nouvelle version de plateforme qui comporte des correctifs et des améliorations de sécurité. Il n'y a pas de nouvelles versions de correctif pour Kubernetes dans cette version de plateforme.	26 juin 2025
AWS mises à jour des politiques gérées	Autorisation `ssmmessages:OpenDataChannel` ajoutée à `AmazonEKSLocalOutpostServiceRolePolicy` .	26 juin 2025
AWS mises à jour des politiques gérées	Ajout d'autorisations à `AmazonEKSServiceRolePolicy` et `AmazonEKSComputePolicy` pour permettre au mode automatique Amazon EKS de lancer des instances à l'aide des réserves de capacité à la demande EC2 de votre compte. De plus, nous avons ajouté les autorisations à `AmazonEKSComputePolicy` permettant à Amazon EKS de créer le rôle lié au service EC2 Spot en votre nom.	20 juin 2025
Mise à jour de politiques gérées	Politique `AmazonEKSDashboardConsoleReadOnly` ajoutée.	19 juin 2025

Mise à jour du mode automatique Amazon EKS vers `NodeClass`	Le modèle `NodeClass` pour les nœuds du mode automatique a ajouté une configuration pour les sous-réseaux de pods séparés. Cela ajoute les clés facultatives `podSubnetSelectorTerms` et `podSecurityGroupSelectorTerms` permet de définir les sous-réseaux et les groupes de sécurité pour les pods.	13 juin 2025
Ciblage des rôles secondaires et inter-comptes avec les identités du pod EKS	Amazon EKS ajoute des rôles IAM cibles aux identités de pod EKS pour un chaînage automatisé des rôles. Vous pouvez utiliser cette fonctionnalité pour assumer automatiquement un rôle dans un autre compte. EKS Pod Identity effectue la rotation des informations d'identification temporaires. Chaque association d'identité de pod doit disposer d'un rôle IAM dans le même compte pour pouvoir assumer le rôle cible.	11 juin 2025
Mise à jour de la version de plateforme Amazon EKS	Il s'agit d'une nouvelle version de plateforme qui comporte des correctifs et des améliorations de sécurité. Il n'y a pas de nouvelles versions de correctif pour Kubernetes dans cette version de plateforme.	11 juin 2025

Expansion de AWS la région Amazon EKS	Amazon EKS est désormais disponible dans la AWS région Asie-Pacifique (Taipeiap-east-2) ().	6 juin 2025
IPv6 Contrôle d'accès IPv6 pour les points d'extrémité publics à double pile pour les nouveaux clusters	Amazon EKS ajoute des blocs CIDR IPv6 pour contrôler l'accès au point de terminais on public du cluster pour les nouveaux clusters IPv6. Auparavant, vous ne pouviez ajouter que des blocs CIDR IPv4 pour autoriser le trafic vers le point de terminaison du cluster public, même pour les points de terminaison de cluster à double pile.	5 juin 2025
Mise à jour de la version de plateforme Amazon EKS	Il s'agit d'une nouvelle version de plateforme qui comporte des correctifs et des améliorations de sécurité. Cela inclut les nouvelles versions correctives de Kubernetes 1.32.5, 1.31.9, et 1.30.13.	30 mai 2025
Nouvelles informations sur les clusters pour les nœuds hybrides EKS	Amazon EKS ajoute de nouvelles informations sur les clusters qui vérifient la configuration de vos nœuds hybrides. Ces vérifications d'informations vous avertiron t des problèmes liés aux nœuds et aux pods sur site et à la configuration réseau à distance du cluster.	29 mai 2025

Version de Kubernetes1.33	Ajout de la prise en charge de la version Kubernetes 1.33 pour les nouveaux clusters et les mises à niveau de version.	29 mai 2025
Support supplémentaire pour le pilote Amazon FSx CSI	Vous pouvez désormais utiliser la console AWS de gestion, la AWS CLI et l'API pour gérer le pilote Amazon FSx CSI.	23 mai 2025
Modifiez les scrapers Prometheus dans la console	Vous pouvez désormais modifier les scrapers du service géré Amazon pour Prometheus dans la console Amazon EKS.	22 mai 2025
Mise à jour de politiques gérées	Politique AmazonEKS DashboardServiceRolePolicy ajoutée.	21 mai 2025
Mise à jour de la version de plateforme Amazon EKS	Il s'agit d'une nouvelle version de plateforme qui comporte des correctifs et des améliorations de sécurité. Il n'y a pas de nouvelles versions de correctif de Kubernetes dans ces versions de plateforme.	16 mai 2025
Nouvelles pages pour Amazon FSx pour les performances de Lustre	Ajout de nouvelles rubriques contenant des informations détaillées sur l'optimisation des performances FSx d'Amazon pour Lustre.	2 mai 2025

Mise à jour du mode automatique Amazon EKS vers `NodeClass`	Le modèle `NodeClass` pour les nœuds du mode automatique a ajouté une configuration pour les proxys réseau en avant. Cela ajoute la clé facultative `advancedNetworking` permettant de configurer votre proxy HTTPS.	30 avril 2025
Bottlerocket pour nœuds hybrides	Bottlerocket est désormais disponible pour les nœuds hybrides EKS.	29 avril 2025
Mise à jour de la version de plateforme Amazon EKS	Il s'agit d'une nouvelle version de plateforme qui comporte des correctifs et des améliorations de sécurité. Il n'y a pas de nouvelles versions de correctif de Kubernetes dans ces versions de plateforme.	29 avril 2025
Nouvelles pages de concepts pour les réseaux hybrides	Des pages ont été ajoutées pour les concepts des nœuds hybrides EKS. Ils couvrent en détail les réseaux sur site et dans le cloud à l'aide de diagrammes.	18 avril 2025

AWS mises à jour des politiques gérées	Ajout d'autorisations à `AmazonEKSClusterPolicy` pour permettre à Amazon EKS d'utiliser les interfaces réseau Elastic créées par le VPC CNI. Cette opération est nécessaire pour permettre à EKS de nettoyer les interfaces réseau Elastic qui restent si le VPC CNI se ferme de manière inattendue.	16 avril 2025
AWS mises à jour des politiques gérées	Des autorisations ont été ajoutées `AmazonEKSServiceRolePolicy` pour permettre aux AI/ML clients d'EKS d'ajouter des règles de sortie au groupe de sécurité par défaut du cluster EKS.	14 avril 2025
État des nœuds pour les nœuds hybrides EKS	Vous pouvez utiliser `eks-node-monitoring-agent` sur les nœuds hybrides, à partir de la version `1.2.0-eksbuild.1` . Exécutez `eks-node-monitoring-agent` en tant que module complémentaire Amazon EKS pour détecter et afficher les problèmes d'état.	31 mars 2025

Nœuds hybrides EKS pour les clusters existants	Vous pouvez désormais ajouter, modifier ou supprimer la configuration des nœuds hybrides des clusters existants . Auparavant, vous ne pouviez ajouter la configuration des nœuds hybrides aux nouveaux clusters que lorsque vous les créiez. Avec les nœuds hybrides Amazon EKS, vous pouvez utiliser votre infrastru cture sur site et périphéri que en tant que nœuds dans des clusters Amazon EKS. AWS gère le plan de contrôle Kubernetes AWS hébergé du cluster Amazon EKS, et vous gérez les nœuds hybrides qui s'exécutent dans vos environnements locaux ou périphériques.	31 mars 2025
Rollback : Empêchez les mises à niveau accidentelles grâce aux informations sur les clusters	Amazon EKS a temporair ement supprimé une fonctionn alité qui vous obligeait à utiliser un indicateur `--force` pour mettre à niveau votre cluster en cas de problèmes liés aux informati ons sur le cluster. Pour plus d'informations, voir Annulatio n temporaire de l'application des informations de mise à niveau lors de la mise à jour de la version du cluster sur GitHub.	28 mars 2025

Bottlerocket FIPS AMIs	Les Bottlerocket FIPS AMIs sont désormais disponibles dans les groupes de nœuds gérés standard.	27 mars 2025
AWS mises à jour des politiques gérées	Ajout d'autorisations à `AmazonEKSServiceRolePolicy` pour permettre à Amazon EKS de mettre fin aux instances EC2 créées par le mode automatique. Ajout d'autorisations à `AmazonEKSServiceRolePolicy` pour permettre à Amazon EKS de mettre fin aux instances EC2 créées par le mode automatique.	28 février 2025
Stratégies de mise à jour pour les groupes de nœuds gérés	Vous pouvez désormais utiliser des stratégies de mise à jour pour configurer le processus de mise à jour de version pour les groupes de nœuds gérés. Cela introduit la stratégie de mise à jour minimale consistant à fermer des nœuds avant d'en créer de nouveaux, ce qui est utile dans les environnements à capacité limitée. La stratégie de mise à jour par défaut conserve le comportement existant.	27 janvier 2025

Version de Kubernetes `1.32`	Ajout de la prise en charge de la version Kubernetes `1.32` pour les nouveaux clusters et les mises à niveau de version.	23 janvier 2025
Expansion de AWS la région Amazon EKS	Amazon EKS est désormais disponible dans les régions AWS Asie-Pacifique (Thaïlande) (`ap-southeast-7`) et Mexique (Centre) (`mx-central-1`). Le mode automatique EKS et les points de terminaison d'un VPC pour l'API EKS ne sont disponibles dans aucune des deux régions.	14 janvier 2025
AWS mises à jour des politiques gérées	Ajout de plusieurs autorisations à `AmazonEBSCSIDriverPolicy` pour permettre au pilote CSI Amazon EBS de restaurer tous les instantanés, d'activer la restauration rapide d'instantanés (FSR) sur les volumes EBS et de modifier les balises sur les volumes.	13 janvier 2025
AWS mises à jour des politiques gérées	Ajout d'autorisations à `AmazonEKSLoadBalancingPolicy` .	26 décembre 2024

Informations actualisées sur le cluster	Les informations sur les mises à niveau Amazon EKS vous avertiront désormais davantage des problèmes liés à l'état du cluster et à la compatibilité des versions. Il peut détecter les problèmes entre différent s composants Kubernete s et Amazon EKS tels que `kubelet`, `kube-proxy` , et les modules complémentaires Amazon EKS.	20 décembre 2024
Agent de surveillance des nœuds et réparation automatiq ue	Vous pouvez utiliser le nouvel `eks-node-monitoring-agent` en tant que module complémentaire Amazon EKS pour détecter et afficher les problèmes d'état. Vous pouvez également activer la réparatio n automatique des nœuds afin de remplacer automatiq uement les nœuds lorsque des problèmes sont détectés.	16 décembre 2024

Nœuds hybrides Amazon EKS	Vous pouvez désormais exécuter un nœud sur site connecté à des clusters Amazon EKS. Avec les nœuds hybrides Amazon EKS, vous pouvez utiliser votre infrastructure sur site et périphérique en tant que nœuds dans des clusters Amazon EKS. AWS gère le plan de contrôle Kubernetes AWS hébergé du cluster Amazon EKS, et vous gérez les nœuds hybrides qui s'exécutent dans vos environnements locaux ou périphériques.	1er décembre 2024
Mode automatique Amazon EKS	Le mode automatique Amazon EKS automatise entièrement la gestion de l’infrastructure des clusters Kubernetes pour le calcul, le stockage et la mise en réseau sur AWS. Il simplifie la gestion de Kubernetes en provisionnant automatiquement l'infrastructure, en sélectionnant les instances de calcul optimales, en dimensionnant les ressources de manière dynamique, en optimisant continuellement les coûts, en appliquant des correctifs aux systèmes d'exploitation et en intégrant les services de sécurité. AWS	1er décembre 2024

Mise à jour de la version de plateforme Amazon EKS	Il s'agit d'une nouvelle version de plateforme qui comporte des correctifs et des améliorations de sécurité. Cela inclut les nouvelles versions de correctif de Kubernetes `1.31.2`, `1.30.6`, `1.29.10` et `1.28.15`.	22 novembre 2024
AWS mises à jour des politiques gérées	`AWSServiceRoleForAmazonEKSNodegroup` mis à jour pour être compatible avec les régions chinoises.	22 novembre 2024
La version 1.30 de Kubernetes est désormais disponible pour les clusters locaux sur Outposts AWS	Vous pouvez désormais créer un cluster local Amazon EKS sur un AWS Outposts à l'aide de Kubernetes version 1.30.	21 novembre 2024
AWS mises à jour des politiques gérées	Politique AWS gérée mise à jour par EKS`AmazonEKSLocalOutpostClusterPolicy` . `ec2:DescribeAvailabilityZones` Autorisation ajoutée afin que le AWS Cloud Controller Manager du plan de contrôle du cluster puisse identifier la zone de disponibilité dans laquelle se trouve chaque nœud.	21 novembre 2024

Bottlerocket utilisant la norme FIPS AMIs 140-3	AMIs Il existe des modèles Bottlerocket préconfigurés pour utiliser des modules cryptographiques validés par la norme FIPS 140-3. Cela inclut le module cryptographique de l'API Kernel Crypto d'Amazon Linux 2023 et le module cryptographique AWS-LC.	20 novembre 2024
AWS mises à jour des politiques gérées	Politique `AWSServiceRoleForAmazonEKSNodegroup` mise à jour pour autoriser `ec2:RebootInstances` sur les instances créées par des groupes de nœuds gérés par Amazon EKS. Restriction des autorisations `ec2:CreateTags` pour les ressources Amazon EC2.	20 novembre 2024
Tableau de bord d'observabilité	Le tableau de bord d'observabilité vous aide à détecter, résoudre et résoudre rapidement les problèmes . De nouvelles CloudWatch métriques vendues sont également disponibles dans l'espace de `AWS/EKS` noms.	18 novembre 2024

