aws

Guide de développement d'Amazon EMR on EKS

Amazon EMR

Amazon EMR: Guide de développement d'Amazon EMR on EKS

Table of Contents

Qu'est-ce qu'Amazon EMR on EKS ?

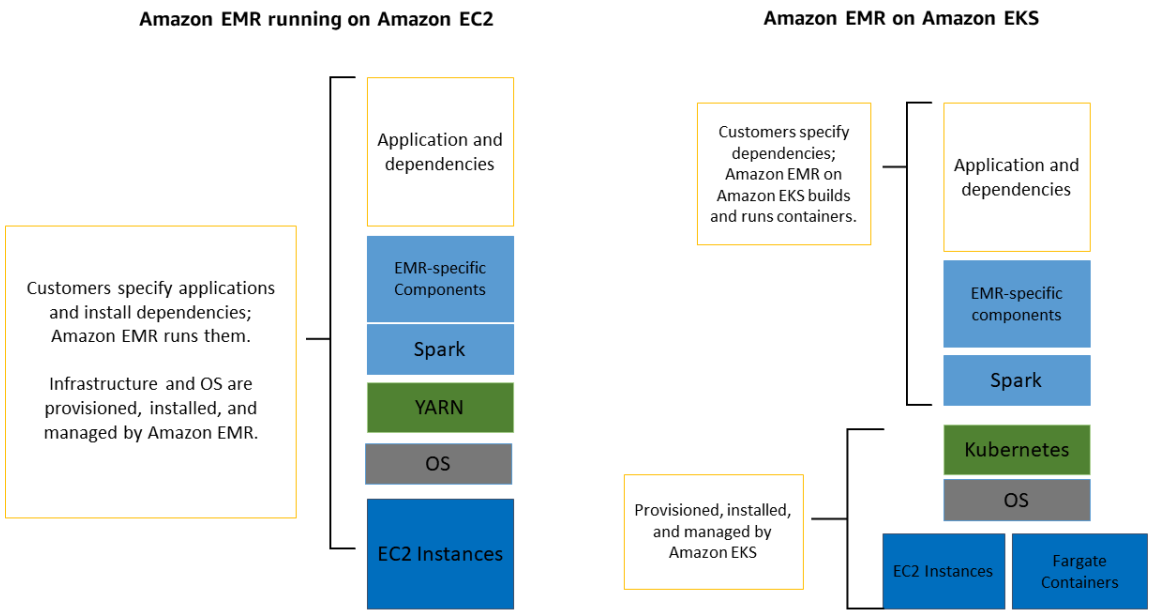
Amazon EMR on EKS offre une option de déploiement pour Amazon EMR qui vous permet d'exécuter des environnements de big data open-source sur Amazon Elastic Kubernetes Service (Amazon EKS). Grâce à cette option de déploiement, vous pouvez vous concentrer sur l'exécution des charges de travail analytiques pendant qu'Amazon EMR on EKS crée, configure et gère les conteneurs pour les applications open-source.

Si vous utilisez déjà Amazon EMR, vous pouvez désormais exécuter des applications basées sur Amazon EMR avec d'autres types d'applications sur le même cluster Amazon EKS. Cette option de déploiement améliore également l'utilisation des ressources et simplifie la gestion de l'infrastructure dans plusieurs zones de disponibilité. Si vous exécutez déjà des environnements de big data sur Amazon EKS, vous pouvez désormais utiliser Amazon EMR pour automatiser le provisionnement et la gestion, et exécuter Apache Spark plus rapidement.

Amazon EMR on EKS permet à votre équipe de collaborer plus efficacement et de traiter de grandes quantités de données plus facilement et à moindre coût :

- Vous pouvez exécuter des applications sur un groupe commun de ressources sans avoir à provisionner l'infrastructure. Vous pouvez utiliser Amazon EMR Studio et le AWS SDK ou AWS CLI pour développer, soumettre et diagnostiquer des applications d'analyse exécutées sur des clusters EKS. Vous pouvez exécuter des tâches planifiées sur Amazon EMR on EKS en utilisant Apache Airflow ou Amazon Managed Workflows for Apache Airflow (MWAA).
- Les équipes d'infrastructure peuvent gérer de manière centralisée une plateforme informatique commune pour consolider les charges de travail Amazon EMR avec d'autres applications basées sur des conteneurs. Vous pouvez simplifier la gestion de l'infrastructure avec les outils courants d'Amazon EKS et tirer parti d'un cluster partagé pour les charges de travail qui nécessitent différentes versions d'environnements open-source. Vous pouvez également réduire les frais généraux opérationnels grâce à l'automatisation de la gestion des clusters Kubernetes et l'application des correctifs du système d'exploitation. Avec Amazon EC2 et AWS Fargate, vous pouvez activer plusieurs ressources informatiques pour répondre aux exigences de performance, opérationnelles ou financières.

Le diagramme suivant illustre les deux modèles de déploiement d'Amazon EMR.



Rubriques

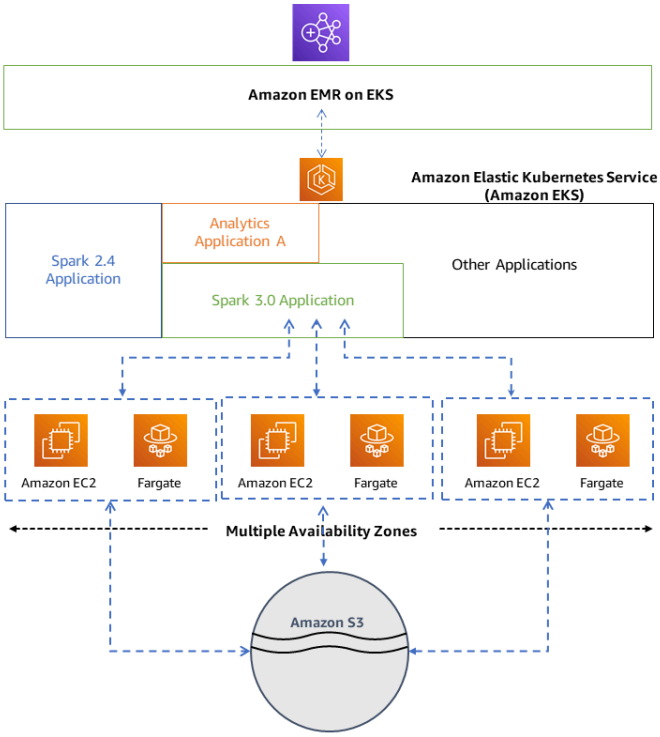
- Architecture pour Amazon EMR sur EKS
- Comprendre les concepts et la terminologie d'Amazon EMR on EKS
- Que se passe-t-il lorsque vous soumettez un travail à un cluster virtuel Amazon EMR sur EKS

Architecture pour Amazon EMR sur EKS

Amazon EMR on EKS associe de manière souple les applications à l'infrastructure sur laquelle elles s'exécutent. Chaque couche d'infrastructure assure l'orchestration de la couche suivante. Lorsque vous soumettez une tâche à Amazon EMR, la définition de votre tâche contient tous les paramètres spécifiques à l'application. Amazon EMR utilise ces paramètres pour indiquer à Amazon EKS quels pods et conteneurs déployer. Amazon EKS met ensuite en ligne les ressources informatiques d'Amazon EC2 AWS Fargate nécessaires à l'exécution de la tâche.

Grâce à ce couplage faible des services, vous pouvez exécuter simultanément plusieurs tâches isolées en toute sécurité. Vous pouvez également comparer la même tâche à différents backends de calcul ou répartir votre tâche sur plusieurs zones de disponibilité pour améliorer la disponibilité.

Le schéma suivant illustre le fonctionnement d'Amazon EMR sur EKS avec d'autres AWS services.



Comprendre les concepts et la terminologie d'Amazon EMR on EKS

Amazon EMR on EKS offre une option de déploiement pour Amazon EMR qui vous permet d'exécuter des environnements de big data open-source sur Amazon Elastic Kubernetes Service (Amazon EKS). Cette rubrique fournit des informations contextuelles sur certains termes courants, notamment les espaces de noms, les clusters virtuels et les exécutions de tâches, qui sont des unités de travail que vous soumettez pour traitement.

Espace de noms Kubernetes

Amazon EKS utilise les espaces de noms Kubernetes pour répartir les ressources du cluster entre plusieurs utilisateurs et applications. Ces espaces de noms constituent la base des environnements multilocataire. Un espace de noms Kubernetes peut avoir Amazon EC2 ou AWS Fargate être le fournisseur de calcul. Cette flexibilité vous offre différentes options en termes de performances et de coûts pour l'exécution de vos tâches.

Cluster virtuel

Le cluster virtuel est un espace de noms Kubernetes que vous enregistrez sur Amazon EMR. Amazon EMR utilise des clusters virtuels pour exécuter des tâches et héberger des points de terminaison. Plusieurs clusters virtuels peuvent être soutenus par le même cluster physique. Toutefois, chaque cluster virtuel correspond à un espace de noms sur un cluster EKS. Les clusters virtuels ne créent aucune ressource active qui contribue à votre facture ou qui nécessite une gestion du cycle de vie en dehors du service.

Exécution de tâche

Une exécution de tâche est une unité de travail, telle qu'un fichier JAR, un PySpark script ou une requête SparkSQL Spark, que vous soumettez à Amazon EMR sur EKS. Une même tâche peut faire l'objet de plusieurs exécutions. Lorsque vous soumettez une exécution de tâche, vous incluez les informations suivantes :

- Un cluster virtuel dans lequel la tâche doit être exécutée.
- Un nom de travail pour identifier la tâche.
- Le rôle d'exécution : un rôle IAM délimité qui exécute la tâche et vous permet d'indiquer les ressources auxquelles la tâche peut accéder.
- L'étiquette de version Amazon EMR qui indique la version des applications open-source à utiliser.
- Les artefacts à utiliser lors de la soumission de votre tâche, tels que les paramètres spark-submit.

Par défaut, les journaux sont chargés sur le serveur d'historique Spark et sont accessibles à partir de la AWS Management Console. Vous pouvez également envoyer des journaux d'événements, des journaux d'exécution et des métriques vers Amazon S3 et Amazon CloudWatch.

Conteneurs Amazon EMR

Les conteneurs Amazon EMR sont le nom de l'API pour Amazon EMR on EKS. Le préfixe `emr-containers` est utilisé dans les scénarios suivants :

- C'est le préfixe des commandes CLI pour Amazon EMR on EKS. Par exemple, `aws emr-containers start-job-run`.
- C'est le préfixe précédant les actions de la politique IAM pour Amazon EMR on EKS. Par exemple, `"Action": [ "emr-containers:StartJobRun"]`. Pour plus d'informations, consultez la rubrique Actions de la politique pour Amazon EMR.