AWS mises à jour des politiques gérées	Politique AWS gérée mise à jour par EKS`AmazonEKS ServiceRolePolicy` . Ajout d'autorisations pour les stratégies d'accès EKS, la gestion des équilibreurs de charge et le nettoyage automatisé des ressources du cluster.	16 novembre 2024
Création d'un nouveau rôle dans la console pour les modules complémentaires prenant en charge les identités de pod EKS	De nouvelles étapes ont été ajoutées lors de l'utilisation de la console pour créer ou mettre à jour des modules complémentaires prenant en charge les identités de pod EKS. Elles vous permettent de générer automatiquement des rôles IAM avec le nom, la stratégie de rôle et la stratégie de confiance appropriés pour le module complémentaire.	15 novembre 2024
Groupes de nœuds gérés dans les Zones AWS Locales	Les groupes de nœuds gérés peuvent désormais être créés dans les Zones AWS Locales.	15 novembre 2024
De nouvelles métriques sont disponibles	De nouvelles métriques sont disponibles dans le groupe API `metrics.eks.amazonaws.com` .	11 novembre 2024

AWS mises à jour des politiques gérées	Politique AWS gérée mise à jour par EKS`AmazonEKSComputePolicy` . Autorisations de ressources mises à jour pour l'action `iam:AddRoleToInstanceProfile` .	7 novembre 2024
AWS mises à jour des politiques gérées	EKS a ajouté une nouvelle politique AWS gérée : `AmazonEKSComputePolicy`	1er novembre 2024
AWS mises à jour des politiques gérées	Ajout d'autorisations à `AmazonEKSClusterPolicy` . Ajout d'une autorisation `ec2:DescribeInstanceTopology` permettant à Amazon EKS d'associer des informations de topologie au nœud sous forme d'étiquettes.	1er novembre 2024
AWS mises à jour des politiques gérées	EKS a ajouté une nouvelle politique AWS gérée : `AmazonEKSBlockStoragePolicy`	30 octobre 2024
AWS mises à jour des politiques gérées	EKS a ajouté une nouvelle politique AWS gérée : `AmazonEKSLoadBalancingPolicy`	30 octobre 2024

AWS mises à jour des politiques gérées	Des `cloudwatch:PutMetricData` autorisations ont été ajoutées `AmazonEKSServiceRolePolicy` pour permettre à Amazon EKS de publier des métriques sur Amazon CloudWatch.	29 octobre 2024
AWS mises à jour des politiques gérées	EKS a ajouté une nouvelle politique AWS gérée : `AmazonEKSNetworkingPolicy`	28 octobre 2024
Terminaux double pile pour les nouveaux clusters `IPv6`	Connectez-vous à de nouveaux clusters `IPv6` avec un point de terminaison `eks-cluster.` *`region`*`.api.aws` double pile. Ce point de terminaison est renvoyé lorsque vous décrivez ces clusters. `kubectl` et d'autres clients de l'API Kubernetes dans des environnements `IPv4`, `IPv6` ou à double pile peuvent résoudre et se connecter à ces points de terminaison pour les clusters publics ou privés.	21 octobre 2024

AWS mises à jour des politiques gérées	Ajout de `autoscaling:ResumeProcesses` , `autoscaling:SuspendProcesses` et des autorisations associées à `AWSServiceRoleForAmazonEKSNodegroup` dans les régions de Chine afin d'assurer l'intégration avec le contrôleur de récupération d'application Amazon pour EKS. Aucun changement dans les autres régions.	21 octobre 2024
AL2023 accéléré AMIs	Vous pouvez désormais utiliser des instances accélérées `NVIDIA` et des instances AWS Neuron pour la AMIs base de. AL2023	11 octobre 2024
Nouveau format source	Nous sommes passés à un nouveau format source avec quelques modifications de mise en page. Nous sommes en train de résoudre certains problèmes de formatage mineurs temporaires.	10 octobre 2024

AWS mises à jour des politiques gérées	Ajout d'autorisations à `AmazonEKSServicePolicy` et `AmazonEKSServiceRolePolicy` . Ajout de `ec2:GetSecurityGroupsForVpc` et association d'autorisations de balises pour permettre à EKS de lire les informations relatives aux groupes de sécurité et de mettre à jour les balises associées.	10 octobre 2024
AWS mises à jour des politiques gérées - Nouvelle politique	EKS a ajouté une nouvelle politique AWS gérée.	3 octobre 2024
Version de Kubernetes `1.31`	Ajout de la prise en charge de la version Kubernetes `1.31` pour les nouveaux clusters et les mises à niveau de version.	24 septembre 2024
AWS mises à jour de politiques gérées - Mise à jour d'une politique existante	Amazon EKS a mis à jour une politique AWS gérée existante.	21 août 2024
La version 1.29 de Kubernetes est désormais disponible pour les clusters locaux sur Outposts AWS	Vous pouvez désormais créer un cluster local Amazon EKS sur un AWS Outposts à l'aide de Kubernetes version 1.29.	20 août 2024

EKS Pod Identity aux États-Unis AWS GovCloud	Les identités de pod Amazon EKS associent un rôle IAM à un compte de service Kubernetes. Grâce à cette fonction, vous n'avez plus besoin de fournir des autorisations étendues au rôle IAM du nœud. De cette façon, les pods de ce nœud peuvent appeler AWS APIs. Contrairement aux rôles IAM pour les comptes de service, les identités de pod EKS sont entièrement intégrées à EKS ; vous n'avez pas besoin d'un fournisseur d'identité OIDC.	14 août 2024
Mises à jour de contenu basées sur des scénarios	Nous avons renommé et mis à jour les rubriques afin qu'elles soient davantage axées sur des scénarios tout au long du guide.	9 août 2024
Points de terminaison d'interface VPC à double pile pour Amazon EKS	Vous pouvez désormais créer des points de terminaison d'interface VPC à double pile pour Amazon EKS avec à la fois des adresses IP `IPv4` et `IPv6` ainsi que des noms DNS.	7 août 2024

Nouveaux points de terminaison à double pile pour Amazon EKS APIs avec adresses IPv6	L'API EKS pour la création et la gestion des clusters, et l'émetteur OIDC URLs pour les clusters disposent de nouveaux points de terminaison à double pile. Le nouveau nom DNS de l'API Amazon EKS est `eks.region.api.aws` ; il se résout vers les adresses IPv4 et les adresses IPv6. Les nouveaux clusters disposent d'une nouvelle URL d'émetteur OIDC à double pile (`oidc-eks. region.api.aws`).	1er août 2024
Blocs de capacité pour les groupes de nœuds gérés	Vous pouvez désormais utiliser les blocs de capacité pour les groupes de nœuds gérés.	1er juillet 2024
Collecte des métriques du groupe Auto Scaling activée par défaut	Les groupes de nœuds gérés Amazon EKS disposent désormais des métriques du groupe Amazon EC2 Auto Scaling activées par défaut, sans frais supplémentaires. Auparavant, vous deviez effectuer plusieurs étapes pour activer cette fonctionnalité.	28 juin 2024
AWS mises à jour de politiques gérées - Mise à jour d'une politique existante	Amazon EKS a mis à jour une politique AWS gérée existante.	27 juin 2024

Version de Kubernetes `1.26`	La version Kubernetes `1.26` bénéficie désormais d'un support étendu.	12 juin 2024
Améliorations apportées aux références d'informations de l'AMI	Nous avons apporté des améliorations aux références d'informations AMI, en particuli er pour Bottlerocket.	12 juin 2024
Version de Kubernetes `1.30`	Ajout de la prise en charge de la version Kubernetes `1.30` pour les nouveaux clusters et les mises à niveau de version.	23 mai 2024
Auto Scaling CoreDNS	L'autoscaler CoreDNS adaptera dynamiquement le nombre de réplicas du déploiement CoreDNS dans un cluster EKS en fonction du nombre de nœuds et de cœurs de processeur. Cette fonctionnalité fonctionne pour CoreDNS `v1.9` ainsi que la dernière version de plateform e EKS `1.25` et les versions ultérieures.	14 mai 2024
Mise à jour de la version de plateforme Amazon EKS	Il s'agit d'une nouvelle version de plateforme qui comporte des correctifs et des améliorations de sécurité. Cela inclut les nouvelles versions correctives de Kubernetes `1.29.4`, `1.28.9`, et `1.27.13`.	14 mai 2024

CloudWatch Support de Container Insights pour Windows	Le module complémentaire Amazon CloudWatch Observability Operator autorise désormais également Container Insights sur les nœuds de travail Windows du cluster.	10 avril 2024
Concepts Kubernetes	Ajout d'une nouvelle rubrique consacrée aux concepts Kubernetes.	5 avril 2024
Restructurer l'accès et le contenu IAM	Déplacer les pages existantes liées aux thèmes de l'accès et de la gestion des identités et des accès (IAM), telles que la carte de configuration d'authentification, les entrées d'accès, l'ID de pod et l'IRSA, vers une nouvelle section. Réviser le contenu de l'aperçu.	2 avril 2024
Prise en charge du système d'exploitation Bottlerocket pour le pilote CSI Amazon S3	Le pilote CSI Mountpoint pour Amazon S3 est désormais compatible avec Bottlerocket.	13 mars 2024
AWS mises à jour de politiques gérées - Mise à jour d'une politique existante	Amazon EKS a mis à jour une politique AWS gérée existante.	4 mars 2024
Amazon Linux 2023	Amazon Linux 2023 (AL2023) est un nouveau système d'exploitation basé sur Linux conçu pour fournir un environnement sécurisé, stable et performant pour vos applications cloud.	29 février 2024

EKS Pod Identity et IRSA prennent en charge les sidecars dans Kubernetes 1.29	Dans Kubernetes 1.29, les conteneurs sidecar sont disponibles dans les clusters Amazon EKS. Les conteneurs Sidecar sont pris en charge avec les rôles IAM pour les comptes de service ou EKS Pod Identity. Pour plus d'informations sur les sidecars, consultez la section Conteneurs sidecar dans la documentation Kubernetes.	26 février 2024
Version de Kubernetes 1.29	Ajout de la prise en charge de la version Kubernetes 1.29 pour les nouveaux clusters et les mises à niveau de version.	23 janvier 2024
Communiqué complet : Assistance étendue Amazon EKS pour les versions Kubernetes	La prise en charge étendue des versions Kubernetes vous permet de conserver une version spécifique de Kubernetes pendant plus de 14 mois.	16 janvier 2024

Détection de l'état du cluster Amazon EKS dans le AWS cloud	Amazon EKS détecte les problèmes liés à vos clusters Amazon EKS et à l'infrastructure des prérequis du cluster et les enregistre dans l'état du cluster. Vous pouvez consulter les problèmes liés à vos clusters EKS dans AWS Management Console et dans `health` le cluster dans l'API EKS. Ces problèmes s'ajoutent aux problèmes détectés et affichés par la console. Auparavant, l'état de santé des clusters n'était disponible que pour les clusters locaux sur AWS Outposts.	28 décembre 2023
Informations sur les clusters	Vous pouvez désormais obtenir des recommandations sur votre cluster sur la base de vérifications récurrentes.	20 décembre 2023
Expansion de AWS la région Amazon EKS	Amazon EKS est désormais disponible dans la AWS région Ouest du Canada (Calgary`ca-west-1`) ().	20 décembre 2023

Vous pouvez désormais accorder aux utilisateurs et rôles IAM l'accès à votre cluster à l'aide d'entrées d'accès	Avant l'introduction des entrées d'accès, vous avez accordé aux utilisateurs et rôles IAM l'accès à votre cluster en ajoutant des entrées à la `ConfigMap aws-auth`. Chaque cluster dispose désormais d'un mode d'accès, et vous pouvez passer à l'utilisation des entrées d'accès selon votre calendrier. Après avoir changé de mode, vous pouvez ajouter des utilisateurs en ajoutant des entrées d'accès dans la AWS CLI AWS CloudFormation, et le AWS SDKs.	18 décembre 2023
Mise à jour de la version de plateforme Amazon EKS	Il s'agit d'une nouvelle version de plateforme qui comporte des correctifs et des améliorations de sécurité. Cela inclut les nouvelles versions de correctif de Kubernetes `1.28.4`, `1.27.8`, `1.26.11` et `1.25.16`.	12 décembre 2023
Mountpoint pour le pilote CSI d’Amazon S	Vous pouvez désormais installer le pilote Mountpoint for Amazon S3 CSI sur les clusters Amazon EKS.	27 novembre 2023

Activer les métriques Prometheus lors de la création d’un cluster	Dans le AWS Management Console, vous pouvez désormais activer les métriques Prometheus lors de la création d'un cluster. Vous pouvez également consulter les détails du scraper Prometheus dans l’onglet Observabilité.	26 novembre 2023
Identités du pod Amazon EKS	Les identités de pod Amazon EKS associent un rôle IAM à un compte de service Kubernetes. Grâce à cette fonction, vous n'avez plus besoin de fournir des autorisations étendues au rôle IAM du nœud. De cette façon, les pods de ce nœud peuvent appeler AWS APIs. Contrairement aux rôles IAM pour les comptes de service, les identités de pod EKS sont entièrement intégrées à EKS ; vous n’avez pas besoin d’un fournisseur d’identité OIDC.	26 novembre 2023
AWS mises à jour de politiques gérées - Mise à jour d'une politique existante	Amazon EKS a mis à jour une politique AWS gérée existante.	26 novembre 2023
Contrôleur d'instantané CSI	Vous pouvez désormais installer le contrôleur d’instantané CSI pour l’utiliser avec des pilotes CSI compatibles, tels que le pilote CSI Amazon EBS.	17 novembre 2023

Réécriture des rubriques de l'opérateur ADOT	La section relative à la prise en charge du module complémentaire Amazon EKS pour ADOT Operator était redondante avec le AWS Distro pour ce qui est de la documentation. OpenTelemetry Nous avons migré les informations essentielles restantes vers cette ressource afin de réduire les informations obsolètes et incohérentes.	14 novembre 2023
Prise en charge de l'add-on CoreDNS EKS pour les métriques Prometheus	Les versions `v1.10.1-eksbuild.5` , `v1.9.3-eksbuild.9` , et `v1.8.7-eksbuild.8` de l'add-on EKS pour CoreDNS exposent le port sur lequel CoreDNS a publié les métriques, dans le service `kube-dns`. Cela facilite l'inclusion des métriques CoreDNS dans vos systèmes de surveillance.	10 novembre 2023
Module complémentaire Amazon EKS CloudWatch Observability Operator	Ajout de la page Amazon EKS CloudWatch Observability Operator.	6 novembre 2023
Blocs de capacité pour les instances P5 autogérées aux USA Est (Ohio)	Aux USA Est (Ohio), vous pouvez désormais utiliser des blocs de capacité pour les instances P5 autogérées.	31 octobre 2023

Les clusters prennent en charge des sous-réseaux et des groupes de sécurité	Vous pouvez mettre à jour le cluster pour modifier les sous-réseaux et les groupes de sécurité qu'il utilise. Vous pouvez effectuer la AWS Management Console mise à jour à partir de la dernière version de la AWS CLI et de `eksctl` la version `v0.164.0-rc.0` ou ultérieur e. AWS CloudFormation Vous pouvez avoir besoin de procéder ainsi pour fournir aux sous-réseaux davantage d'adresses IP disponibles afin de réussir la mise à niveau d'une version de cluster.	24 octobre 2023
Le rôle de cluster et le rôle de groupe de nœuds gérés prennent en charge les politiques AWS d'Identity and Access Management gérées par le client	Vous pouvez utiliser une politique IAM personnalisée sur le rôle de cluster, au lieu de la EKSClusterpolitique AWS gérée par Amazon Policy. Vous pouvez également utiliser une politique IAM personnalisée sur le rôle du nœud dans un groupe de nœuds gérés, au lieu de la politique EKSWorker NodePolicy AWS gérée par Amazon. Cela permet de créer une politique avec le moins de privilèges possible afin de répondre à des exigences strictes en matière de conformité.	23 octobre 2023

Correction du lien vers l'installation d'eksctl	Correction du lien d'installation d'eksctl après que la page ait été déplacée.	6 octobre 2023
Version préliminaire : prise en charge étendue d'Amazon EKS pour les versions Kubernetes	La prise en charge étendue des versions Kubernetes vous permet de conserver une version spécifique de Kubernetes pendant plus de 14 mois.	4 octobre 2023
Supprimer les références à AWS l'intégration d'App Mesh	Les intégrations d'Amazon EKS avec AWS App Mesh restent réservées aux clients existants d'App Mesh.	29 septembre 2023
Version de Kubernetes `1.28`	Ajout de la prise en charge de la version Kubernetes `1.28` pour les nouveaux clusters et les mises à niveau de version.	26 septembre 2023
Les clusters existants prennent en charge l'application des politiques réseau Kubernetes dans le plug-in Amazon VPC CNI pour Kubernetes	Vous pouvez utiliser la politique réseau Kubernetes dans les clusters existants avec le plug-in Amazon VPC CNI pour Kubernetes, sans avoir besoin d'une solution tierce. Vous pouvez utiliser la politique réseau Kubernetes dans les clusters existants avec le plug-in Amazon VPC CNI pour Kubernetes, sans avoir besoin d'une solution tierce.	15 septembre 2023

Le module complémentaire CoreDNS Amazon EKS prend en charge la modification des PDB	Vous pouvez modifier le `PodDisruptionBudget` l'add-on EKS pour CoreDNS dans les versions `v1.9.3-eksbuild.7` et ultérieures et `v1.10.1-eksbuild.4` et ultérieures.	15 septembre 2023
Prise en charge d'Amazon EKS pour les sous-réseaux partagés	Nouvelles exigences et considérations relatives aux sous-réseaux partagéspour créer des clusters Amazon EKS dans des sous-réseaux partagés.	7 septembre 2023
Mises à jour de Qu'est-ce qu'Amazon EKS ?	Ajout de nouveaux cas d'utilisation courants et de nouvelles rubriques Architecture. Autres rubriques actualisées.	6 septembre 2023
Application des politiques réseau Kubernetes dans le plug-in Amazon VPC CNI pour Kubernetes	Vous pouvez utiliser la politique réseau Kubernetes avec le plug-in Amazon VPC CNI pour Kubernetes, au lieu d'avoir recours à une solution tierce. Vous pouvez utiliser la politique réseau Kubernetes avec le plug-in Amazon VPC CNI pour Kubernetes, au lieu d'avoir recours à une solution tierce.	29 août 2023
Expansion de AWS la région Amazon EKS	Amazon EKS est désormais disponible dans la région Israël (Tel Aviv) `il-central-1` (AWS).	1er août 2023

Stockage éphémère configurable pour Fargate	Vous avez la possibilité d'augmenter la capacité totale de stockage éphémère pour chaque pod en cours d'exécution sur Amazon EKS Fargate.	31 juillet 2023
Prise en charge du module complémentaire pour le pilote CSI d'Amazon EFS	Vous pouvez désormais utiliser la AWS Management Console AWS CLI et l'API pour gérer le pilote Amazon EFS CSI.	26 juillet 2023
AWS mises à jour des politiques gérées - Nouvelle politique	Amazon EKS a ajouté une nouvelle politique AWS gérée.	26 juillet 2023
Les mises à jour des versions 1.27, 1.26, 1.25 et 1.24 de Kubernetes sont désormais disponibles pour les clusters locaux sur Outposts AWS	Les mises à jour des versions 1.27.3, 1.26.6, 1.25.11 et 1.24.15 de Kubernetes sont désormais disponibles pour les clusters locaux sur Outposts AWS	20 juillet 2023
Prise en charge des préfixes IP pour les nœuds Windows	L'attribution de préfixes IP à vos nœuds vous permet d'héberger un nombre nettement plus élevé de pods sur vos nœuds que lorsque vous attribuez des adresses IP secondaires individuelles à vos nœuds.	6 juillet 2023
Pilote FSx CSI Amazon pour OpenZFS	Vous pouvez désormais installer le pilote CSI Amazon FSx pour OpenZFS sur les clusters Amazon EKS.	30 juin 2023

Les pods sur les nœuds Linux dans les clusters IPv4 peuvent désormais communiquer avec les points de terminaison IPv6.	Après avoir attribué une IPv6 adresse à votre nœud, l'IPv4adresse de vos pods est l'adresse réseau traduite en IPv6 adresse du nœud sur lequel ils s'exécutent.	19 juin 2023
Groupes de nœuds gérés par Windows dans les régions AWS GovCloud (États-Unis)	Dans les régions AWS GovCloud (États-Unis), les groupes de nœuds gérés par Amazon EKS peuvent désormais exécuter des conteneurs Windows.	30 mai 2023
Version de Kubernetes 1.27	Ajout de la prise en charge de la version Kubernetes 1.27 pour les nouveaux clusters et les mises à niveau de version.	24 mai 2023
Version de Kubernetes 1.26	Ajout de la prise en charge de la version Kubernetes 1.26 pour les nouveaux clusters et les mises à niveau de version.	11 avril 2023
Domainless gMSA	Vous pouvez désormais utiliser gMSA sans domaine avec Windows Pods.	27 mars 2023
Expansion de AWS la région Amazon EKS	Amazon EKS est désormais disponible dans la AWS région Asie-Pacifique (Melbourneap-southeast-4) ().	10 mars 2023
Pilote CSI Amazon File Cache	Vous pouvez désormais installer le pilote CSI Amazon File Cache sur les clusters Amazon EKS.	3 mars 2023

La version 1.25 de Kubernetes est désormais disponible pour les clusters locaux sur Outposts AWS	Vous pouvez désormais créer un cluster local Amazon EKS sur un Outpost à l'aide des versions Kubernetes `1.22` – `1.25`.	1er mars 2023
Version de Kubernetes `1.25`	Ajout de la prise en charge de la version Kubernetes `1.25` pour les nouveaux clusters et les mises à niveau de version.	22 février 2023
AWS mises à jour de politiques gérées - Mise à jour d'une politique existante	Amazon EKS a mis à jour une politique AWS gérée existante.	7 février 2023
Expansion de AWS la région Amazon EKS	Amazon EKS est désormais disponible dans les régions Asie-Pacifique (Hyderabad) (`ap-south-2`), Europe (Zurich) (`eu-central-2`) et Europe (Espagne) (`eu-south-2`). AWS	6 février 2023
Versions de Kubernetes `1.21` : `1.24` elles sont désormais disponibles pour les clusters locaux sur Outposts. AWS	Vous pouvez désormais créer un cluster local Amazon EKS sur un Outpost à l'aide des versions Kubernetes `1.21` – `1.24`. Auparavant, seule la version `1.21` était disponible.	17 janvier 2023
Amazon EKS prend désormais en charge AWS PrivateLink	Vous pouvez utiliser an AWS PrivateLink pour créer une connexion privée entre votre VPC et Amazon EKS.	16 décembre 2022

Prise en charge par Windows des groupes de nœuds gérés	Vous pouvez désormais utiliser Windows pour les groupes de nœuds gérés Amazon EKS.	15 décembre 2022
Les modules complémentaires Amazon EKS proposés par des éditeurs de logiciels indépendants sont désormais disponibles AWS sur le Marketplace	Vous pouvez désormais parcourir les modules complémentaires Amazon EKS proposés par des fournisseurs de logiciels indépendants et vous y abonner via le AWS Marketplace.	28 novembre 2022
AWS mises à jour de politiques gérées - Mise à jour d'une politique existante	Amazon EKS a mis à jour une politique AWS gérée existante.	17 novembre 2022
Version de Kubernetes `1.24`	Ajout de la prise en charge de la version Kubernetes `1.24` pour les nouveaux clusters et les mises à niveau de version.	15 novembre 2022
Expansion de AWS la région Amazon EKS	Amazon EKS est désormais disponible dans la AWS région Moyen-Orient (Émirats arabes unis`me-central-1`) ().	3 novembre 2022
AWS mises à jour de politiques gérées - Mise à jour d'une politique existante	Amazon EKS a mis à jour une politique AWS gérée existante.	24 octobre 2022
AWS mises à jour de politiques gérées - Mise à jour d'une politique existante	Amazon EKS a mis à jour une politique AWS gérée existante.	20 octobre 2022

Les clusters locaux sur AWS Outposts sont désormais disponibles	Vous pouvez désormais créer un cluster local Amazon EKS sur un Outpost.	19 septembre 2022
Quotas basés sur les vCPU Fargate	Fargate effectue une transition des quotas basés sur les pods vers des quotas basés sur les vCPU.	8 septembre 2022
AWS mises à jour de politiques gérées - Mise à jour d'une politique existante	Amazon EKS a mis à jour une politique AWS gérée existante.	31 août 2022
Suivi des coûts	Amazon EKS prend désormais en charge Kubecost, qui vous permet de surveiller les coûts ventilés par ressources Kubernetes, notamment les pods, les nœuds, les espaces de noms et les étiquettes.	24 août 2022
AWS mises à jour des politiques gérées - Nouvelle politique	Amazon EKS a ajouté une nouvelle politique AWS gérée.	24 août 2022
AWS mises à jour des politiques gérées - Nouvelle politique	Amazon EKS a ajouté une nouvelle politique AWS gérée.	23 août 2022
Balisage des ressources pour la facturation	Ajout de la prise en charge de balises de répartition des coûts généré par `aws:eks:cluster-name` pour tous les clusters.	16 août 2022