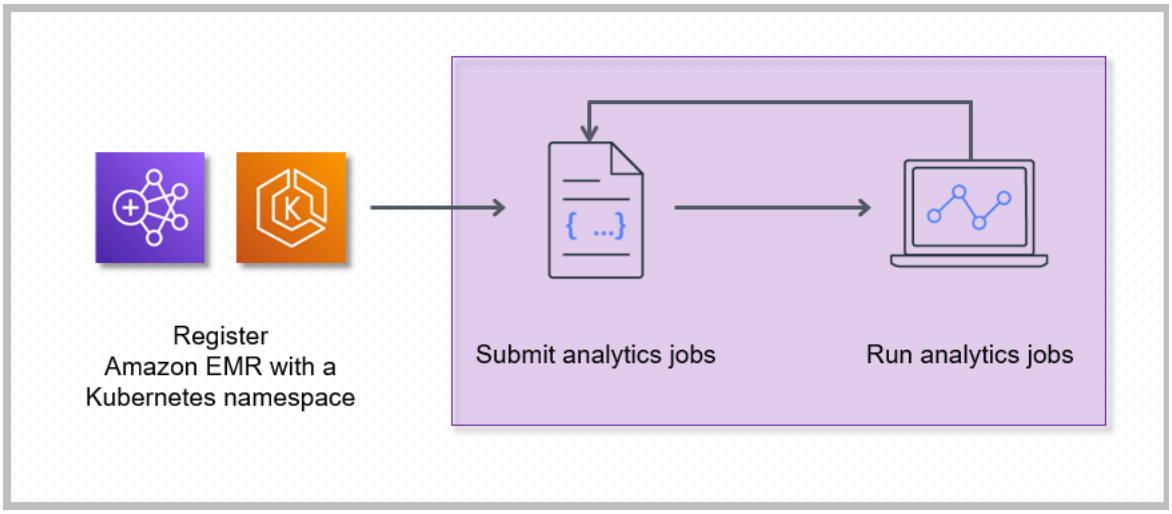
- C'est le préfixe utilisé pour les points de terminaison de service Amazon EMR on EKS. Par exemple, `emr-containers.us-east-1.amazonaws.com`. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Que se passe-t-il lorsque vous soumettez un travail à un cluster virtuel Amazon EMR sur EKS

En enregistrant Amazon EMR dans un espace de noms Kubernetes sur Amazon EKS, vous créez un cluster virtuel. Amazon EMR peut alors exécuter des charges de travail analytiques sur cet espace de noms. Lorsque vous utilisez Amazon EMR on EKS pour soumettre des tâches Spark au cluster virtuel, Amazon EMR on EKS demande au planificateur Kubernetes sur Amazon EKS de planifier des pods.

Les étapes et le schéma diagramme illustrent le flux de travail Amazon EMR on EKS :

- Utilisez un cluster Amazon EKS existant ou créez-en un à l'aide de l'utilitaire de ligne de commande eksctl ou de la console Amazon EKS.
- Créez un cluster virtuel en enregistrant Amazon EMR avec un espace de noms sur un cluster EKS.
- Soumettez votre tâche au cluster virtuel à l'aide du SDK AWS CLI ou.



Pour chaque tâche que vous exécutez, Amazon EMR on EKS crée un conteneur avec une image de base Amazon Linux 2, Apache Spark, et les dépendances associées. Chaque tâche s'exécute dans un pod qui télécharge le conteneur et commence à l'exécuter. Le pod s'arrête une fois la tâche terminée. Si l'image du conteneur a déjà été déployée sur le nœud, une image en cache est

utilisée et le téléchargement est évité. Des conteneurs sidecar, tels que ceux pour la redirection de journaux ou de métriques, peuvent être déployés dans le pod. Une fois la tâche terminée, vous pouvez toujours la déboguer à l'aide de l'interface utilisateur de l'application Spark dans la console Amazon EMR.

Commencer à utiliser Amazon EMR sur EKS

Cette rubrique vous aide à commencer à utiliser Amazon EMR on EKS en déployant une application Spark sur un cluster virtuel. Il inclut les étapes permettant de configurer les autorisations appropriées et de démarrer une tâche. Avant de commencer, assurez-vous d'avoir terminé les étapes de Configuration d'Amazon EMR on EKS. Cela vous permet d'obtenir des outils tels que la AWS CLI configuration avant de créer votre cluster virtuel. Pour d'autres modèles qui peuvent vous aider à démarrer, consultez notre guide des meilleures pratiques en matière de conteneurs EMR sur. GitHub

Lors des étapes de configuration, vous aurez besoin des informations suivantes :

- L'identifiant du cluster virtuel pour le cluster Amazon EKS et l'espace de noms Kubernetes enregistrés dans Amazon EMR

  > **Important**
  >
  > Lors de la création d'un cluster EKS, veillez à utiliser m5.xlarge comme type d'instance, ou tout autre type d'instance disposant d'une capacité plus élevée en matière de CPU et de mémoire. L'utilisation d'un type d'instance dont la CPU ou la mémoire sont inférieurs à celles de m5.xlarge peut entraîner l'échec de la tâche en raison de l'insuffisance des ressources disponibles dans le cluster.

- Nom du rôle IAM utilisé pour l'exécution de la tâche
- Étiquette de version Amazon EMR (par exemple, `emr-6.4.0-latest`)
- Cibles de destination pour la journalisation et la surveillance :
  - Nom du groupe de CloudWatch journaux Amazon et préfixe du flux de journaux
  - Emplacement Amazon S3 pour stocker les journaux des événements et des conteneurs

> **Important**
>
> Les tâches Amazon EMR on EKS utilisent Amazon CloudWatch et Amazon S3 comme cibles de destination pour la surveillance et la journalisation. Vous pouvez suivre l'avancement des tâches et résoudre les échecs en consultant les journaux des tâches envoyés à ces destinations. Pour activer la journalisation, la politique IAM associée au rôle IAM pour l'exécution des tâches doit disposer des autorisations requises pour accéder aux ressources cibles. Si la politique IAM ne dispose pas des autorisations requises, vous devez suivre les

étapes décrites dans Mise à jour la politique d'approbation du rôle d'exécution des tâches Configurer l'exécution d'une tâche pour utiliser les journaux Amazon S3 et Configurer une tâche exécutée pour utiliser les CloudWatch journaux avant d'exécuter cet exemple de tâche.

Exécution d'une application Spark

Pour exécuter une application Spark simple sur Amazon EMR on EKS, procédez comme suit. Le fichier d'application `entryPoint` d'une application Spark Python se trouve à l'adresse `s3://`*`REGION`*`.elasticmapreduce/emr-containers/samples/wordcount/scripts/wordcount.py`. *`REGION`*Il s'agit de la région dans laquelle réside votre cluster virtuel Amazon EMR on EKS, par exemple. *`us-east-1`*

1. Mettez à jour la politique IAM pour le rôle d'exécution des tâches avec les autorisations requises, comme le montrent les déclarations de politique ci-dessous.

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "ReadFromLoggingAndInputScriptBuckets",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::*.elasticmapreduce",
        "arn:aws:s3:::*.elasticmapreduce/*",
        "arn:aws:s3:::amzn-s3-demo-bucket",
        "arn:aws:s3:::amzn-s3-demo-bucket/*",
        "arn:aws:s3:::amzn-s3-demo-bucket-b",
        "arn:aws:s3:::amzn-s3-demo-bucket-b/*"
      ]
    },
    {
      "Sid": "WriteToLoggingAndOutputDataBuckets",
      "Effect": "Allow",
```

```
      "Action": [
        "s3:PutObject",
        "s3:DeleteObject"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket/*",
        "arn:aws:s3:::amzn-s3-demo-bucket-b/*"
      ]
    },
    {
      "Sid": "DescribeAndCreateCloudwatchLogStream",
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogStream",
        "logs:DescribeLogGroups",
        "logs:DescribeLogStreams"
      ],
      "Resource": [
        "arn:aws:logs:*:*:*"
      ]
    },
    {
      "Sid": "WriteToCloudwatchLogs",
      "Effect": "Allow",
      "Action": [
        "logs:PutLogEvents"
      ],
      "Resource": [
        "arn:aws:logs:*:*:log-group:my_log_group_name:log-
stream:my_log_stream_prefix/*"
      ]
    }
  ]
}
```

- La première déclaration `ReadFromLoggingAndInputScriptBuckets` de cette politique accorde à `ListBucket` et `GetObjects` l'accès aux compartiments Amazon S3 suivants :
  - *`REGION`*`.elasticmapreduce` : le compartiment dans lequel se trouve le fichier d'application `entryPoint`.
  - *`amzn-s3-demo-destination-bucket`*- un compartiment que vous définissez pour vos données de sortie.

- *amzn-s3-demo-logging-bucket*- un compartiment que vous définissez pour vos données de journalisation.

- La deuxième déclaration `WriteToLoggingAndOutputDataBuckets` de cette politique accorde à la tâche l'autorisation d'écrire des données dans vos compartiments de sortie et de journalisation, respectivement.
- La troisième déclaration `DescribeAndCreateCloudwatchLogStream` accorde à la tâche l'autorisation de décrire et de créer Amazon CloudWatch Logs.
- La quatrième instruction `WriteToCloudwatchLogs` accorde l'autorisation d'écrire des journaux dans un groupe de CloudWatch journaux Amazon nommé *my_log_group_name* sous un flux de journaux nommé*my_log_stream_prefix*.

2. Pour exécuter une application Spark Python, utilisez la commande ci-dessous. Remplacez toutes les *red italicized* valeurs remplaçables par les valeurs appropriées. *REGION*Il s'agit de la région dans laquelle réside votre cluster virtuel Amazon EMR on EKS, par exemple. *us-east-1*

```
aws emr-containers start-job-run \
--virtual-cluster-id cluster_id \
--name sample-job-name \
--execution-role-arn execution-role-arn \
--release-label emr-6.4.0-latest \
--job-driver '{
  "sparkSubmitJobDriver": {
    "entryPoint": "s3://REGION.elasticmapreduce/emr-containers/samples/wordcount/
scripts/wordcount.py",
    "entryPointArguments": ["s3://amzn-s3-demo-destination-bucket/
wordcount_output"],
    "sparkSubmitParameters": "--conf spark.executor.instances=2 --
conf spark.executor.memory=2G --conf spark.executor.cores=2 --conf
 spark.driver.cores=1"
  }
}' \
--configuration-overrides '{
  "monitoringConfiguration": {
    "cloudWatchMonitoringConfiguration": {
      "logGroupName": "my_log_group_name",
      "logStreamNamePrefix": "my_log_stream_prefix"
    },
    "s3MonitoringConfiguration": {
       "logUri": "s3://amzn-s3-demo-logging-bucket"
    }
  }
```