Caractères génériques du profil de Fargate	Ajout de la prise en charge des caractères génériques du profil Fargate dans les critères de sélection pour les espaces de noms, les clés d'étiquette et les valeurs d'étiquette.	16 août 2022
Version de Kubernetes `1.23`	Ajout de la prise en charge de la version Kubernetes `1.23` pour les nouveaux clusters et les mises à niveau de version.	11 août 2022
Consultez les ressources Kubernetes dans le AWS Management Console	Vous pouvez désormais afficher les informations relatives aux ressources Kubernetes déployées sur votre cluster à l'aide de la commande AWS Managemen t Console.	3 mai 2022
Expansion de AWS la région Amazon EKS	Amazon EKS est désormais disponible dans la AWS région Asie-Pacifique (Jakartaap-`southeast-3`) ().	2 mai 2022
Prise en charge de la page Observability et du module complémentaire ADOT	Ajout d'une page d'observa bilité et d' AWS une distribut ion pour OpenTelemetry (ADOT).	21 avril 2022
Version de Kubernetes `1.22`	Ajout de la prise en charge de la version Kubernetes `1.22` pour les nouveaux clusters et les mises à niveau de version.	4 avril 2022
AWS mises à jour des politiques gérées - Nouvelle politique	Amazon EKS a ajouté une nouvelle politique AWS gérée.	4 avril 2022

Ajout de détails sur les correctifs de pods Fargate	Lors de la mise à niveau des pods Fargate, Amazon EKS essaie d'abord d'expulser les pods en fonction des budgets d'interruption de votre pod. Vous pouvez créer des règles d'événements pour réagir aux expulsions ayant échoué avant la suppression des pods.	1er avril 2022
Version complète : prise en charge du module complémentaire pour le pilote CSI d'Amazon EBS	Vous pouvez désormais utiliser la AWS Management Console AWS CLI et l'API pour gérer le pilote Amazon EBS CSI.	31 mars 2022
AWS Mise à jour du contenu d'Outposts	Instructions pour déployer un cluster Amazon EKS sur AWS Outposts.	22 mars 2022
AWS mises à jour de politiques gérées - Mise à jour d'une politique existante	Amazon EKS a mis à jour une politique AWS gérée existante.	21 mars 2022
Prise en charge de Windows `containerd`	Vous pouvez désormais sélectionner l'exécution `containerd` pour les nœuds Windows.	14 mars 2022
Ajout de considérations Amazon EKS Connector à la documentation de sécurité	Décrit le modèle de responsabilité partagée en ce qui concerne les clusters connectés.	25 février 2022

Attribuer des adresses `IPv6` à vos pods et services	Vous pouvez désormais créer un cluster de version `1.21` ou ultérieure qui attribue des adresses `IPv6` à vos pods et services.	6 janvier 2022
AWS mises à jour de politiques gérées - Mise à jour d'une politique existante	Amazon EKS a mis à jour une politique AWS gérée existante.	13 décembre 2021
Version préliminaire : prise en charge du module complémentaire pour le pilote CSI d'Amazon EBS	Vous pouvez désormais prévisualiser à l' AWS Management Console aide de la AWS CLI et de l'API pour gérer le pilote Amazon EBS CSI.	9 décembre 2021
Prise en charge de Karpenter Autoscaler	Vous pouvez désormais utiliser le projet open source Karpenter pour mettre à l'échelle automatique vos nœuds.	29 novembre 2021
Prise en charge du filtre Kubernetes Fluent Bit dans la journalisation Fargate	Vous pouvez désormais utiliser le filtre Kubernetes Fluent Bit avec la journalisation Fargate.	10 novembre 2021
Prise en charge de Windows disponible dans le plan de contrôle	La prise en charge de Windows est désormais disponible dans votre plan de contrôle. Vous n'avez plus besoin de l'activer dans votre plan de données.	9 novembre 2021

Bottlerocket ajouté en tant que type d'AMI pour les groupes de nœuds gérés	Auparavant, Bottlerocket était uniquement disponible en tant qu'option de nœud autogéré. Il peut désormais être configuré comme un groupe de nœuds gérés, ce qui réduit l'effort nécessaire pour répondre aux exigences de conformité des nœuds.	28 octobre 2021
DL1 assistance au conducteur	Amazon Linux personnalisé prend AMIs désormais en charge les charges de travail d'apprentissage profond pour Amazon Linux 2. Cette activation permet une configuration de base générique sur site ou dans le cloud.	25 octobre 2021
VT1 support vidéo	Amazon Linux personnalisé AMIs est désormais compatible VT1 avec certaines distributions. Cette mise en œuvre annonce les dispositifs Xilinx U30 sur votre cluster Amazon EKS.	13 septembre 2021
Amazon EKS Connector est désormais disponible	Vous pouvez utiliser le connecteur Amazon EKS pour enregistrer et connecter tout cluster Kubernetes conforme à la console Amazon EKS AWS et le visualiser dans la console Amazon EKS.	8 septembre 2021

Amazon EKS Anywhere est désormais disponible	Amazon EKS Anywhere est une nouvelle option de déploiement pour Amazon EKS que vous pouvez utiliser pour créer et exploiter facilement des clusters Kubernetes sur site.	8 septembre 2021
Pilote Amazon FSx pour NetApp ONTAP CSI	Ajout d'une rubrique qui résume le pilote Amazon FSx for NetApp ONTAP CSI et fournit des liens vers d'autres références.	2 septembre 2021
Les groupes de nœuds gérés calculent désormais automatiquement le nombre maximal de pods recommandés par Amazon EKS pour les nœuds	Les groupes de nœuds gérés calculent désormais automatiquement le nombre maximal de pods Amazon EKS pour les nœuds que vous déployez sans modèle de lancement ou avec un modèle de lancement dans lequel vous n'avez pas spécifié d'ID d'AMI.	30 août 2021
Supprimer la gestion Amazon EKS des paramètres du module complémentaire sans supprimer le logiciel complémentaire Amazon EKS	Vous pouvez désormais supprimer un module complémentaire Amazon EKS sans supprimer le logiciel complémentaire de votre cluster.	20 août 2021
Créer des pods multi-résidents à l'aide de Multus	Vous pouvez maintenant ajouter plusieurs interfaces réseau à un pod à l'aide de Multus.	2 août 2021

Ajouter d'autres adresses IP à vos nœuds Linux Amazon EC2	Vous pouvez maintenant ajouter beaucoup plus d'adresses IP à vos nœuds Linux Amazon EC2. Cela signifie que vous pouvez exécuter une densité plus élevée de pods sur chaque nœud. Vous pouvez maintenant ajouter beaucoup plus d'adresses IP à vos nœuds Linux Amazon EC2. Cela signifie que vous pouvez exécuter une densité plus élevée de pods sur chaque nœud.	27 Juillet 2021
Amorçage d'exécution `containerd`	L'Amazon Machine Image (AMI) accélérée optimisée pour Amazon EKS contient désormais un indicateur bootstrap que vous pouvez utiliser pour activer l'`containerd` exécution dans Amazon EKS Optimized et AMIs Bottlerocket. Cet indicateur est disponible dans toutes les versions de Kubernetes prises en charge de l'AMI.	19 juillet 2021
Version de Kubernetes `1.21`	Support Kubernetes version `1.21` ajouté.	19 juillet 2021

Ajout d'une rubrique de politiques gérées	Liste de toutes les politiques gérées par Amazon EKS IAM et des modifications qui leur ont été apportées depuis le 17 juin 2021.	17 juin 2021
Utiliser des groupes de sécurité pour les pods avec Fargate	Vous pouvez désormais utiliser les groupes de sécurité pour les pods avec Fargate, en plus de les utiliser avec les nœuds Amazon EC2.	1er juin 2021
Ajout de CoreDNS et de modules complémentaires `kube-proxy` Amazon EKS	Amazon EKS peut désormais vous aider à gérer les CoreDNS et les modules complémentaires Amazon EKS `kube-proxy` pour votre cluster.	19 mai 2021
Version de Kubernetes `1.20`	Ajout de la prise en charge de la version Kubernetes `1.20` pour les nouveaux clusters et les mises à niveau de version.	18 mai 2021
AWS Lancement du Load Balancer Controller `2.2.0`	Vous pouvez désormais utiliser le AWS Load Balancer Controller pour créer des Elastic Load Balancers à l'aide d'instances ou de cibles IP.	14 mai 2021
Taints de noeuds pour les groupes de nœuds gérés	Amazon EKS prend désormais en charge l'ajout de taints de notes aux groupes de nœuds gérés.	11 mai 2021

Chiffrement des secrets pour les clusters existants	Amazon EKS prend désormais en charge l'ajout du chiffrement des secrets aux clusters existants.	26 février 2021
Version de Kubernetes `1.19`	Ajout de la prise en charge de la version Kubernetes `1.19` pour les nouveaux clusters et les mises à niveau de version.	16 février 2021
Amazon EKS prend désormais en charge les fournisseurs d'identité OpenID Connect (OIDC) comme méthode d'authentification des utilisateurs d'un cluster de version `1.16` ou ultérieure.	Les fournisseurs d'identité OIDC peuvent être utilisés avec ou comme alternative à AWS Identity and Access Management (IAM).	12 février 2021
Afficher les ressources du nœud et de la charge de travail dans AWS Management Console	Vous pouvez désormais consulter les détails de vos nœuds gérés, autogérés et Fargate ainsi que de vos charges de travail Kubernetes déployées dans le. AWS Management Console	1er décembre 2020
Déployer des types d'instances Spot dans un groupe de nœuds gérés	Vous pouvez désormais déployer plusieurs types d'instances Spot ou à la demande dans un groupe de nœuds gérés.	1er décembre 2020

Amazon EKS peut désormais gérer des modules complémentaires spécifiques pour votre cluster	Vous pouvez gérer vous-même les modules complémentaires ou autoriser Amazon EKS à contrôler le lancement et la version d'un module complémentaire via l'API Amazon EKS.	1er décembre 2020
Partager un ALB sur plusieurs entrées	Vous pouvez désormais partager un AWS Application Load Balancer (ALB) entre plusieurs entrées Kubernetes. Auparavant, vous deviez déployer un ALB distinct pour chaque entrée.	23 octobre 2020
Prise en charge des cibles IP NLB	Vous pouvez désormais déployer un Network Load Balancer avec des cibles IP. Cela signifie que vous pouvez utiliser un NLB pour équilibrer la charge du trafic réseau vers les pods Fargate et directement vers les pods qui fonctionnent sur les nœuds Amazon EC2.	23 octobre 2020
Version de Kubernetes `1.18`	Ajout de la prise en charge de la version Kubernetes `1.18` pour les nouveaux clusters et les mises à niveau de version.	13 octobre 2020
Spécifiez un bloc CIDR personnalisé pour l'attribution de l'adresse IP du service Kubernetes.	Vous pouvez désormais spécifier un bloc CIDR personnalisé à partir duquel Kubernetes attribue les adresses IP de service.	29 septembre 2020

Attribuer des groupes de sécurité à des pods individuels	Vous pouvez maintenant associer différents groupes de sécurité à certains des pods individuels qui fonctionnent sur de nombreux types d'instanc es Amazon EC2.	9 septembre 2020
Déployer Bottlerocket sur vos nœuds	Vous pouvez désormais déployer des nœuds qui exécutent Bottlerocket.	31 août 2020
La possibilité de lancer des nœuds Arm est généralement disponible	Vous pouvez maintenant lancer des nœuds Arm dans des groupes de nœuds gérés et autogérés.	17 août 2020
Modèles de lancement de groupes de nœuds gérés et AMI personnalisée	Vous pouvez maintenant déployer un groupe de nœuds géré qui utilise un modèle de lancement Amazon EC2. Le modèle de lancement peut spécifier une AMI personnal isée, si vous le souhaitez.	17 août 2020
Support EFS pour AWS Fargate	Vous pouvez désormais utiliser Amazon EFS avec AWS Fargate.	17 août 2020

Mise à jour de la version de plateforme Amazon EKS	Il s'agit d'une nouvelle version de plateforme qui comporte des correctifs et des améliorations de sécurité. Cela inclut la prise en charge UDP pour les services de type `LoadBalancer` lors de l'utilisation de Network Load Balancers avec la version Kubernetes `1.15` ou ultérieure. Pour plus d'informations, consultez le problème Allow UDP for AWS Network Load Balancer sur. GitHub	12 août 2020
Expansion de AWS la région Amazon EKS	Amazon EKS est maintenant disponible dans les régions AWS Afrique (Le Cap) (`af-south-1`) et Europe (Milan) (`eu-south-1`).	6 août 2020
Métriques d'utilisation de Fargate	AWS Fargate CloudWatch fournit des statistiques d'utilisation qui fournissent une visibilité sur l'utilisation des ressources Fargate On-Demand par votre compte.	3 août 2020
Version de Kubernetes `1.17`	Ajout de la prise en charge de la version Kubernetes `1.17` pour les nouveaux clusters et les mises à niveau de version.	10 juillet 2020

Créer et gérer des ressources App Mesh depuis Kubernetes avec le contrôleur App Mesh for Kubernetes	Vous pouvez créer et gérer des ressources App Mesh depuis Kubernetes. Le contrôleur injecte également automatiquement le proxy Envoy et les conteneurs init dans les pods que vous déployez.	18 juin 2020
Amazon EKS prend désormais en charge les nœuds Inf1 Amazon EC2	Vous pouvez ajouter des nœuds Amazon EC2 Inf1 à votre cluster.	4 juin 2020
Expansion de AWS la région Amazon EKS	Amazon EKS est désormais disponible dans les AWS GovCloud régions (USA Est) (`us-gov-east-1`) et AWS GovCloud (USA Ouest) (`us-gov-west-1`). AWS	13 mai 2020
Kubernetes `1.12` n'est plus pris en charge sur Amazon EKS	La version Kubernetes `1.12` n'est plus prise en charge sur Amazon EKS. Veuillez mettre à jour les clusters `1.12` vers la version `1.13` ou une version ultérieure pour éviter toute interruption de service.	12 mai 2020
Version de Kubernetes `1.16`	Ajout de la prise en charge de la version Kubernetes `1.16` pour les nouveaux clusters et les mises à niveau de version.	30 avril 2020
Ajout du rôle lié au service AWSServiceRoleForAmazonEKS	Le rôle lié au service AWSServiceRoleForAmazonEKS a été ajouté.	16 avril 2020