```
}'
```

Les données de sortie de cette tâche seront disponibles à l'adresse `s3://amzn-s3-demo-destination-bucket/wordcount_output`.

Vous pouvez également créer un fichier JSON avec des paramètres spécifiques pour l'exécution de votre tâche. Exécutez ensuite la commande `start-job-run` avec un chemin d'accès au fichier JSON. Pour de plus amples informations, veuillez consulter Soumission d'une tâche exécutée avec `StartJobRun`. Pour plus d'informations sur la configuration des paramètres d'exécution des tâches, consultez Options de configuration d'une exécution de tâche.

3. Pour exécuter une application Spark SQL, utilisez la commande ci-dessous. Remplacez toutes les *red italicized* valeurs par les valeurs appropriées. *REGION*Il s'agit de la région dans laquelle réside votre cluster virtuel Amazon EMR on EKS, par exemple. *us-east-1*

```
aws emr-containers start-job-run \
--virtual-cluster-id cluster_id \
--name sample-job-name \
--execution-role-arn execution-role-arn \
--release-label emr-6.7.0-latest \
--job-driver '{
  "sparkSqlJobDriver": {
    "entryPoint": "s3://query-file.sql",
    "sparkSqlParameters": "--conf spark.executor.instances=2 --
conf spark.executor.memory=2G --conf spark.executor.cores=2 --conf
 spark.driver.cores=1"
  }
}' \
--configuration-overrides '{
  "monitoringConfiguration": {
    "cloudWatchMonitoringConfiguration": {
      "logGroupName": "my_log_group_name",
      "logStreamNamePrefix": "my_log_stream_prefix"
    },
    "s3MonitoringConfiguration": {
       "logUri": "s3://amzn-s3-demo-logging-bucket"
    }
  }
}'
```

Un exemple de fichier de requête SQL est présenté ci-dessous. Vous devez disposer d'un magasin de fichiers externe, tel que S3, où les données des tables sont stockées.

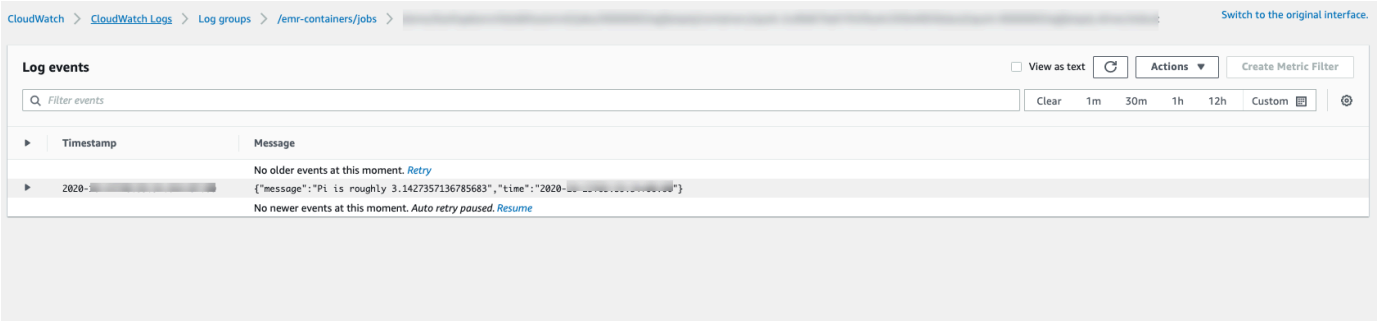
```
CREATE DATABASE demo;
CREATE EXTERNAL TABLE IF NOT EXISTS demo.amazonreview( marketplace string,
 customer_id string, review_id  string, product_id  string, product_parent  string,
 product_title  string, star_rating  integer, helpful_votes  integer, total_votes
  integer, vine  string, verified_purchase  string, review_headline  string,
 review_body  string, review_date  date, year  integer) STORED AS PARQUET LOCATION
 's3://URI to parquet files';
SELECT count(*) FROM demo.amazonreview;
SELECT count(*) FROM demo.amazonreview WHERE star_rating = 3;
```

La sortie de cette tâche sera disponible dans les journaux stdout du pilote dans S3 ou CloudWatch, `monitoringConfiguration` selon la configuration.

4. Vous pouvez également créer un fichier JSON avec des paramètres spécifiques pour l'exécution de votre tâche. Exécutez ensuite la commande start-job-run avec un chemin d'accès au fichier JSON. Pour plus d'informations, consultez la rubrique Soumission d'une tâche. Pour plus d'informations sur la configuration des paramètres d'exécution d'une tâche, consultez la rubrique Options de configuration d'une exécution de tâche.

   Pour suivre la progression de la tâche ou corriger les échecs, vous pouvez inspecter les journaux chargés sur Amazon S3, CloudWatch les journaux ou les deux. Reportez-vous au chemin du journal dans Amazon S3 à la section Configurer l'exécution d'une tâche pour utiliser les journaux S3 et pour les journaux Cloudwatch à la section Configurer une exécution de tâche pour utiliser CloudWatch les journaux. Pour voir les CloudWatch journaux dans Logs, suivez les instructions ci-dessous.

   - Ouvrez la CloudWatch console à l'adresse https://console.aws.amazon.com/cloudwatch/.
   - Dans le volet Navigation, choisissez Journaux. Puis choisissez Groupes de journaux.
   - Choisissez le groupe de journaux pour Amazon EMR on EKS, puis consultez les événements du journal chargés.



> **Important**
>
> Les tâches ont une politique de relance configurée par défaut. Pour plus d'informations sur la modification ou la désactivation de la configuration, consultez la rubrique Utilisation des politiques de relance des tâches.

Liens vers les guides des meilleures pratiques d'Amazon EMR on EKS sur GitHub

Nous avons élaboré le guide des meilleures pratiques Amazon EMR on EKS en utilisant la collaboration communautaire open source afin de pouvoir itérer rapidement et fournir des recommandations sur les aspects de la création et de l'exécution d'un cluster virtuel. Nous vous recommandons d'utiliser le Guide des bonnes pratiques Amazon EMR on EKS pour les sections concernées. Choisissez les liens dans chaque section pour accéder au GitHub site.

Sécurité

> **Note**
>
> Pour plus d'informations sur la sécurité avec Amazon EMR on EKS, consultez Bonnes pratiques de sécurité pour Amazon EMR on EKS.

Bonnes pratiques en matière de chiffrement : comment utiliser le chiffrement pour les données au repos et en transit.

Gestion de la sécurité du réseau : explique comment configurer les groupes de sécurité pour les pods d'Amazon EMR on EKS lorsque vous vous connectez à des sources de données hébergées dans des Services AWS comme Amazon RDS et Amazon Redshift.

Utiliser le gestionnaire de AWS secrets pour stocker des secrets.

Soumission de tâches PySpark

Soumission de tâches PySpark : spécifie différents types d'empaquetage pour les applications PySpark en utilisant des formats d'empaquetage tels que zip, egg, wheel et pex.

Stockage

Utilisation des volumes EBS : comment utiliser le provisionnement statique et dynamique pour les tâches nécessitant des volumes EBS.

Utilisation d'Amazon FSx pour les volumes Lustre : comment utiliser le provisionnement statique et dynamique pour les tâches nécessitant des volumes Amazon FSx for Luster.

Utilisation des volumes de stockage d'instances : comment utiliser les volumes de stockage d'instances pour le traitement des tâches.

Intégration de métastore

Utilisation du métastore Hive : propose différentes manières d'utiliser le métastore Hive.

Utiliser AWS Glue : propose différentes manières de configurer le catalogue AWS Glue.

Débogage

Utilisation du débogage Spark : comment modifier le niveau de journalisation.

Connexion à l'interface utilisateur Spark sur le pod pilote.

Utilisation du serveur d'historique Spark auto-hébergé avec Amazon EMR on EKS.

Résolution des problèmes liés à Amazon EMR on EKS

Résolution des problèmes.

Placement des nœuds

Utilisation des sélecteurs de nœuds Kubernetes pour `single-az` et d'autres cas d'utilisation.

Utilisation du placement des nœuds Fargate.

Performance

Utilisation de l'allocation dynamique des ressources (DRA).

Bonnes pratiques EKS relatives au plug-in Amazon VPC Container Network Interface (CNI), Cluster Autoscaler et Core DNS.

Optimisation des coûts

Utilisation d'instances ponctuelles : bonnes pratiques relatives aux instances EC2 ponctuelles Amazon et comment utiliser la fonctionnalité de mise hors service des nœuds Spark.

En utilisant AWS Outposts

Exécution d'Amazon EMR sur EKS à l'aide de AWS Outposts

Personnalisation d'images Docker pour Amazon EMR on EKS

Vous pouvez utiliser des images Docker personnalisées avec Amazon EMR on EKS. La personnalisation de l'image d'exécution Amazon EMR on EKS présente les avantages suivants :

- Regroupez les dépendances et le moteur d'exécution de l'application dans un seul conteneur immuable qui favorise la portabilité et simplifie la gestion des dépendances pour chaque charge de travail.
- Installez et configurez des packages optimisés pour vos charges de travail. Il est possible que ces packages ne soient pas largement disponibles dans la distribution publique des moteurs d'exécution Amazon EMR.
- Intégrez Amazon EMR on EKS aux processus de création, de test et de déploiement existants au sein de votre organisation, y compris le développement et les tests locaux.
- Appliquer des processus de sécurité établis, tels que la numérisation d'images, qui répondent aux exigences de conformité et de gouvernance au sein de votre organisation.

Rubriques

Instructions de personnalisation des images Docker

Suivez ces étapes pour personnaliser les images Docker pour Amazon EMR sur EKS. Les étapes vous indiquent comment obtenir une image de base, la personnaliser et la publier, et soumettre une charge de travail à l'aide de cette image.

- Étape 4 : Publier une image personnalisée
- Étape 5 : Soumettre une charge de travail Spark dans Amazon EMR à l'aide d'une image personnalisée

> **Note**
>
> Parmi les autres options que vous pouvez envisager lors de la personnalisation des images Docker, citons la personnalisation pour les points de terminaison interactifs, afin de vous assurer que vous disposez des dépendances requises, ou l'utilisation d'images de conteneurs multi-architecturales :
>
> - Personnalisation des images Docker pour les points de terminaison interactifs
> - Utilisation d'images multi-architectures

Conditions préalables

- Suivez les étapes Configuration d'Amazon EMR on EKS pour Amazon EMR on EKS.
- Installez Docker dans votre environnement. Pour plus d'informations, consultez Obtenir Docker.

Étape 1 : Récupération d'une image de base à partir d'Amazon Elastic Container Registry (Amazon ECR)

L'image de base contient le moteur d'exécution Amazon EMR et les connecteurs utilisés pour accéder à d'autres services AWS . Pour Amazon EMR 6.9.0 et les versions ultérieures, vous pouvez obtenir les images de base à partir de la galerie publique d'Amazon ECR. Parcourez la galerie pour trouver le lien de l'image et extrayez l'image dans votre espace de travail local. Par exemple, pour la version 7.12.0 d'Amazon EMR, la `docker pull` commande suivante permet d'obtenir la dernière image de base standard. Vous pouvez remplacer `emr-7.12.0:latest` par `emr-7.12.0-spark-rapids:latest` pour récupérer l'image qui possède l'accélérateur Nvidia RAPIDS. Vous pouvez également remplacer `emr-7.12.0:latest` par `emr-7.12.0-java11:latest` pour récupérer l'image avec le moteur d'exécution Java 11.

```
docker pull public.ecr.aws/emr-on-eks/spark/emr-7.12.0:latest
```

Si vous souhaitez récupérer l'image de base d'Amazon EMR 6.9.0 ou de versions antérieures, ou si vous préférez récupérer l'image à partir des comptes de registre Amazon ECR de chaque région, procédez comme suit :

1. Choisissez l'URI de l'image de base. Le format de l'URI de l'image est *ECR-registry-account*.dkr.ecr.*Region*.amazonaws.com/spark/*container-image-tag*, comme le montre l'exemple ci-dessous.

   ```
   895885662937.dkr.ecr.us-west-2.amazonaws.com/spark/emr-6.6.0:latest
   ```

   Pour choisir une image de base dans votre région, consultez Détails relatifs à la sélection d'une URI d'image de base.
2. Connectez-vous au référentiel Amazon ECR dans lequel l'image de base est stockée. Remplacez *895885662937* et *us-west-2* par le compte de registre Amazon ECR et la AWS région que vous avez sélectionnée.

   ```
   aws ecr get-login-password --region us-west-2 | docker login --username AWS --
   password-stdin 895885662937.dkr.ecr.us-west-2.amazonaws.com
   ```
3. Extrayez l'image de base dans votre espace de travail local. *emr-6.6.0:latest*Remplacez-le par le tag d'image du conteneur que vous avez sélectionné.

   ```
   docker pull 895885662937.dkr.ecr.us-west-2.amazonaws.com/spark/emr-6.6.0:latest
   ```

Étape 2 : Personnaliser une image de base

Suivez ces étapes pour personnaliser l'image de base que vous avez extraite d'Amazon ECR.

1. Créez un nouveau espace de travail `Dockerfile` sur votre espace de travail local.
2. Modifiez le `Dockerfile` que vous venez de créer et ajoutez le contenu qui suit.
   Ce `Dockerfile` utilise l'image du conteneur que vous avez extrait à partir de `895885662937.dkr.ecr.us-west-2.amazonaws.com/spark/emr-6.6.0:latest`.

   ```
   FROM 895885662937.dkr.ecr.us-west-2.amazonaws.com/spark/emr-6.6.0:latest
   USER root
   ### Add customization commands here ####
   USER hadoop:hadoop
   ```

3. Ajoutez des commandes dans le `Dockerfile` pour personnaliser l'image de base. Par exemple, ajoutez une commande pour installer les bibliothèques Python, comme le montre le fichier `Dockerfile` ci-dessous.

```
FROM 895885662937.dkr.ecr.us-west-2.amazonaws.com/spark/emr-6.6.0:latest
USER root
RUN pip3 install --upgrade boto3 pandas numpy // For python 3
USER hadoop:hadoop
```

4. À partir du répertoire où le `Dockerfile` est créé, exécutez la commande suivante pour créer l'image Docker. Donnez un nom à l'image Docker, *`emr6.6_custom`* par exemple.

```
docker build -t emr6.6_custom .
```

Étape 3 : (facultative, mais recommandée) Valider une image personnalisée

Nous vous recommandons de tester la compatibilité de votre image personnalisée avant de la publier. Vous pouvez utiliser l'interface CLI pour les images personnalisées d'Amazon EMR on EKS pour vérifier si votre image possède les structures de fichiers requises et les configurations correctes pour être exécutée sur Amazon EMR on EKS.

> **Note**
>
> L'interface CLI pour les images personnalisées d'Amazon EMR on EKS ne peut pas confirmer que votre image est exempte d'erreur. Faites attention lorsque vous supprimez des dépendances des images de base.

Suivez les étapes ci-dessous pour valider votre image personnalisée.

1. Téléchargez et installez l'interface CLI pour les images personnalisées d'Amazon EMR on EKS. Pour plus d'informations, consultez le Guide d'installation de l'interface CLI pour les images personnalisées d'Amazon EMR on EKS.
2. Exécutez la commande suivante pour tester l'installation.

```
emr-on-eks-custom-image --version
```

   Voici un exemple de résultat.

```
Amazon EMR on EKS Custom Image CLI
Version: x.xx
```

3. Exécutez la commande suivante pour valider votre image personnalisée.

```
emr-on-eks-custom-image validate-image -i image_name -r release_version [-
t image_type]
```

- `-i` indique l'URI de l'image locale qui doit être validée. Il peut s'agir de l'URI de l'image, d'un nom ou d'une balise que vous avez défini pour votre image.
- `-r` indique la version exacte de l'image de base, par exemple, `emr-6.6.0-latest`.
- `-t` indique le type d'image. S'il s'agit d'une image Spark, saisissez `spark`. La valeur par défaut est `spark`. La version actuelle de l'interface CLI pour les images personnalisées d'Amazon EMR on EKS ne prend en charge que les images d'exécution Spark.

Si vous exécutez la commande avec succès et que l'image personnalisée respecte toutes les configurations et structures de fichiers requises, le résultat renvoyé affiche les résultats de tous les tests, comme le montre l'exemple ci-dessous.

```
Amazon EMR on EKS Custom Image Test
Version: x.xx
... Checking if docker cli is installed
... Checking Image Manifest
[INFO] Image ID: xxx
[INFO] Created On: 2021-05-17T20:50:07.986662904Z
[INFO] Default User Set to hadoop:hadoop : PASS
[INFO] Working Directory Set to /home/hadoop : PASS
[INFO] Entrypoint Set to /usr/bin/entrypoint.sh : PASS
[INFO] SPARK_HOME is set with value: /usr/lib/spark : PASS
[INFO] JAVA_HOME is set with value: /etc/alternatives/jre : PASS
[INFO] File Structure Test for spark-jars in /usr/lib/spark/jars: PASS
[INFO] File Structure Test for hadoop-files in /usr/lib/hadoop: PASS
[INFO] File Structure Test for hadoop-jars in /usr/lib/hadoop/lib: PASS
[INFO] File Structure Test for bin-files in /usr/bin: PASS
... Start Running Sample Spark Job
[INFO] Sample Spark Job Test with local:///usr/lib/spark/examples/jars/spark-
examples.jar : PASS
----------------------------------------------------------------
Overall Custom Image Validation Succeeded.
```

```
----------------------------------------------------------------
```

Si l'image personnalisée ne répond pas aux configurations ou aux structures de fichiers requises, des messages d'erreur apparaissent. Le résultat renvoyé fournit des informations sur les configurations ou les structures de fichiers incorrectes.

Étape 4 : Publier une image personnalisée

Publiez la nouvelle image Docker dans votre registre Amazon ECR.

1. Exécutez la commande suivante pour créer un référentiel Amazon ECR pour stocker votre image Docker. Donnez un nom à votre dépôt, par exemple,*emr6.6_custom_repo*. Remplacez *us-west-2* par votre région.

   ```
   aws ecr create-repository \
       --repository-name emr6.6_custom_repo \
       --image-scanning-configuration scanOnPush=true \
       --region us-west-2
   ```

   Pour plus d'informations, consultez la rubrique Création d'un référentiel dans le Guide de l'utilisateur Amazon ECR.
2. Exécutez la commande suivante pour vous authentifier dans votre registre par défaut.

   ```
   aws ecr get-login-password --region us-west-2 | docker login --username AWS --
   password-stdin aws_account_id.dkr.ecr.us-west-2.amazonaws.com
   ```

   Pour plus d'informations, consultez la rubrique Authentification dans votre registre par défaut dans le Guide de l'utilisateur Amazon ECR.
3. Marquez et publiez une image dans le référentiel Amazon ECR que vous avez créé.

   Balisez l'image.

   ```
   docker tag emr6.6_custom aws_account_id.dkr.ecr.us-
   west-2.amazonaws.com/emr6.6_custom_repo
   ```

   Transmettez l'image.

```
docker push aws_account_id.dkr.ecr.us-west-2.amazonaws.com/emr6.6_custom_repo
```

Pour plus d'informations, consultez la rubrique Transmission d'une image à Amazon ECR dans le Guide de l'utilisateur Amazon ECR.

Étape 5 : Soumettre une charge de travail Spark dans Amazon EMR à l'aide d'une image personnalisée

Une fois qu'une image personnalisée a été créée et publiée, vous pouvez soumettre une tâche Amazon EMR on EKS à l'aide d'une image personnalisée.

Créez d'abord un fichier start-job-run-request .json et spécifiez le `spark.kubernetes.container.image` paramètre pour référencer l'image personnalisée, comme le montre l'exemple de fichier JSON suivant.

> **Note**
>
> Vous pouvez utiliser le schéma `local://` pour faire référence aux fichiers disponibles dans l'image personnalisée, comme le montre l'argument `entryPoint` dans l'extrait JSON ci-dessous. Vous pouvez également utiliser le schéma `local://` pour faire référence aux dépendances des applications. Tous les fichiers et dépendances auxquels il est fait référence à l'aide du schéma `local://` doivent déjà être présents au chemin spécifié dans l'image personnalisée.