Version de Kubernetes `1.15`	Ajout de la prise en charge de la version Kubernetes `1.15` pour les nouveaux clusters et les mises à niveau de version.	10 mars 2020
Expansion de AWS la région Amazon EKS	Amazon EKS est désormais disponible dans les régions de Pékin AWS (`cn-north-1`) et Ningxia (`cn-northwest-1`).	26 février 2020
FSx pour le pilote Lustre CSI	Ajout d'une rubrique pour l'installation du pilote CSI FSx for Lustre sur les clusters Kubernetes `1.14` Amazon EKS.	23 décembre 2019
Restreindre l'accès réseau au point de terminaison d'accès public d'un cluster	Avec cette mise à jour, vous pouvez utiliser Amazon EKS pour restreindre les plages CIDR qui peuvent communiquer avec le point de terminaison d'accès public du serveur API Kubernetes.	20 décembre 2019
Résoudre l'adresse du point de terminaison de l'accès privé pour un cluster depuis l'extérieur d'un VPC	Avec cette mise à jour, vous pouvez utiliser Amazon EKS pour résoudre le point de terminaison d'accès privé du serveur API Kubernetes depuis l'extérieur d'un VPC.	13 décembre 2019
(Bêta) Nœuds d'instance Amazon EC2 A1 Amazon EC2	Lancez les nœuds d'instance Amazon EC2 Amazon EC2 A1 qui s'enregistrent dans votre cluster Amazon EKS.	4 décembre 2019

Création d'un cluster sur AWS Outposts	Amazon EKS prend désormais en charge la création de clusters sur AWS Outposts.	3 décembre 2019
AWS Fargate sur Amazon EKS	Les clusters Amazon EKS Kubernetes prennent désormais en charge les pods exécutée sur Fargate.	3 décembre 2019
Expansion de AWS la région Amazon EKS	Amazon EKS est désormais disponible dans la AWS région Canada (Centre`ca-centra l-1`) ().	21 novembre 2019
Groupes de nœuds gérés	Les groupes de nœuds gérés Amazon EKS automatisent l'approvisionnement et la gestion du cycle de vie des nœuds (instances Amazon EC2) pour les clusters Amazon EKS Kubernetes.	18 novembre 2019
Mise à jour de la version de plateforme Amazon EKS	Nouvelles versions de plateforme pour résoudre CVE-2019-11253.	6 novembre 2019
Kubernetes `1.11` n'est plus pris en charge sur Amazon EKS	La version Kubernetes `1.11` n'est plus prise en charge sur Amazon EKS. Veuillez mettre à jour les clusters `1.11` vers la version `1.12` ou ultérieure pour éviter toute interruption de service.	4 novembre 2019
Expansion de AWS la région Amazon EKS	Amazon EKS est désormais disponible dans la AWS région Amérique du Sud (São Paulo`sa-east-1`) ().	16 octobre 2019

Prise en charge de Windows	Les clusters Amazon EKS exécutant Kubernetes version `1.14` prennent désormais en charge les applications Windows.	7 octobre 2019
Autoscaling	Ajout d'un chapitre pour couvrir certains des différents types de scalabilité automatique Kubernetes pris en charge dans les clusters Amazon EKS.	30 septembre 2019
Mise à jour du tableau de bord Kubernetes	Rubrique mise à jour pour l'installation du tableau de bord Kubernetes dans les clusters Amazon EKS afin d'utiliser la version bêta `2.0`.	28 septembre 2019
Pilote CSI Amazon EFS	Ajout d'une rubrique consacrée à l'installation du pilote CSI Amazon EFS sur les clusters Kubernetes `1.14` Amazon EKS.	19 septembre 2019
Paramètre Amazon EC2 Systems Manager pour l'ID d'AMI optimisée pour Amazon EKS	Ajout d'une rubrique pour la récupération de l'ID d'AMI optimisée pour Amazon EKS à l'aide d'un Amazon EC2 Systems Manager. Ce paramètre vous évite d'avoir à rechercher l'AMI IDs.	18 septembre 2019
Étiquetage des ressources Amazon EKS	Vous pouvez gérer l'identification de vos clusters Amazon EKS.	16 septembre 2019

Pilote CSI Amazon EBS	Ajout d'une rubrique pour l'installation du pilote CSI Amazon EBS dans les clusters Kubernetes `1.14` Amazon EKS.	9 septembre 2019
Nouvelle AMI optimisée pour Amazon EKS corrigée pour `CVE-2019-9512` et `CVE-2019-9514`	Amazon EKS a mis à jour l'AMI optimisée pour Amazon EKS pour résoudre CVE-2019-9512 et CVE-2019-9514.	6 septembre 2019
Annonce de l'obsolescence de Kubernetes `1.11` dans Amazon EKS	Amazon EKS a cessé de prendre en charge la version Kubernetes `1.11` le 4 novembre 2019.	4 septembre 2019
Version de Kubernetes `1.14`	Ajout de la prise en charge de la version Kubernetes `1.14` pour les nouveaux clusters et les mises à niveau de version.	3 septembre 2019
Rôles IAM pour les comptes de service	Avec les rôles IAM pour les comptes de service sur les clusters Amazon EKS, vous pouvez associer un rôle IAM à un compte de service Kubernetes. Grâce à cette fonction, vous n'avez plus besoin de fournir des autorisations étendues au rôle IAM du nœud. De cette façon, les pods de ce nœud peuvent appeler AWS APIs.	3 septembre 2019

Expansion de AWS la région Amazon EKS	Amazon EKS est désormais disponible dans la région Moyen-Orient (Bahreïn) `me-south-1` (AWS).	29 août 2019
Mise à jour de la version de plateforme Amazon EKS	Nouvelles versions de plateforme pour résoudre CVE-2019-9512 et CVE-2019-9514.	28 août 2019
Mise à jour de la version de plateforme Amazon EKS	Nouvelles versions de plateforme pour résoudre CVE-2019-11247 et CVE-2019-11249.	5 août 2019
Extension de région Amazon EKS	Amazon EKS est désormais disponible dans la AWS région Asie-Pacifique (Hong Kong`ap-east-1`) ().	31 juillet 2019
Kubernetes version `1.10` n'est plus pris en charge sur Amazon EKS	La version Kubernetes `1.10` n'est plus prise en charge sur Amazon EKS. Mettez à jour tous les clusters `1.10` vers la version `1.11` ou une version ultérieure pour éviter toute interruption de service.	30 juillet 2019
Ajout d'une rubrique sur le contrôleur d'entrée ALB	Le contrôleur d'entrée AWS ALB pour Kubernetes est un contrôleur qui provoque la création d'un ALB lors de la création de ressources d'entrée.	11 juillet 2019
Nouvelle AMI optimisée pour Amazon EKS	Supprimer le `kubectl` binaire inutile de AMIs.	3 juillet 2019

Version de Kubernetes `1.13`	Ajout de la prise en charge de la version Kubernetes `1.13` pour les nouveaux clusters et les mises à niveau de version.	18 juin 2019
Nouvelle AMI optimisée pour Amazon EKS corrigée pour `AWS-2019-005`	Amazon EKS a mis à jour l'AMI optimisée Amazon EKS afin de corriger les vulnérabilités décrites dans le lien : security/security-bulletins/AWS-2019-005/[AWS-2019-005,type="marketing"].	17 juin 2019
Annonce de l'arrêt de la prise en charge de Kubernetes `1.10` dans Amazon EKS	Amazon EKS a cessé de prendre en charge la version Kubernetes `1.10` le 22 juillet 2019.	21 mai 2019
Mise à jour de la version de plateforme Amazon EKS	Nouvelle version de plateforme pour Kubernetes `1.11` et les clusters `1.10` afin de prendre en charge les noms DNS personnalisés dans le certificat `kubelet` et d'améliorer les performances `etcd`.	21 mai 2019
Démarrer avec `eksctl`	Ce guide de démarrage explique comment installer toutes les ressources nécessaires pour démarrer avec Amazon EKS en utilisant `eksctl`. Il s'agit d'un utilitaire de ligne de commande simple pour créer et gérer les clusters Kubernetes sur Amazon EKS.	10 mai 2019

AWS`get-token` commande CLI	La `aws eks get-token` commande a été ajoutée à la AWS CLI. Il n'est plus nécessaire d'installer l'authentificateur AWS IAM pour Kubernetes pour créer des jetons de sécurité client pour les communications avec le serveur d'API du cluster. Mettez à niveau votre installation AWS CLI vers la dernière version pour utiliser cette nouvelle fonctionnalité. Pour plus d'informations, consultez la section Installation de l'interface de ligne de AWS commande dans le guide de l'utilisateur de l'interface de ligne de AWS commande.	10 mai 2019
Mise à jour de la version de plateforme Amazon EKS	Nouvelle version de plateforme pour les clusters Kubernetes `1.12` afin de prendre en charge les noms DNS personnalisés dans le certificat `kubelet` et d'améliorer les performances `etcd`. Cela corrige un bogue qui faisait que les démons `kubelet` des nœuds redemandaient un nouveau certificat au bout de quelques secondes.	8 mai 2019

Didacticiel Prometheus	Ajout d'une rubrique pour le déploiement de Prometheus dans votre cluster Amazon EKS.	5 avril 2019
Journalisation de plan de contrôle d'Amazon EKS	Avec cette mise à jour, vous pouvez obtenir des journaux d'audit et de diagnostic directement à partir du volet de contrôle d'Amazon EKS. Vous pouvez utiliser ces CloudWatch journaux dans votre compte comme référence pour sécuriser et exécuter des clusters.	4 avril 2019
Version de Kubernetes `1.12`	Ajout de la prise en charge de la version Kubernetes `1.12` pour les nouveaux clusters et les mises à niveau de version.	28 mars 2019
Ajout du guide de démarrage App Mesh	Ajout de la documentation pour démarrer avec App Mesh et Kubernetes.	27 mars 2019
Accès privé au point de terminaison du serveur d'API Amazon EKS	Ajout de la documentation sur la désactivation de l'accès public pour le point de terminaison du serveur API Kubernetes de votre cluster Amazon EKS.	19 mars 2019
Ajout d'une rubrique pour l'installation du serveur de métriques Kubernetes	Le serveur de métriques Kubernetes est un agrégateur de données sur l'utilisation des ressources dans votre cluster.	18 mars 2019

Ajout d'une liste de projets open source connexes	Ces projets open source étendent les fonctionnalités des clusters Kubernetes exécutés sur ces clusters AWS, y compris les clusters gérés par Amazon EKS.	15 mars 2019
Ajout d'une rubrique pour l'installation de Helm en local	Le gestionnaire de package `helm` pour Kubernetes vous permet d'installer et de gérer des applications dans votre cluster Kubernetes. Cette rubrique explique comment installer et exécuter localemen t les fichiers binaires `helm` et `tiller`. Vous pouvez ainsi installer et gérer des graphiques à l'aide de la CLI Helm sur votre système local.	11 mars 2019
Mise à jour de la version de plateforme Amazon EKS	Nouvelle version de plateform e qui met à jour les clusters `1.11` Amazon EKS Kubernete s au niveau du correctif `1.11.8` afin de corriger la vulnérabilité CVE-2019-1002100.	8 mars 2019
Augmentation de la limite de clusters	Amazon EKS a augmenté le nombre de clusters que vous pouvez créer dans une AWS région de 3 à 50.	13 février 2019

Expansion de AWS la région Amazon EKS	Amazon EKS est désormais disponible dans les AWS régions Europe (Londreseu-west-2) (), Europe (Pariseu-west-3) () et Asie-Pacifique (Mumbai) (ap-south-1).	13 février 2019
Nouvelle AMI optimisée pour Amazon EKS corrigée pour ALAS-2019-1156	Amazon EKS a mis à jour l'AMI optimisée pour Amazon EKS afin de résoudre la vulnérabilité décrite dans ALAS-2019-1156.	11 février 2019
Nouvelle AMI optimisée pour Amazon EKS corrigée pour ALAS2-2019-1141	Amazon EKS a mis à jour l'AMI optimisée Amazon EKS pour répondre aux problèmes mentionnés dans le CVEs document ALAS2-2019-1141.	9 janvier 2019
Expansion de AWS la région Amazon EKS	Amazon EKS est désormais disponible dans la AWS région Asie-Pacifique (Séoulap-northeast-2) ().	9 janvier 2019
Extension de région Amazon EKS	Amazon EKS est désormais disponible dans les AWS régions supplémentaires suivantes : Europe (Francfor teu-central-1) (), Asie-Pacifique (Tokyoap-northeast-1) (), Asie-Pacifique (Singapour) ap-southeast-1 () et Asie-Pacifique (Sydney) ap-southeast-2 ().	19 décembre 2018