```
{
    "name": "spark-custom-image",
    "virtualClusterId": "virtual-cluster-id",
    "executionRoleArn": "execution-role-arn",
    "releaseLabel": "emr-6.6.0-latest",
    "jobDriver": {
      "sparkSubmitJobDriver": {
        "entryPoint": "local:///usr/lib/spark/examples/jars/spark-examples.jar",
        "entryPointArguments": [
                  "10"
              ],
```

```
        "sparkSubmitParameters": "--class org.apache.spark.examples.SparkPi --conf
 spark.kubernetes.container.image=123456789012.dkr.ecr.us-west-2.amazonaws.com/
emr6.6_custom_repo"
       }
    }
}
```

Vous pouvez également référencer l'image personnalisée à l'aide des propriétés `applicationConfiguration`, comme le montre l'exemple ci-dessous.

```
{
    "name": "spark-custom-image",
    "virtualClusterId": "virtual-cluster-id",
    "executionRoleArn": "execution-role-arn",
    "releaseLabel": "emr-6.6.0-latest",
    "jobDriver": {
      "sparkSubmitJobDriver": {
        "entryPoint": "local:///usr/lib/spark/examples/jars/spark-examples.jar",
        "entryPointArguments": [
                  "10"
             ],
         "sparkSubmitParameters": "--class org.apache.spark.examples.SparkPi"
       }
    },
    "configurationOverrides": {
        "applicationConfiguration": [
            {
                "classification": "spark-defaults",
                "properties": {
                    "spark.kubernetes.container.image": "123456789012.dkr.ecr.us-
west-2.amazonaws.com/emr6.6_custom_repo"
                }
            }
        ]
    }
}
```

Exécutez ensuite la commande `start-job-run` pour soumettre la tâche.

```
aws emr-containers start-job-run --cli-input-json file://./start-job-run-request.json
```

Dans les exemples JSON ci-dessus, remplacez-le *emr-6.6.0-latest* par la version de votre version d'Amazon EMR. Nous vous recommandons vivement d'utiliser la version `-latest` pour vous assurer que la version sélectionnée contient les dernières mises à jour de sécurité. Pour plus d'informations sur les versions Amazon EMR et leurs balises d'image, consultez Détails relatifs à la sélection d'une URI d'image de base.

> **Note**
>
> Vous pouvez utiliser `spark.kubernetes.driver.container.image` et `spark.kubernetes.executor.container.image` indiquer une image différente pour les pods de pilote et d'exécuteur.

Personnalisation des images Docker pour les points de terminaison interactifs

Vous pouvez également personnaliser les images Docker pour les points de terminaison interactifs afin de pouvoir exécuter des images de noyau de base personnalisées. Cela vous permet de vous assurer que vous disposez des dépendances dont vous avez besoin lorsque vous exécutez des charges de travail interactives à partir d'EMR Studio.

1. Suivez les étapes 1 à 4 décrites ci-dessus pour personnaliser une image Docker. Pour les versions 6.9.0 et ultérieures d'Amazon EMR, vous pouvez obtenir l'URI de l'image de base à partir de la galerie publique d'Amazon ECR. Pour les versions antérieures à Amazon EMR 6.9.0, vous pouvez obtenir l'image dans les comptes du registre Amazon ECR de chaque Région AWS. La seule différence réside dans l'URI de l'image de base de votre Dockerfile. L'URI de l'image de base respecte le format ci-dessous :

   ```
   ECR-registry-account.dkr.ecr.Region.amazonaws.com/notebook-spark/container-image-tag
   ```

   Vous devez utiliser `notebook-spark` dans l'URI de l'image de base, au lieu de `spark`. L'image de base contient le moteur d'exécution Spark et les noyaux de bloc-notes qui s'exécutent avec celui-ci. Pour plus d'informations sur la sélection des balises de régions et d'images de conteneurs, consultez Détails relatifs à la sélection d'une URI d'image de base.

> **Note**
>
> Actuellement, seuls les remplacements d'images de base sont pris en charge et l'introduction de nouveaux noyaux d'autres types que ceux AWS fournis par les images de base n'est pas prise en charge.

2. Créez un point de terminaison interactif qui peut être utilisé avec l'image personnalisée.

   Tout d'abord, créez un fichier JSON appelé `custom-image-managed-endpoint.json` avec le contenu suivant.

```
{
    "name": "endpoint-name",
    "virtualClusterId": "virtual-cluster-id",
    "type": "JUPYTER_ENTERPRISE_GATEWAY",
    "releaseLabel": "emr-6.6.0-latest",
    "executionRoleArn": "execution-role-arn",
    "certificateArn": "certificate-arn",
    "configurationOverrides": {
        "applicationConfiguration": [
            {
                "classification": "jupyter-kernel-overrides",
                "configurations": [
                    {
                        "classification": "python3",
                        "properties": {
                            "container-image": "123456789012.dkr.ecr.us-
west-2.amazonaws.com/custom-notebook-python:latest"
                        }
                    },
                    {
                        "classification": "spark-python-kubernetes",
                        "properties": {
                            "container-image": "123456789012.dkr.ecr.us-
west-2.amazonaws.com/custom-notebook-spark:latest"
                        }
                    }
                ]
            }
        ]
    }
```

```
}
```

Ensuite, créez un point de terminaison interactif en utilisant les configurations spécifiées dans le fichier JSON, comme le montre l'exemple ci-dessous.

```
aws emr-containers create-managed-endpoint --cli-input-json custom-image-managed-
endpoint.json
```

Pour plus d'informations, consultez la rubrique Création d'un point de terminaison interactif pour votre cluster virtuel.

3. Connectez-vous au point de terminaison interactif via EMR Studio. Pour plus d'informations, consultez la rubrique Connexion à partir de Studio.

Utilisation d'images multi-architectures

Amazon EMR on EKS prend en charge les images de conteneurs multi-architectures pour Amazon Elastic Container Registry (Amazon ECR). Pour plus d'informations, consultez la rubrique Présentation des images de conteneurs multi-architectures pour Amazon ECR.

Les images personnalisées Amazon EMR on EKS prennent en charge à la fois les instances EC2 AWS basées sur Graviton et les instances EC2. non-Graviton-based Les images basées sur Graviton sont stockées dans les mêmes référentiels d'images d'Amazon ECR que les images. non-Graviton-based

Par exemple, pour inspecter la liste des manifestes Docker pour les images 6.6.0, exécutez la commande ci-dessous.

```
docker manifest inspect 895885662937.dkr.ecr.us-west-2.amazonaws.com/spark/
emr-6.6.0:latest
```

Voici la sortie. L'architecture `arm64` est destinée aux instances Graviton. L'architecture `amd64` est destinée aux instances non-Graviton.

```
{
   "schemaVersion": 2,
   "mediaType": "application/vnd.docker.distribution.manifest.list.v2+json",
   "manifests": [
      {
```

```
         "mediaType": "application/vnd.docker.distribution.manifest.v2+json",
         "size": 1805,
         "digest":
 "xxx123:6b971cb47d11011ab3d45fff925e9442914b4977ae0f9fbcdcf5cfa99a7593f0",
         "platform": {
            "architecture": "arm64",
            "os": "linux"
         }
      },
      {
         "mediaType": "application/vnd.docker.distribution.manifest.v2+json",
         "size": 1805,
         "digest":
 "xxx123:6f2375582c9c57fa9838c1d3a626f1b4fc281e287d2963a72dfe0bd81117e52f",
         "platform": {
            "architecture": "amd64",
            "os": "linux"
         }
      }
   ]
}
```

Pour créer des images multi-architectures, procédez comme suit :

1. Créez un `Dockerfile` avec le contenu suivant afin de pouvoir extraire l'image `arm64`.

   ```
   FROM --platform=arm64 895885662937.dkr.ecr.us-west-2.amazonaws.com/spark/
   emr-6.6.0:latest
   USER root

   RUN pip3 install boto3 // install customizations here
   USER hadoop:hadoop
   ```

2. Suivez les instructions de la rubrique Présentation des images de conteneurs multi-architectures pour Amazon ECR pour créer une image multi-architectures.

   > **Note**
   >
   > Vous devez créer des images `arm64` sur les instances `arm64`. De même, vous devez créer des images amd64 sur des instances amd64.

Grâce à la commande Docker `buildx`, nous pouvez également créer des images multi-architectures sans utiliser chaque type d'instance spécifique. Pour plus d'informations, consultez la rubrique Exploitation de la prise en charge de l'architecture multi-CPU.

3. Après avoir créé l'image multi-architectures, vous pouvez soumettre une tâche avec le même paramètre `spark.kubernetes.container.image` et la pointer vers l'image. Dans un cluster hétérogène comprenant à la fois des instances AWS basées sur Graviton et des instances non-Graviton-based EC2, l'instance détermine l'image d'architecture correcte en fonction de l'architecture d'instance qui extrait l'image.

Détails relatifs à la sélection d'une URI d'image de base

> **Note**
>
> Pour les versions 6.9.0 et ultérieures d'Amazon EMR, vous pouvez récupérer l'image de base à partir de la galerie publique d'Amazon ECR. Il n'est donc pas nécessaire de construire l'URI de l'image de base comme l'indiquent les instructions de cette page. Pour trouver la balise d'image de conteneur pour votre image de base, reportez-vous à la page des notes de mise à jour de la version correspondante d'Amazon EMR on EKS.

Les images Docker de base que vous pouvez sélectionner sont stockées dans Amazon Elastic Container Registry (Amazon ECR). Le format de l'URI de l'image est *ECR-registry-account*`.dkr.ecr.`*Region*`.amazonaws.com/spark/`*container-image-tag*, comme le montre l'exemple ci-dessous.

```
895885662937.dkr.ecr.us-west-2.amazonaws.com/spark/emr-7.12.0:latest
```

Le format de l'URI de l'image des points de terminaison interactifs est *ECR-registry-account*`.dkr.ecr.`*Region*`.amazonaws.com/notebook-spark/`*container-image-tag*, comme le montre l'exemple ci-dessous. Vous devez utiliser `notebook-spark` dans l'URI de l'image de base, au lieu de `spark`.

```
895885662937.dkr.ecr.us-west-2.amazonaws.com/notebook-spark/emr-7.12.0:latest
```

De même, pour les images python3 non-Spark des points de terminaison interactifs, l'URI de l'image est *ECR-registry-account*.dkr.ecr.*Region*.amazonaws.com/notebook-python/*container-image-tag*. L'exemple d'URI suivant est correctement formaté :

```
895885662937.dkr.ecr.us-west-2.amazonaws.com/notebook-python/emr-7.12.0:latest
```

Pour trouver la balise d'image de conteneur pour votre image de base, reportez-vous à la page des notes de mise à jour de la version correspondante d'Amazon EMR on EKS.

Comptes de registre Amazon ECR par région

Pour éviter une latence réseau élevée, extrayez une image de base de l'image la plus proche Région AWS. Sélectionnez le compte de registre Amazon ECR correspondant à la région d'où vous extrayez l'image en vous basant sur le tableau suivant.

Régions	Comptes de registre Amazon ECR
ap-east-1	736135916053
ap-northeast-1	059004520145
ap-northeast-2	996579266876
ap-northeast-3	705689932349
ap-southeast-3	946962994502
ap-south-1	235914868574
ap-south-2	691480105545
ap-southeast-1	671219180197
ap-southeast-2	038297999601
ca-central-1	351826393999
eu-central-1	107292555468

Régions	Comptes de registre Amazon ECR	
eu-central-2	314408114945	
eu-north-1	830386416364	
eu-west-1	483788554619	
eu-west-2	118780647275	
eu-west-3	307523725174	
eu-south-1	238014973495	
eu-south-2	350796622945	
il-central-1	395734710648	
me-south-1	008085056818	
me-central-1	818935616732	
sa-east-1	052806832358	
us-gov-west-1	299385240661	
us-gov-east-1	299393998622	
us-east-1	755674844232	
us-east-2	711395599931	
us-west-1	608033475327	
us-west-2	895885662937	
af-south-1	358491847878	
cn-north-1	068337069695	
cn-northwest-1	068420816659	

Considérations relatives à la personnalisation des images

Lorsque vous personnalisez des images Docker, vous pouvez choisir précisément l'environnement d'exécution de votre tâche de manière détaillée. Tenez compte de ces meilleures pratiques lorsque vous utilisez cette fonctionnalité. Il s'agit notamment de considérations relatives à la sécurité, à la configuration et au montage d'une image :

- La sécurité est une responsabilité partagée entre vous AWS et vous. Vous êtes responsable de l'application des correctifs de sécurité aux binaires que vous ajoutez à l'image. Suivez les Bonnes pratiques de sécurité pour Amazon EMR on EKS, en particulier Obtention des dernières mises à jour de sécurité des images personnalisées et Application du principe du moindre privilège.
- Lorsque vous personnalisez une image de base, vous devez modifier l'utilisateur Docker en `hadoop:hadoop` afin que les tâches ne s'exécutent pas avec l'utilisateur root.
- Amazon EMR on EKS monte les fichiers au-dessus des configurations de l'image, telles que `spark-defaults.conf`, au moment de l'exécution. Pour remplacer ces fichiers de configuration, nous vous recommandons d'utiliser le paramètre `applicationOverrides` lors de la soumission de la tâche et de ne pas modifier directement les fichiers de l'image personnalisée.
- Amazon EMR on EKS monte certains dossiers au moment de l'exécution. Les modifications que vous apportez à ces dossiers ne sont pas disponibles dans le conteneur. Si vous souhaitez ajouter une application ou ses dépendances pour les images personnalisées, nous vous recommandons de choisir un répertoire qui ne fait pas partie des chemins prédéfinis suivants :
  - `/var/log/fluentd`
  - `/var/log/spark/user`
  - `/var/log/spark/apps`
  - `/mnt`
  - `/tmp`
  - `/home/hadoop`
- Vous pouvez charger votre image personnalisée dans n'importe quel référentiel compatible avec Docker, tel qu'Amazon ECR, Docker Hub ou un référentiel d'entreprise privé. Pour plus d'informations sur la configuration de l'authentification du cluster Amazon EKS dans le référentiel Docker sélectionné, consultez la rubrique Extraction d'une image à partir d'un registre privé.

Exécution de tâches Flink à l'aide d'Amazon EMR on EKS

Les versions 6.13.0 et ultérieures d'Amazon EMR prennent en charge Amazon EMR sur EKS avec Apache Flink, ou l'opérateur Kubernetes pour Flink, en tant que modèle de soumission de tâches pour Amazon EMR sur EKS. Avec Amazon EMR on EKS associé à Apache Flink, vous pouvez déployer et gérer des applications Flink en utilisant le moteur d'exécution de la version Amazon EMR au sein de vos clusters Amazon EKS personnels. Une fois que vous avez déployé l'opérateur Kubernetes pour Flink dans votre cluster Amazon EKS, vous pouvez directement soumettre des applications Flink à l'aide de cet opérateur. L'opérateur gère le cycle de vie des applications Flink.

Rubriques

Configuration et utilisation de l'opérateur Flink Kubernetes

Les pages suivantes décrivent comment configurer et utiliser l'opérateur Kubernetes pour Flink pour exécuter des tâches Flink avec Amazon EMR on EKS. Les sujets disponibles incluent les prérequis requis, la configuration de votre environnement et l'exécution d'une application Flink sur Amazon EMR sur EKS.

Rubriques

Configuration de l'opérateur Kubernetes pour Flink sur Amazon EMR on EKS

Effectuez les tâches suivantes pour vous préparer avant d'installer l'opérateur Kubernetes pour Flink sur Amazon EKS. Si vous êtes déjà inscrit à Amazon Web Services (AWS) et que vous avez utilisé Amazon EKS, vous êtes presque prêt à utiliser Amazon EMR on EKS. Effectuez les tâches suivantes pour vous préparer pour l'utilisation de l'opérateur Flink sur Amazon EKS. Si vous avez déjà rempli l'une des conditions préalables, vous pouvez l'ignorer et passer à la suivante.

- Installation ou mise à jour vers la dernière version du AWS CLI — Si vous avez déjà installé le AWS CLI, vérifiez que vous disposez de la dernière version.
- Configurer kubectl et eksctl — eksctl est un outil de ligne de commande que vous utilisez pour communiquer avec Amazon EKS.
- Installer Helm – Le gestionnaire de packages Helm pour Kubernetes vous aide à installer et à gérer des applications sur votre cluster Kubernetes.
- Commencez avec Amazon EKS — eksctl — Suivez les étapes pour créer un nouveau cluster Kubernetes avec des nœuds dans Amazon EKS.
- Choisissez une étiquette de version Amazon EMR (version 6.13.0 ou supérieure) : l'opérateur Flink Kubernetes est pris en charge par les versions 6.13.0 et supérieures d'Amazon EMR.
- Activez les rôles IAM pour les comptes de service (IRSA) sur le cluster Amazon EKS.
- Créez un rôle d'exécution des tâches.
- Mettez à jour la politique d'approbation du rôle d'exécution des tâches.
- Créez un rôle d'exécution d'opérateur. Cette étape est facultative. Vous pouvez utiliser le même rôle pour les tâches et l'opérateur Flink. Si vous souhaitez attribuer un rôle IAM différent à votre opérateur, vous pouvez créer un rôle distinct.
- Mettez à jour la politique d'approbation du rôle d'exécution de l'opérateur. Vous devez ajouter explicitement une entrée de politique d'approbation pour les rôles que vous souhaitez utiliser pour le compte de service de l'opérateur Kubernetes pour Flink sur Amazon EMR. Vous pouvez suivre cet exemple de format :

  JSON

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

```
    {
      "Effect": "Allow",
      "Action": [
        "sts:AssumeRoleWithWebIdentity"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "StringLike": {
          "aws:userid": "system:serviceaccount:emr:emr-containers-sa-flink-
operator"
        }
      },
      "Sid": "AllowSTSAssumerolewithwebidentity"
    }
  ]
}
```

Installation de l'opérateur Flink Kubernetes pour Amazon EMR sur EKS

Cette rubrique vous aide à commencer à utiliser l'opérateur Flink Kubernetes sur Amazon EKS en préparant un déploiement Flink.

Installez l'opérateur Kubernetes

Procédez comme suit pour installer l'opérateur Kubernetes pour Apache Flink.

1. Si vous ne l'avez pas déjà fait, suivez les étapes de the section called “Configuration”.
2. Installez le *cert-manager* (une fois par cluster Amazon EKS) pour permettre l'ajout du composant webhook.

   ```
   kubectl apply -f https://github.com/cert-manager/cert-manager/releases/download/
   v1.12.0/cert-manager.yaml
   ```

3. Installez les Charts de Helm.

   ```
   export VERSION=7.12.0 # The Amazon EMR release version
   export NAMESPACE=The Kubernetes namespace to deploy the operator

   helm install flink-kubernetes-operator \
   ```

```
oci://public.ecr.aws/emr-on-eks/flink-kubernetes-operator \
--version $VERSION \
--namespace $NAMESPACE
```

Exemple de sortie :

```
NAME: flink-kubernetes-operator
LAST DEPLOYED: Tue May 31 17:38:56 2022
NAMESPACE: $NAMESPACE
STATUS: deployed
REVISION: 1
TEST SUITE: None
```

4. Attendez que le déploiement soit terminé et vérifiez l'installation des charts.

```
kubectl wait deployment flink-kubernetes-operator --namespace $NAMESPACE --for
 condition=Available=True --timeout=30s
```

5. Le message suivant devrait s'afficher lorsque le déploiement est terminé.

```
deployment.apps/flink-kubernetes-operator condition met
```

6. Utilisez la commande suivante pour voir l'opérateur déployé.

```
helm list --namespace $NAMESPACE
```

Voici un exemple de sortie, où la version de l'application `x.y.z-amzn-n` correspondrait à la version de l'opérateur Flink pour votre version Amazon EMR sur EKS. Pour de plus amples informations, veuillez consulter Versions prises en charge pour Amazon EMR sur EKS avec Apache Flink.