Mise à jour du cluster Amazon EKS	Ajout de documentation relative à la mise à jour de la version de Kubernetes des clusters et au remplacement de nœuds pour Amazon EKS.	12 décembre 2018
Expansion de AWS la région Amazon EKS	Amazon EKS est désormais disponible dans la AWS région Europe (Stockholm`eu-north-1`) ().	11 décembre 2018
Mise à jour de la version de plateforme Amazon EKS	Nouvelle version de plateform e mettant à jour Kubernetes au niveau du correctif `1.10.11` pour corriger le lien : security/ security-bulletins/AWS-2018 -020/[CVE-2018-1002105,type ="marketing"].	4 décembre 2018
Ajout de la prise en charge de la version `1.0.0` du contrôleu r d'entrée ALB	Le contrôleur d'entrée ALB publie une version `1.0.0` avec le support formel de. AWS	20 novembre 2018
Ajout de la prise en charge de la configuration réseau CNI	Le plug-in CNI Amazon VPC pour Kubernetes version `1.2.1` prend désormais en charge la configuration personnalisée du réseau pour les interfaces réseau des pods secondaires.	16 octobre 2018

Ajout de la prise en charge de `MutatingAdmissionWebhook` et `ValidatingAdmissionWebhook`	La version `1.10-eks.2` de la plateforme Amazon EKS prend désormais en charge les contrôleurs d'admission `MutatingAdmissionWebhook` et `ValidatingAdmissionWebhook`.	10 octobre 2018
Ajout d'informations d'AMI de partenaire	Canonical s'est associé à Amazon EKS pour créer un nœud AMIs que vous pouvez utiliser dans vos clusters.	3 octobre 2018
Instructions ajoutées pour la `update-kubeconfig` commande AWS CLI	Amazon EKS a ajouté le `update-kubeconfig` à la AWS CLI pour simplifier le processus de création d'un `kubeconfig` fichier permettant d'accéder à votre cluster.	21 septembre 2018
Nouvel Amazon EKS optimisé AMIs	Amazon EKS a mis à jour la version optimisée d'Amazon EKS AMIs (avec et sans prise en charge du GPU) afin de fournir divers correctifs de sécurité et optimisations des AMI.	13 septembre 2018
Expansion de AWS la région Amazon EKS	Amazon EKS est désormais disponible dans la région Europe (Irlande) (`eu-west-1`).	5 septembre 2018

Mise à jour de la version de plateforme Amazon EKS	Nouvelle version de plateform e avec support de la couche d'agrégation et Horizonta l Pod Autoscaler (HPA) de Kubernetes.	31 août 2018
Nouvel Amazon EKS optimisé AMIs et prise en charge du GPU	Amazon EKS a mis à jour l'AMI optimisée Amazon EKS afin d'utiliser un nouveau modèle de AWS CloudForm ation nœud et un nouveau script d'amorçage. En outre, une nouvelle AMI optimisée pour Amazon EKS avec prise en charge GPU est disponible.	22 août 2018
Nouvelle AMI optimisée pour Amazon EKS corrigée pour `ALAS2-2018-1058`	Amazon EKS a mis à jour l'AMI optimisée Amazon EKS pour répondre aux problèmes mentionnés dans le CVEs document ALAS2-2018-1058.	14 août 2018
Scripts de génération d'AMI optimisée pour Amazon EKS	Amazon EKS comporte des scripts de génération open source qui sont utilisés pour créer l'AMI optimisée pour Amazon EKS. Ces scripts de compilation sont désormais disponibles sur GitHub.	10 juillet 2018
Version initiale d'Amazon EKS	Documentation initiale pour le lancement de service	5 juin 2018

Contribution au guide de l’utilisateur EKS

AWS a lancé une expérience de contribution améliorée pour le Guide de l’utilisateur d’EKS.

Vous pouvez désormais modifier la source du Guide de l’utilisateur EKS directement sur GitHub.

Les documents utilisent désormais AsciiDoc, un puissant langage de création similaire au markdown. AsciiDoc combine une syntaxe simple avec des fonctionnalités de documentation d’entreprise telles que le formatage avancé, les références croisées et les contrôles de sécurité.

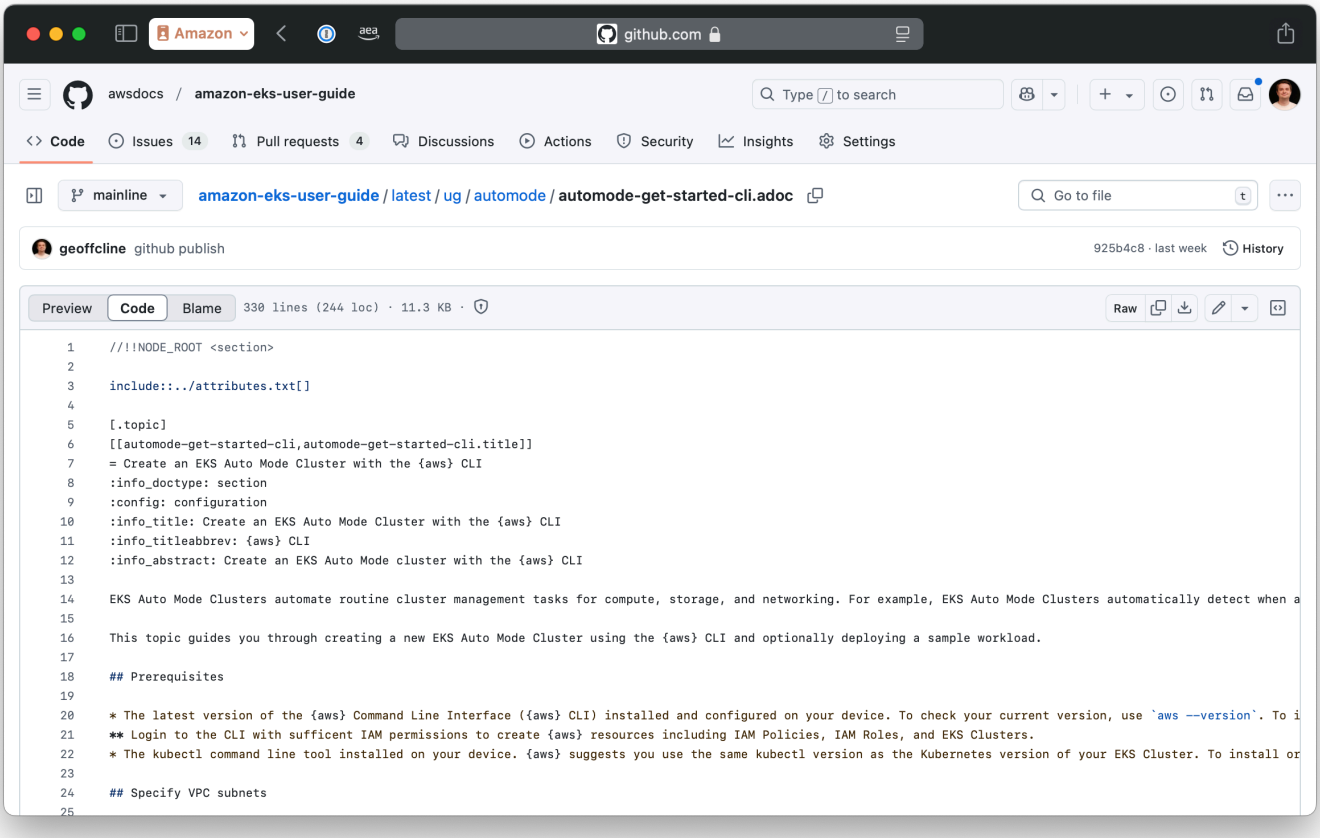
Vous pouvez désormais modifier les documents EKS directement sur GitHub. Notre processus simplifié inclut :

- Traitement plus rapide des pull requests
- Étapes manuelles réduites
- Contrôles automatisés de la qualité du contenu

Nous attendons vos contributions avec impatience.

Modifier une seule page à partir d’un navigateur Web

Vous pouvez facilement modifier une seule page du guide de l’utilisateur EKS directement depuis votre navigateur Web.



Si vous souhaitez modifier plusieurs pages à partir de votre navigateur Web, consultez the section called "Modifiez des fichiers avec GitHub".

Prérequis

- La page Docs à modifier s'ouvre dans le navigateur Web.
- Connecté à GitHub.

Procédure

1. Accédez à la page que vous souhaitez modifier dans le Guide de l'utilisateur Amazon EKS.
2. Dans le volet droit, cliquez sur le lien Modifier cette page sur GitHub.
3. Une fois sur GitHub, ouvrez l'éditeur en :
   - En appuyant sur la touche e de votre clavier.

- Cliquez sur l'icône en forme de crayon et sélectionnez Modifier sur place dans le menu déroulant.
- Si vous n'avez pas la possibilité de le modifier, vous devez vous connecter à GitHub. Votre compte GitHub n'a pas besoin d'autorisations spéciales pour suggérer des modifications. Toutefois, les contributeurs internes d'Amazon doivent associer leur profil GitHub.

4. Apportez les modifications nécessaires au contenu dans l'éditeur GitHub.
   - L'éditeur fournit des fonctionnalités de surlignage syntaxique et de prévisualisation.
   - Vous pouvez utiliser le balisage AsciiDoc pour formater vos modifications.
   - Vous pouvez utiliser `ctrl-f` pour ouvrir une interface de recherche/remplacement.
5. (Facultatif) Prévisualisez vos modifications.
   - Utilisez l'onglet `preview` pour prévisualiser vos modifications grâce à une mise en forme enrichie.
   - Utilisez l'option `show diff` pour mettre en évidence les sections modifiées. Les sections supprimées ont un indicateur rouge dans la marge gauche. Les nouvelles sections ont un indicateur vert dans la marge gauche.
6. Une fois les modifications terminées, cliquez sur le bouton Valider les modifications… en haut de l'éditeur.
7. Dans la boîte de dialogue de validation :
   - Vérifier que votre adresse e-mail est correcte.
   - Ajoutez un message de validation court mais descriptif expliquant vos modifications.
   - Si nécessaire, ajoutez une description plus détaillée.
   - Sélectionnez cette option pour créer une nouvelle branche et une nouvelle pull request.

Vous avez créé une pull request incluant les modifications proposées.

Vue d'ensemble des extraction

Lorsque vous créez un PR :

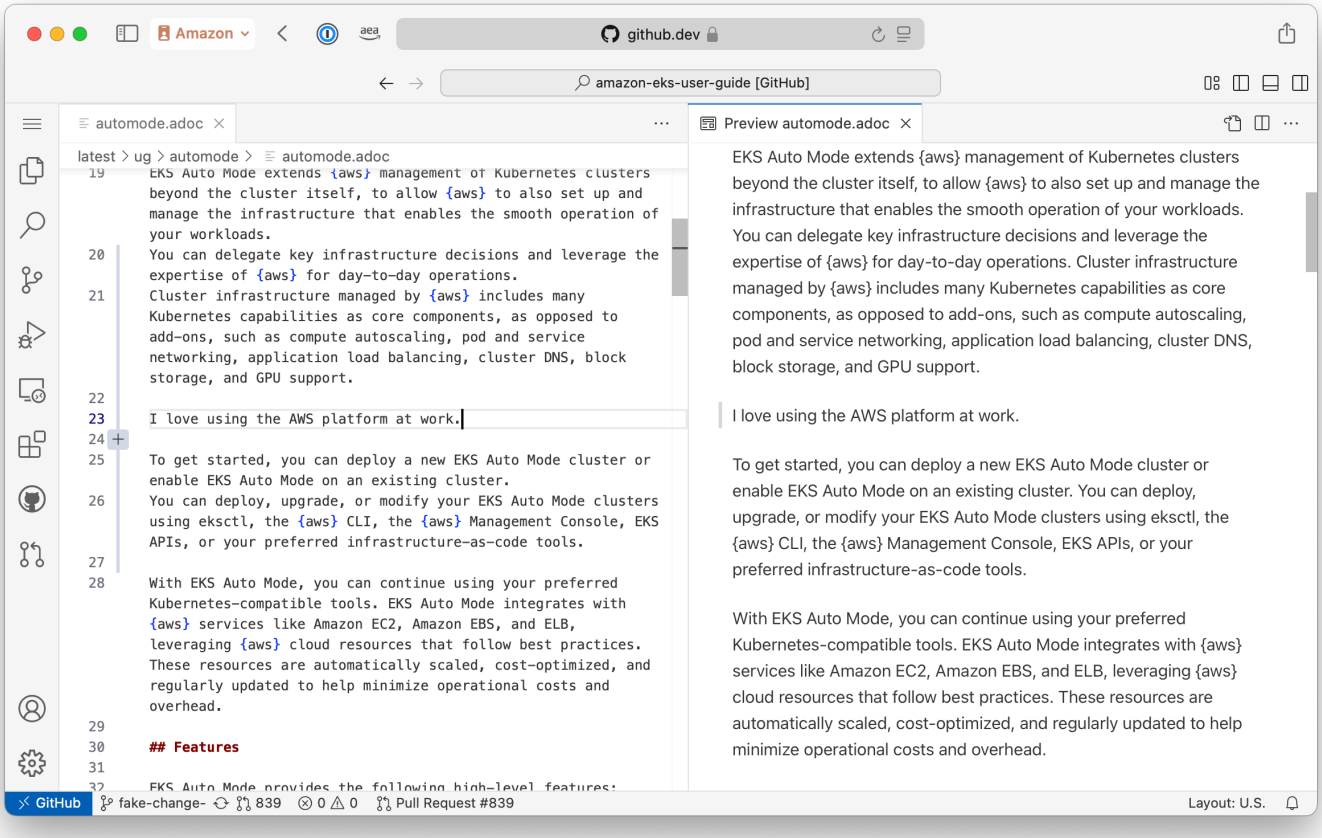
- Vos modifications sont soumises pour examen par les responsables du dépôt.
- Les réviseurs peuvent commenter vos modifications et demander des modifications.
- Des tests automatisés peuvent être exécutés pour valider vos modifications.
- Une fois approuvées, vos modifications peuvent être fusionnées dans le référentiel principal.

Les pull requests contribuent à garantir la qualité et permettent de discuter des changements avant qu'ils ne soient intégrés.

Découvrez comment les pull requests sont examinées et approuvées dans les documents GitHub.

Modifier plusieurs fichiers à partir d'un navigateur Web à l'aide de l'éditeur GitHub Web

Si vous souhaitez proposer la modification de plusieurs pages ou créer une nouvelle page de documentation, utilisez l'éditeur Web GitHub .dev. Cet éditeur Web est basé sur le célèbre éditeur de texte Visual Studio Code.



Conditions préalables

- Connecté à GitHub
- Connaissance de Visual Studio Code Editor
- Connaissance de Git

Procédure

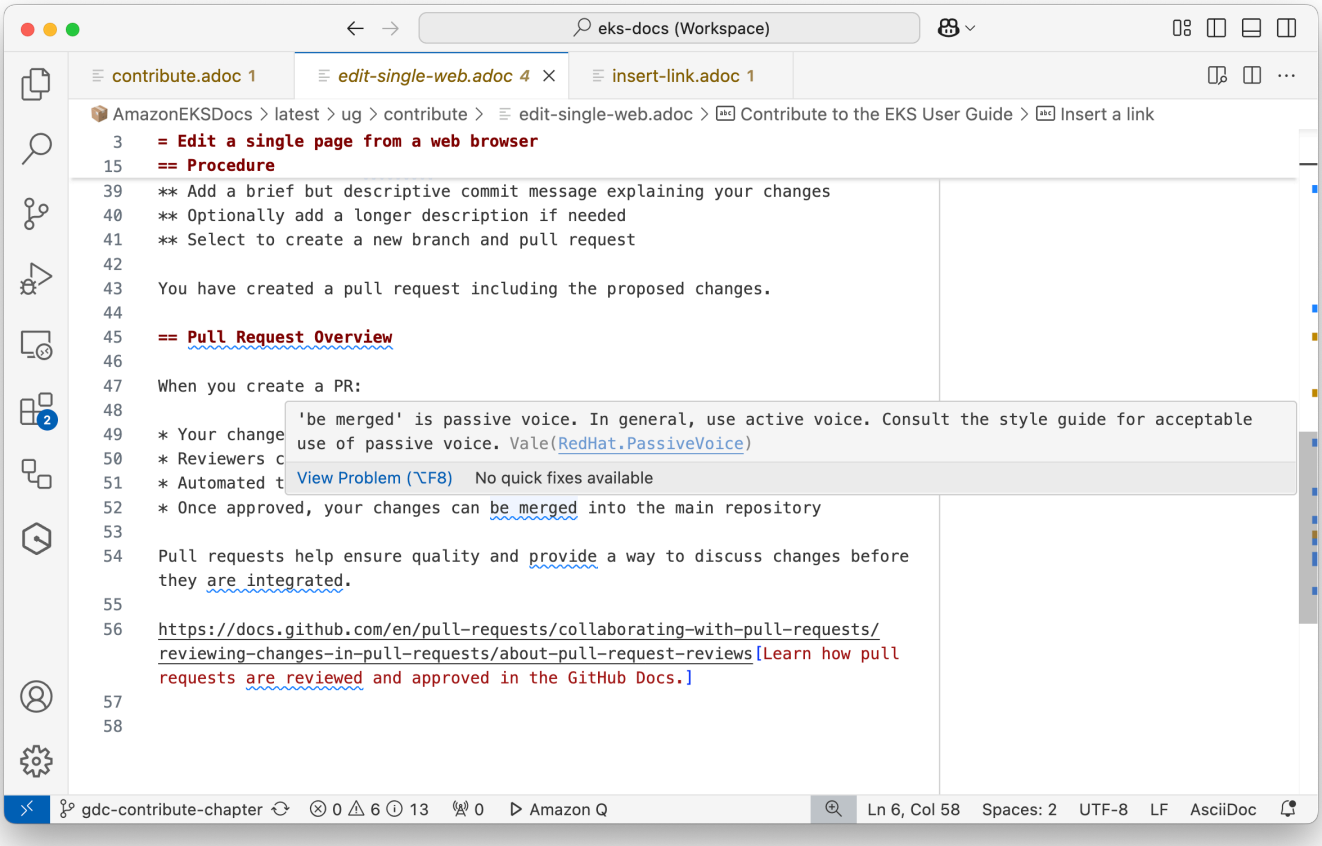
> **Note**
>
> L'équipe EKS Docs a créé un fichier d'espace de travail qui inclut des suggestions de configurations pour l'éditeur, telles que l'habillage du texte et le surlignage AsciiDoc syntaxique. Nous vous suggérons de charger ce fichier d'espace de travail.

1. Ouvrez l'espace de travail sur GitHub .dev.
   - Vous pouvez ajouter l'URL à vos favoris `https://github.dev/awsdocs/amazon-eks-user-guide/blob/mainline/eks-docs.code-workspace?workspace=true`
2. (Première configuration uniquement) Vous serez peut-être invité à créer un fork du dépôt dans votre propre GitHub compte. Acceptez cette invite. Pour plus d'informations, consultez la section À propos des fourchettes dans la GitHub documentation.
3. (Première installation uniquement) Acceptez l'invite en bas à droite pour installer l' AsciiDoc extension.
4. Accédez au contenu de la documentation à l'adresse `latest/ug`.
   - Les fichiers Docs sont organisés par section de niveau supérieur. Par exemple, les pages du chapitre « Sécurité » ont des fichiers source dans le répertoire « security/ ».
5. Pour afficher un aperçu d'une page de documentation, utilisez le bouton Ouvrir l'aperçu sur le côté en haut à droite. L'icône inclut une petite loupe.
6. Utilisez l'onglet Contrôle de source sur la gauche pour valider vos modifications. Pour plus d'informations, consultez les documents Visual Studio Code :
   - Valider les modifications
   - Créer une demande d'extraction

Une fois que vous avez créé une pull request, elle sera examinée par l'équipe de documentation.

Affichez les commentaires sur le style au fur et à mesure que vous tapez en installant Vale localement

Vous pouvez voir les commentaires sur le style au fur et à mesure que vous tapez. Cela permet d'identifier les formulations maladroites et les fautes de frappe.



Présentation :

- La CLI Vale charge les guides de style et les applique aux fichiers source.
- Le référentiel EKS Docs comprend un fichier de configuration Vale qui charge les guides de style et les règles locales.
- L'extension Vale pour Visual Studio (VS) Code affiche les commentaires Vale dans l'éditeur.

Installer Vale

Suivez les instructions de la documentation de la CLI Vale pour installer Vale avec un gestionnaire de packages.

Installer l'extension VS Code Vale

1. Ouvrez VS Code.
2. Cliquez sur l'icône Extensions dans la barre d'activité (ou appuyez sur Ctrl+Maj+X).

3. Recherchez « Vale ».
4. Cliquez sur Installer sur l'extension « Vale VSCode » de Chris Chinchilla.
5. Rechargez VS Code lorsque vous y êtes invité.

Synchroniser Vale

Vale utilise le fichier de configuration `.vale.ini` situé à la racine de votre projet pour déterminer les règles de style à appliquer.

1. Ouvrez VS Code.
2. Cliquez sur Afficher > Terminal (ou appuyez sur Ctrl+`).
3. Accédez au répertoire racine de votre projet si nécessaire.
4. Exécutez la commande :

```
vale sync
```

5. Patientez pendant que Vale termine le téléchargement et la synchronisation des règles de style

Afficher les commentaires sur le style dans VS Code

1. Ouvrez un fichier Markdown ou AsciiDoc dans VS Code.
2. L'extension Vale vérifiera automatiquement votre texte par rapport aux règles de style.
3. Les problèmes de style seront soulignés dans l'éditeur.
4. Passez la souris sur le texte souligné pour voir la suggestion de style spécifique.
5. Corrigez les problèmes en suivant les suggestions ou en consultant le guide de style.

Création d'une nouvelle page

Découvrez comment créer une nouvelle page de documentation. Cette rubrique fournit les instructions pour créer les métadonnées initiales d'une page et l'ajouter à la table des matières du guide.

Création d'une page

1. Accédez au répertoire du chapitre. Par exemple, si vous souhaitez créer une nouvelle page dans la section « Sécurité », accédez au répertoire `latest/ug/security`.
2. Déterminez l'ID de la page. Par convention, l'ID de la page est entièrement en lettres minuscules et séparé par des `-`. L'ID de cette page est `create-page`.
3. Créez un nouveau fichier avec l'ID de la page et l'extension `adoc`. Par exemple, `create-page.adoc`.
4. Insérez les métadonnées de la page à l'aide du modèle suivant :

```
include::../attributes.txt[]

[.topic]
[#unique-page-id]
= Page title
:info_titleabbrev: Short title

[abstract]
--
Brief summary of the page's content.
--

Introduction paragraph goes here.
```

Ajout de la page à la navigation

1. Accédez à la page parente. La page parente des sections de premier niveau est `book.adoc`.
2. Ajoutez la page enfant au bas de la page parente.

   ```
   include::${filename}[leveloffset=+1]
   ```

   Par exemple :

   ```
   include::create-page.adoc[leveloffset=+1]
   ```

Insérer un lien

AsciiDoc prend en charge plusieurs types de liens. Il est important d'utiliser le bon type de lien pour que le lien fonctionne correctement dans différents environnements.

Lien vers une page ou une section du Guide de l'utilisateur EKS

Utilisez les références croisées (xref) pour créer des liens entre les pages ou les sections d'un même site de documentation, tel que le Guide de l'utilisateur EKS. Ils sont automatiquement mis à jour si la section cible est déplacée ou renommée.

Utiliser le titre de page comme texte de lien

Dans la plupart des cas, lorsque vous créez un lien vers un autre identifiant dans ce guide de l'utilisateur, utilisez l'approche suivante pour que le texte du lien soit automatiquement mis à jour avec le dernier titre, si nécessaire.

```
For more information, see <<page-or-section-id>>.
```

Définition de texte de lien personnalisé

Dans les cas où vous devez avoir un texte de lien personnalisé, utilisez le format suivant.

```
Here's an example of a <<page-or-section-id,link with custom text>>.
```

Lien vers un autre guide dans les AWS documents

1. Trouvez le lien vers la page de AWS documentation.
2. Supprimez le préfixe `https://docs.aws.amazon.com/` en ne conservant que le chemin. Le chemin doit commencer par une lettre.
3. Créez un lien comme indiqué ci-dessous :

```
link:AmazonS3/latest/userguide/create-bucket-overview.html[Create a bucket,
 type="documentation"]
```

Lien vers une page Web externe

Ce format crée un lien standard vers une page qui n'est pas hébergée par Amazon. Par exemple, utilisez-le pour les GitHub liens.