```
NAME                            NAMESPACE    REVISION    UPDATED
               STATUS      CHART                                    APP VERSION

flink-kubernetes-operator    $NAMESPACE   1           2023-02-22 16:43:45.24148
 -0500 EST    deployed    flink-kubernetes-operator-emr-7.12.0    x.y.z-amzn-n
```

Mettre à niveau l'opérateur Kubernetes

Pour mettre à jour la version de l'opérateur Flink, procédez comme suit :

1. Désinstallez l'ancien `flink-kubernetes-operator` :`helm uninstall flink-kubernetes-operator -n <NAMESPACE>`.
2. Supprimer le CRD (puisque helm ne supprimera pas automatiquement l'ancien CRD) :. `kubectl delete crd flinkdeployments.flink.apache.org flinksessionjobs.flink.apache.org`
3. Réinstallez `flink-kubernetes-operator` avec la version la plus récente.

Exécution d'une application Flink

Avec Amazon EMR 6.13.0 et versions ultérieures, vous pouvez exécuter une application Flink avec l'opérateur Kubernetes pour Flink en mode application sur Amazon EMR sur EKS. Avec Amazon EMR 6.15.0 et versions ultérieures, vous pouvez également exécuter une application Flink en mode session. Cette section présente plusieurs manières d'exécuter une application Flink avec Amazon EMR sur EKS.

> **Note**
>
> Il est nécessaire de disposer d'un compartiment Amazon S3 préalablement créé pour conserver les métadonnées de haute disponibilité lorsque vous soumettez votre tâche Flink. Si vous ne souhaitez pas utiliser cette fonctionnalité, vous pouvez la désactiver. Elle est activée par défaut.

Prérequis : pour pouvoir exécuter une application Flink avec l'opérateur Kubernetes pour Flink, procédez comme suit dans the section called "Configuration" et the section called "Installez l'opérateur Kubernetes".

Application mode

Avec Amazon EMR 6.13.0 et versions ultérieures, vous pouvez exécuter une application Flink avec l'opérateur Kubernetes pour Flink en mode application sur Amazon EMR sur EKS.

1. Créez un fichier de `FlinkDeployment` définition `basic-example-app-cluster.yaml` comme dans l'exemple suivant. Si vous avez activé et utilisé l'un des opt-in Régions AWS, assurez-vous de décommenter et de configurer la configuration. `fs.s3a.endpoint.region`

```
apiVersion: flink.apache.org/v1beta1
```

```
kind: FlinkDeployment
metadata:
  name: basic-example-app-cluster
spec:
  flinkConfiguration:
    taskmanager.numberOfTaskSlots: "2"
    #fs.s3a.endpoint.region: OPT_IN_AWS_REGION_NAME
    state.checkpoints.dir: CHECKPOINT_S3_STORAGE_PATH
    state.savepoints.dir: SAVEPOINT_S3_STORAGE_PATH
  flinkVersion: v1_17
  executionRoleArn: JOB_EXECUTION_ROLE_ARN
  emrReleaseLabel: "emr-6.13.0-flink-latest" # 6.13 or higher
  jobManager:
    storageDir: HIGH_AVAILABILITY_STORAGE_PATH
    resource:
      memory: "2048m"
      cpu: 1
  taskManager:
    resource:
      memory: "2048m"
      cpu: 1
  job:
    # if you have your job jar in S3 bucket you can use that path as well
    jarURI: local:///opt/flink/examples/streaming/StateMachineExample.jar
    parallelism: 2
    upgradeMode: savepoint
    savepointTriggerNonce: 0
  monitoringConfiguration:
    cloudWatchMonitoringConfiguration:
       logGroupName: LOG_GROUP_NAME
```

2. Soumettez le déploiement Flink à l'aide de la commande ci-dessous. Cela créera également un objet `FlinkDeployment` nommé `basic-example-app-cluster`.

```
kubectl create -f basic-example-app-cluster.yaml -n <NAMESPACE>
```

3. Accédez à l'interface utilisateur de Flink.

```
kubectl port-forward deployments/basic-example-app-cluster 8081 -n NAMESPACE
```

4. Ouvrez `localhost:8081` pour consulter vos tâches Flink localement.

5. Nettoyez la tâche. N'oubliez pas de nettoyer les artefacts S3 créés pour cette tâche, tels que le pointage de contrôle, la haute disponibilité, les métadonnées de pointage de sauvegarde et les journaux. CloudWatch

Pour plus d'informations sur la soumission de candidatures à Flink via l'opérateur Flink Kubernetes, consultez les exemples d'opérateurs Flink Kubernetes dans le dossier sur. `apache/flink-kubernetes-operator` GitHub

Session mode

Avec Amazon EMR 6.15.0 et versions ultérieures, vous pouvez exécuter une application Flink avec l'opérateur Kubernetes pour Flink en mode session sur Amazon EMR sur EKS.

1. Créez un fichier de `FlinkDeployment` définition nommé `basic-example-app-cluster.yaml` comme dans l'exemple suivant. Si vous avez activé et utilisé l'un des opt-in Régions AWS, assurez-vous de décommenter et de configurer la configuration. `fs.s3a.endpoint.region`

```
apiVersion: flink.apache.org/v1beta1
kind: FlinkDeployment
metadata:
  name: basic-example-session-cluster
spec:
  flinkConfiguration:
    taskmanager.numberOfTaskSlots: "2"
    #fs.s3a.endpoint.region: OPT_IN_AWS_REGION_NAME
    state.checkpoints.dir: CHECKPOINT_S3_STORAGE_PATH
    state.savepoints.dir: SAVEPOINT_S3_STORAGE_PATH
  flinkVersion: v1_17
  executionRoleArn: JOB_EXECUTION_ROLE_ARN
  emrReleaseLabel: "emr-6.15.0-flink-latest"
  jobManager:
    storageDir: HIGH_AVAILABILITY_S3_STORAGE_PATH
    resource:
      memory: "2048m"
      cpu: 1
  taskManager:
    resource:
      memory: "2048m"
      cpu: 1
  monitoringConfiguration:
    s3MonitoringConfiguration:
```

```
        logUri:
    cloudWatchMonitoringConfiguration:
        logGroupName: LOG_GROUP_NAME
```

2. Soumettez le déploiement Flink à l'aide de la commande ci-dessous. Cela créera également un objet `FlinkDeployment` nommé `basic-example-session-cluster`.

```
kubectl create -f basic-example-app-cluster.yaml -n NAMESPACE
```

3. Utilisez la commande suivante pour vérifier que le cluster de session `LIFECYCLE` est défini sur `STABLE` :

```
kubectl get flinkdeployments.flink.apache.org basic-example-session-cluster -
n NAMESPACE
```

   Voici un exemple de sortie :

```
NAME                              JOB STATUS   LIFECYCLE STATE
basic-example-session-cluster                          STABLE
```

4. Créez un fichier de définition de ressources personnalisé `FlinkSessionJob` `basic-session-job.yaml` avec l'exemple de contenu suivant :

```
apiVersion: flink.apache.org/v1beta1
kind: FlinkSessionJob
metadata:
  name: basic-session-job
spec:
  deploymentName: basic-session-deployment
  job:
    # If you have your job jar in an S3 bucket you can use that path.
    # To use jar in S3 bucket, set
    # OPERATOR_EXECUTION_ROLE_ARN (--set emrContainers.operatorExecutionRoleArn=
$OPERATOR_EXECUTION_ROLE_ARN)
    # when you install Spark operator
    jarURI: https://repo1.maven.org/maven2/org/apache/flink/flink-examples-
streaming_2.12/1.16.1/flink-examples-streaming_2.12-1.16.1-TopSpeedWindowing.jar
    parallelism: 2
    upgradeMode: stateless
```

5. Soumettez la tâche de session avec la commande ci-dessous. Cela créera également un objet `FlinkSessionJob` `basic-session-job`.

```
kubectl apply -f basic-session-job.yaml -n $NAMESPACE
```

6. Utilisez la commande suivante pour vérifier que le cluster de session `LIFECYCLE` est défini sur `STABLE`, et que `JOB STATUS` indique `RUNNING` :

   ```
   kubectl get flinkdeployments.flink.apache.org basic-example-session-cluster -
   n NAMESPACE
   ```

   Voici un exemple de sortie :

   ```
   NAME                              JOB STATUS   LIFECYCLE STATE
   basic-example-session-cluster     RUNNING      STABLE
   ```

7. Accédez à l'interface utilisateur de Flink.

   ```
   kubectl port-forward deployments/basic-example-session-cluster 8081 -n NAMESPACE
   ```

8. Ouvrez `localhost:8081` pour consulter vos tâches Flink localement.
9. Nettoyez la tâche. N'oubliez pas de nettoyer les artefacts S3 créés pour cette tâche, tels que le pointage de contrôle, la haute disponibilité, les métadonnées de pointage de sauvegarde et les journaux. CloudWatch

Autorisations relatives aux rôles de sécurité pour exécuter une application Flink

Cette rubrique décrit les rôles de sécurité pour le déploiement et l'exécution d'une application Flink. Deux rôles sont nécessaires pour gérer un déploiement et pour créer et gérer des tâches, le rôle d'opérateur et le rôle de poste. Cette rubrique les présente et répertorie leurs autorisations.

Contrôle d'accès basé sur les rôles

Pour déployer l'opérateur et exécuter des tâches Flink, nous devons créer deux rôles Kubernetes : un rôle d'opérateur et un rôle de tâche. Amazon EMR crée les deux rôles par défaut lorsque vous installez l'opérateur.

Rôle d'opérateur

Nous utilisons le rôle d'opérateur `flinkdeployments` pour gérer la création et la JobManager gestion de chaque tâche Flink et d'autres ressources, telles que les services.

Le nom par défaut du rôle d'opérateur est e`mr-containers-sa-flink-operator` et nécessite les autorisations ci-dessous.

```
rules:
- apiGroups:
  - ""
  resources:
  - pods
  - services
  - events
  - configmaps
  - secrets
  - serviceaccounts
  verbs:
  - '*'
- apiGroups:
  - rbac.authorization.k8s.io
  resources:
  - roles
  - rolebindings
  verbs:
  - '*'
- apiGroups:
  - apps
  resources:
  - deployments
  - deployments/finalizers
  - replicasets
  verbs:
  - '*'
- apiGroups:
  - extensions
  resources:
  - deployments
  - ingresses
  verbs:
  - '*'
- apiGroups:
```

```
  - flink.apache.org
  resources:
  - flinkdeployments
  - flinkdeployments/status
  - flinksessionjobs
  - flinksessionjobs/status
  verbs:
  - '*'
- apiGroups:
  - networking.k8s.io
  resources:
  - ingresses
  verbs:
  - '*'
- apiGroups:
  - coordination.k8s.io
  resources:
  - leases
  verbs:
  - '*'
```

Rôle de tâche

JobManager Utilise le rôle de travail pour créer et gérer TaskManagers et ConfigMaps pour chaque tâche.

```
rules:
- apiGroups:
  - ""
  resources:
  - pods
  - configmaps
  verbs:
  - '*'
- apiGroups:
  - apps
  resources:
  - deployments
  - deployments/finalizers
  verbs:
  - '*'
```

Désinstallation de l'opérateur Kubernetes pour Flink sur Amazon EMR on EKS

Suivez ces étapes pour désinstaller l'opérateur Kubernetes pour Flink.

1. Supprimez l'opérateur.

```
helm uninstall flink-kubernetes-operator -n <NAMESPACE>
```

2. Supprimez les ressources Kubernetes que Helm ne désinstalle pas.