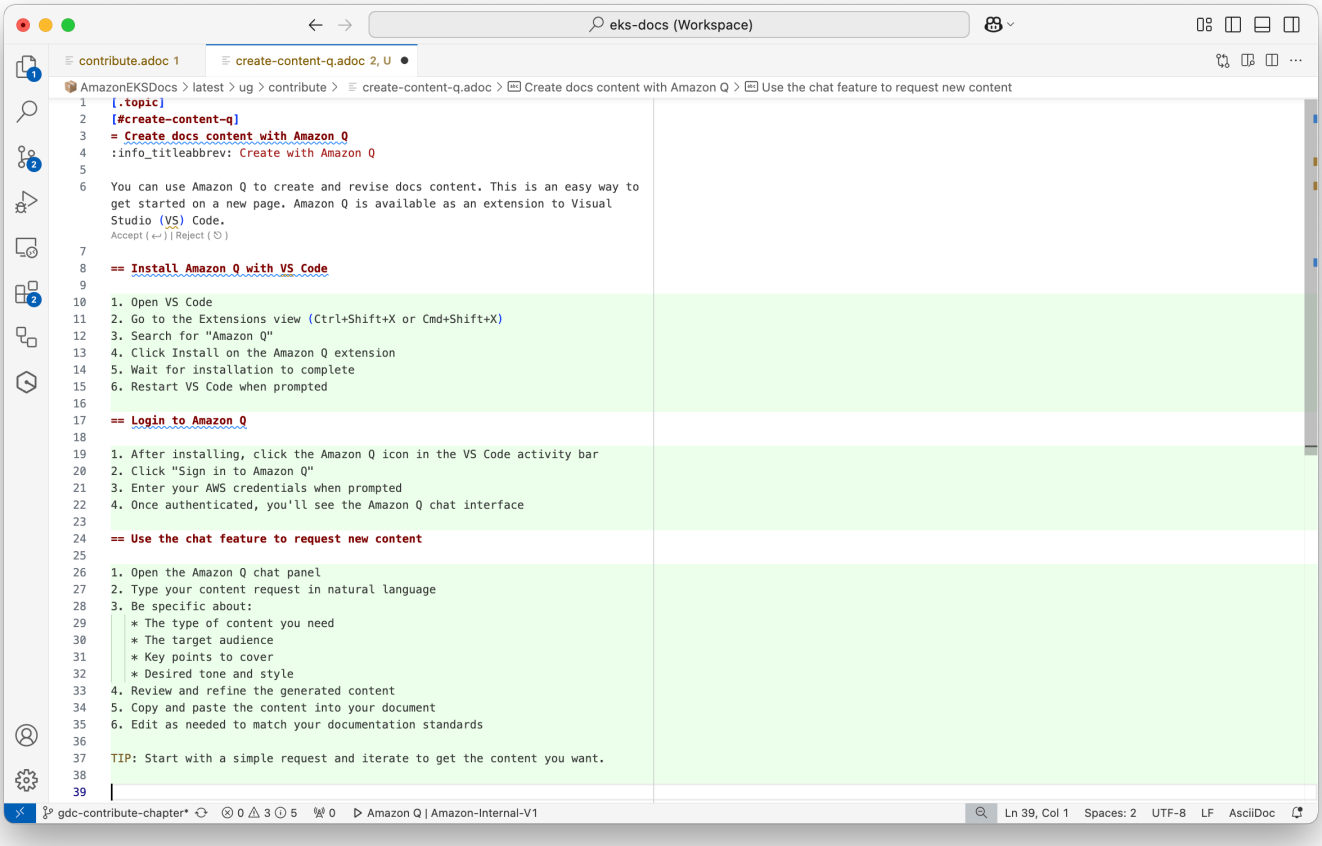
```
https://example.com[Link text]
```

> **Note**
>
> Nous avons une liste d'autorisations pour les domaines externes. La liste des autorisations se trouve à `vale/styles/EksDocs/ExternalDomains.yml`

Création de contenu de documents avec Amazon Q

Vous pouvez utiliser Amazon Q pour créer et réviser le contenu des documents. C'est un moyen simple de démarrer une nouvelle page. Amazon Q est disponible sous forme d'extension pour Visual Studio (VS) Code.

Dans l'image suivante, Amazon Q a généré les lignes marquées en vert.



Installation d'Amazon Q avec VS Code

1. Ouvrir VS Code
2. Accédez à la vue Extensions (Ctrl+Shift+X ou Cmd+Shift+X)
3. Recherchez « Amazon Q »
4. Cliquez sur Installer dans l'extension Amazon Q
5. Attendez la fin de l'installation du logiciel
6. Redémarrez VS Code lorsque vous y êtes invité

Connexion à Amazon Q

1. Après installation, choisir l'icône Amazon Q dans la barre d'activité de VS Code.
2. Sélectionnez Se connecter à Amazon Q.
3. Saisissez vos informations d'identification AWS lorsque vous y êtes invité.

4. Une fois l'authentification réussie, l'interface de conversation Amazon Q apparaît.

Utilisation d'Amazon Q pour créer du contenu

1. Ouvrez le fichier que vous souhaitez modifier dans VS Code.
2. Sélectionnez le texte que vous souhaitez réviser ou l'emplacement du nouveau contenu.
3. Appuyez sur Ctrl+I ou Cmd+I.
4. Dans l'invite, précisez clairement :
   - Le type de contenu dont vous avez besoin.
   - Le public cible.
   - Principaux points à aborder.
   - Ton et style souhaités.
5. Passez en revue le contenu généré dans l'aperçu intégré.
6. Appuyez sur Entrée pour accepter les modifications ou sur Échap pour les refuser.
7. Modifiez davantage si nécessaire.

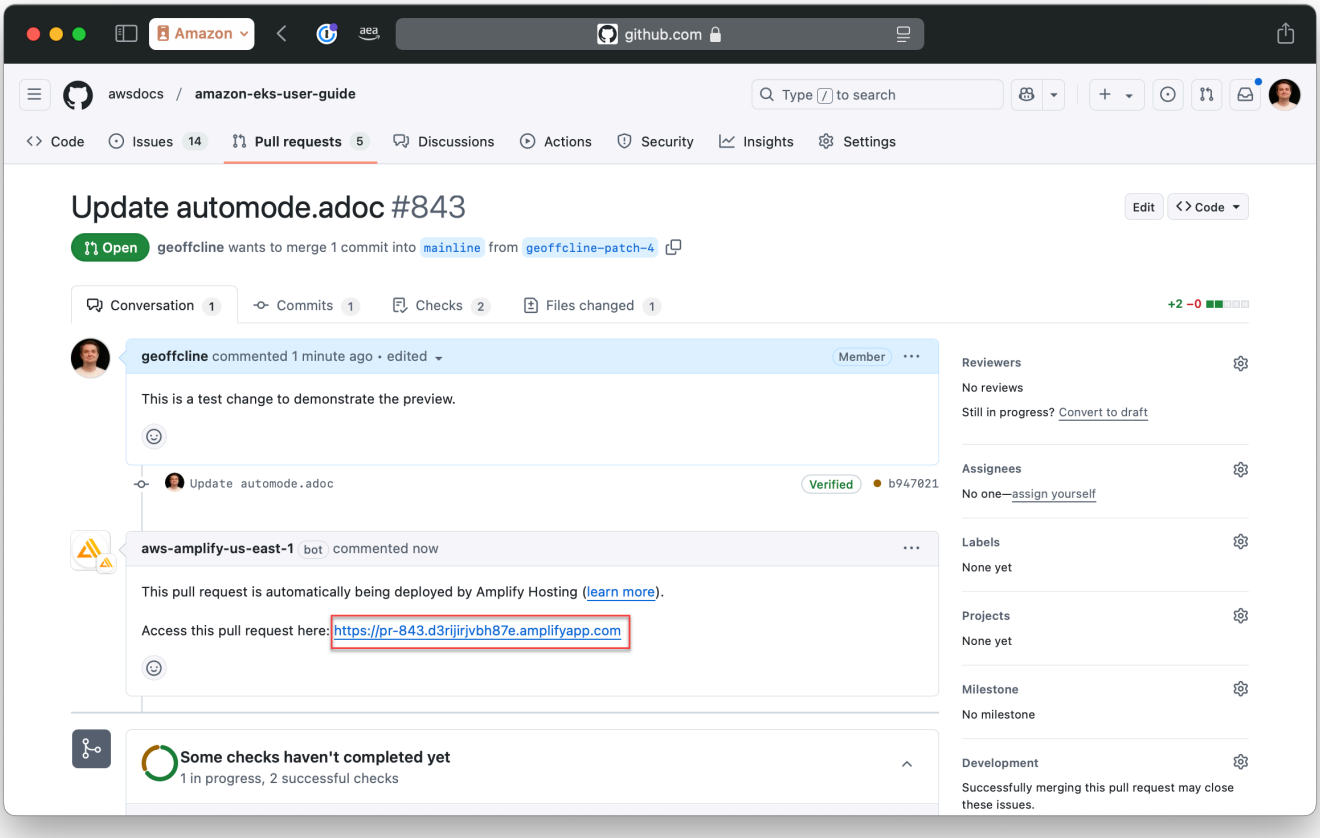
Conseils

- Commencez par une demande simple et itérez pour obtenir le contenu souhaité.
- Créez un premier brouillon des titres de page, puis demander à Q de les développer.
- Amazon Q peut générer du Markdown. Cela ne pose aucun problème. L'outil AsciiDoc comprend la plupart de la syntaxe Markdown.

Pour en savoir plus sur Amazon Q Developer, consultez Utilisation d'Amazon Q Developer dans l'IDE.

Afficher un aperçu du contenu de la demande d'extraction

Le guide de l'utilisateur Amazon EKS GitHub est configuré pour créer et générer un aperçu du site de documentation. Cet aperçu ne présente pas le thème AWS complet, mais il permet de vérifier que le contenu est correctement créé et que les liens fonctionnent.



Cet aperçu est hébergé à une URL temporaire par AWS Amplify.

Afficher l'aperçu

Lorsque vous soumettez une pull request, AWS Amplify tente de créer et de déployer un aperçu du contenu.

Si la création réussit, aws-amplify-us-east-1 ajoute un commentaire à la pull request avec un lien vers l'aperçu. Sélectionnez le lien à droite de « Accédez à cette pull request ici » (comme indiqué dans la capture d'écran avec un contour rouge).

Si la construction échoue, les administrateurs du dépôt peuvent consulter les journaux et fournir des commentaires.

> **Note**
>
> Si vous n'avez jamais contribué auparavant, un responsable du projet devra peut-être approuver l'exécution de la construction.

Limitations propres à la version préliminaire

L'aperçu est créé sous la forme d'un seul fichier HTML volumineux. Il s'affichera sous forme de plusieurs pages lors de sa publication.

Ce qui fonctionne :

- Références croisées (`xref`)
- Liens vers Internet
- Images
- Contenu hébergé à partir de `samples/`

Ce qui ne fonctionne pas :

- Liens vers d'autres contenus AWS, utilisant `type="documentation"`. En effet, ce contenu n'existe pas dans l'environnement de prévisualisation.
- L'attribut `{aws}` ne s'affichera pas correctement. Sa valeur varie en fonction de l'environnement.

Référence de la syntaxe AsciiDoc

AsciiDoc est le langage de balisage privilégié pour la documentation AWS. Même si l'outil prend partiellement en charge la syntaxe Markdown (par exemple, les titres et les listes), nous recommandons d'utiliser la syntaxe AsciiDoc pour l'ensemble du contenu afin d'assurer la cohérence et un rendu correct.

Ce guide couvre les éléments syntaxiques les plus courants dont vous aurez besoin pour contribuer à la documentation Amazon EKS. Pour des informations plus avancées et détaillées sur la syntaxe, consultez la documentation AsciiDoc.

Nouvelle page

Chaque nouveau document AsciiDoc doit commencer avec la structure définie dans the section called "Création d'une page".

En-têtes

Utilisez des en-têtes pour organiser votre contenu de manière hiérarchique :

```
= Page/topic title (level 1)
== Section title (level 2)
=== Level 3 heading
==== Level 4 heading
===== Level 5 heading
====== Level 6 heading
```

Remarque : utilisez toujours la casse phrase pour les en-têtes de la documentation AWS.

Formatage du texte

Formatez le texte pour souligner les informations importantes :

```
Use *bold text* for UI labels.
Use _italic text_ for introducing terms or light emphasis.
Use `monospace text` for code, file names, and commands.
```

Listes

Listes non ordonnées

Créez des listes à puces pour des éléments sans ordre particulier :

```
* First item
* Second item
** Nested item
** Another nested item
* Third item
```

Listes ordonnées

Créez des listes numérotées pour des étapes séquentielles ou des éléments par ordre de priorité :

```
. First step
. Second step
.. Substep 1
.. Substep 2
. Third step
```

Liens

Pour plus d'informations sur la manière de formater correctement les liens, consultez the section called "Insérer le lien". Les liens de type Markdown ne sont pas pris en charge.

Exemples de code

Code incorporé

Utilisez des accents graves pour le code incorporé :

```
Use the `kubectl get pods` command to list all pods.
```

Blocs de code

Créez des blocs de code avec coloration de syntaxe et prise en charge d'attributs (similaires aux entités AsciiDoc) :

```
[source,python,subs="verbatim,attributes"]
----
def hello_world():
    print("Hello, World!")
----
```

Images

Insérez des images avec du texte alternatif pour l'accessibilité :

```
image::images/image-file.png[Alt text description]
```

Tables

Créez des tables pour organiser les informations :

```
[%header,cols="2"]
|===
|Header 1
|Header 2

|Cell 1,1
|Cell 1,2

|Cell 2,1
|Cell 2,2
|===
```

Pour des tables plus complexes, consultez la documentation AsciiDoc sur les tables.

Alertes

Utilisez des alertes pour mettre en évidence les informations et les avertissements importants :

```
NOTE: This is a note callout for general information.

TIP: This is a tip callout for helpful advice.

IMPORTANT: This is an important callout for critical information.
```

Aperçu :

> **Note**
>
> Ceci est une alerte de type note.

Inclusion d'autres fichiers

Inclure du contenu provenant d'autres fichiers :

```
include::filename.adoc[]
```

Attributs (similaires aux entités)

Utilisez des attributs prédéfinis pour assurer la cohérence. En particulier, vous DEVEZ utiliser des attributs pour AWS et `arn:aws:` .

```
{aws} provides Amazon EKS as a managed Kubernetes service.
```

```
 [source,bash,subs="verbatim,attributes"]
 ----
 aws iam attach-role-policy \
     --role-name AmazonEKSAutoClusterRole \
     --policy-arn {arn-aws}iam::aws:policy/AmazonEKSClusterPolicy
 ----
```

Pour obtenir la liste des attributs, consultez le fichier `../attributes.txt`.

Procédures

Formater les procédures étape par étape :

```
To create an Amaozon EKS cluster. do the following steps.

. Sign in to the {aws} Management Console.
. Open the Amazon EKS console.
. Choose *Create cluster*.

...
```