```
kubectl delete serviceaccounts, roles, rolebindings -l emr-
containers.amazonaws.com/component=flink.operator --namespace <namespace>
kubectl delete crd flinkdeployments.flink.apache.org
 flinksessionjobs.flink.apache.org
```

3. (Facultatif) Supprimez le gestionnaire de certificats.

```
kubectl delete -f https://github.com/jetstack/cert-manager/releases/download/
v1.12.0/cert-manager.yaml
```

Utilisation de Flink Native Kubernetes

Les versions 6.13.0 et supérieures d'Amazon EMR prennent en charge Flink Native Kubernetes en tant qu'outil de ligne de commande que vous pouvez utiliser pour soumettre et exécuter des applications Flink sur un cluster Amazon EMR on EKS.

Rubriques

- Configuration de Flink Native Kubernetes pour Amazon EMR on EKS
- Les premiers pas avec Flink Native Kubernetes sur Amazon EMR on EKS
- Exigences de sécurité du compte JobManager de service Flink pour Native Kubernetes

Configuration de Flink Native Kubernetes pour Amazon EMR on EKS

Effectuez les tâches suivantes pour vous préparer à exécuter une application en utilisant la CLI Flink sur Amazon EMR on EKS. Si vous êtes déjà inscrit à Amazon Web Services (AWS) et que vous avez

utilisé Amazon EKS, vous êtes presque prêt à utiliser Amazon EMR on EKS. Si vous avez déjà rempli l'une des conditions préalables, vous pouvez l'ignorer et passer à la suivante.

- Installation ou mise à jour vers la dernière version du AWS CLI — Si vous avez déjà installé le AWS CLI, vérifiez que vous disposez de la dernière version.
- Commencez avec Amazon EKS — eksctl — Suivez les étapes pour créer un nouveau cluster Kubernetes avec des nœuds dans Amazon EKS.
- Sélectionnez l'URI d'une image de base Amazon EMR (version 6.13.0 ou supérieure). La commande Kubernetes pour Flink est prise en charge par les versions 6.13.0 et supérieures d'Amazon EMR.
- Vérifiez que le compte JobManager de service dispose des autorisations appropriées pour créer et regarder TaskManager des pods. Pour plus d'informations, consultez les exigences de sécurité du compte JobManager de service Flink pour Native Kubernetes.
- Configurez votre profil d'informations d'identification AWS local.
- Créez ou mettez à jour un fichier kubeconfig pour le cluster Amazon EKS sur lequel vous souhaitez exécuter les applications Flink.

Les premiers pas avec Flink Native Kubernetes sur Amazon EMR on EKS

Ces étapes vous montrent comment configurer, configurer un compte de service et exécuter une application Flink. Flink Native Kubernetes est utilisé pour déployer Flink sur un cluster Kubernetes en cours d'exécution.

Configuration et exécution d'une application Flink

Amazon EMR en version 6.13.0 et supérieure prend en charge Flink Native Kubernetes pour l'exécution d'applications Flink sur un cluster Amazon EKS. Pour exécuter une application Spark, procédez comme suit :

1. Pour pouvoir exécuter une application Flink à l'aide de la commande Flink Native Kubernetes, suivez les étapes indiquées dans the section called “Configuration”.
2. Téléchargez et installez Flink.
3. Définissez les valeurs des variables d'environnement suivantes.

```
#Export the FLINK_HOME environment variable to your local installation of Flink
export FLINK_HOME=/usr/local/bin/flink #Will vary depending on your installation
export NAMESPACE=flink
```

```
export CLUSTER_ID=flink-application-cluster
export IMAGE=<123456789012.dkr.ecr.sample-Région AWS-.amazonaws.com/flink/
emr-6.13.0-flink:latest>
export FLINK_SERVICE_ACCOUNT=emr-containers-sa-flink
export FLINK_CLUSTER_ROLE_BINDING=emr-containers-crb-flink
```

4. Créez un compte de service pour gérer les ressources Kubernetes.

```
kubectl create serviceaccount $FLINK_SERVICE_ACCOUNT -n $NAMESPACE
kubectl create clusterrolebinding $FLINK_CLUSTER_ROLE_BINDING --clusterrole=edit --
serviceaccount=$NAMESPACE:$FLINK_SERVICE_ACCOUNT
```

5. Exécutez la commande CLI `run-application`.

```
$FLINK_HOME/bin/flink run-application \
    --target kubernetes-application \
    -Dkubernetes.namespace=$NAMESPACE \
    -Dkubernetes.cluster-id=$CLUSTER_ID \
    -Dkubernetes.container.image.ref=$IMAGE \
    -Dkubernetes.service-account=$FLINK_SERVICE_ACCOUNT \
    local:///opt/flink/examples/streaming/Iteration.jar
2022-12-29 21:13:06,947 INFO  org.apache.flink.kubernetes.utils.KubernetesUtils
            [] - Kubernetes deployment requires a fixed port. Configuration
 blob.server.port will be set to 6124
2022-12-29 21:13:06,948 INFO  org.apache.flink.kubernetes.utils.KubernetesUtils
            [] - Kubernetes deployment requires a fixed port. Configuration
 taskmanager.rpc.port will be set to 6122
2022-12-29 21:13:07,861 WARN
 org.apache.flink.kubernetes.KubernetesClusterDescriptor      [] - Please note that
 Flink client operations(e.g. cancel, list, stop, savepoint, etc.) won't work from
 outside the Kubernetes cluster since 'kubernetes.rest-service.exposed.type' has
 been set to ClusterIP.
2022-12-29 21:13:07,868 INFO
 org.apache.flink.kubernetes.KubernetesClusterDescriptor      [] - Create flink
 application cluster flink-application-cluster successfully, JobManager Web
 Interface: http://flink-application-cluster-rest.flink:8081
```

6. Examinez les ressources Kubernetes créées.

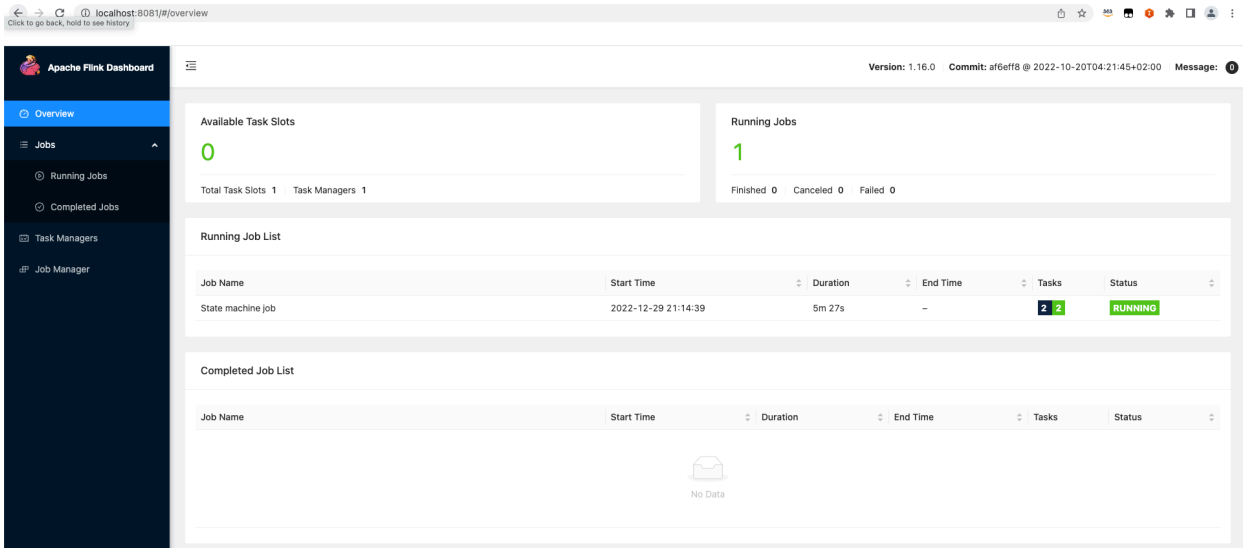
```
kubectl get all -n <namespace>
NAME READY STATUS RESTARTS AGE
pod/flink-application-cluster-546687cb47-w2p2z 1/1 Running 0 3m37s
```

```
pod/flink-application-cluster-taskmanager-1-1 1/1 Running 0 3m24s

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE
service/flink-application-cluster ClusterIP None <none> 6123/TCP,6124/TCP 3m38s
service/flink-application-cluster-rest ClusterIP 10.100.132.158 <none> 8081/TCP
 3m38s

NAME READY UP-TO-DATE AVAILABLE AGE
deployment.apps/flink-application-cluster 1/1 1 1 3m38s

NAME DESIRED CURRENT READY AGE
replicaset.apps/flink-application-cluster-546687cb47 1 1 1 3m38s
```

7. Transfert de port vers 8081.

```
kubectl port-forward service/flink-application-cluster-rest 8081 -n <namespace>
Forwarding from 127.0.0.1:8081 -> 8081
```

8. Accédez localement à l'interface utilisateur de Flink.



9. Supprimez l'application Flink.

```
kubectl delete deployment.apps/flink-application-cluster -n <namespace>
deployment.apps "flink-application-cluster" deleted
```

Pour plus d'informations sur la soumission d'applications à Flink, consultez Native Kubernetes dans la documentation d'Apache Flink.

Exigences de sécurité du compte JobManager de service Flink pour Native Kubernetes

Le JobManager module Flink utilise un compte de service Kubernetes pour accéder au serveur d'API Kubernetes afin de créer et de regarder des pods. TaskManager Le compte de JobManager service doit disposer des autorisations appropriées sur les create/delete TaskManager pods et TaskManager autoriser le responsable de surveillance ConfigMaps à récupérer l'adresse de JobManager et ResourceManager dans votre cluster.

Les règles suivantes s'appliquent à ce compte de service.

```
rules:
- apiGroups:
  - ""
  resources:
  - pods
  verbs:
  - "*"
- apiGroups:
  - ""
  resources:
  - services
  verbs:
  - "*"
- apiGroups:
  - ""
  resources:
  - configmaps
  verbs:
  - "*"
- apiGroups:
  - "apps"
  resources:
  - deployments
  verbs:
  - "*"
```

Personnalisation des images Docker pour Flink et FluentD

Procédez comme suit pour personnaliser les images Docker pour Amazon EMR sur EKS avec des images Apache Flink ou FluentD. Il s'agit notamment de conseils techniques pour obtenir une image de base, la personnaliser, la publier et soumettre une charge de travail.

Rubriques

Conditions préalables

Avant de personnaliser votre image Docker, assurez-vous de remplir les conditions préalables suivantes :

- Vous avez terminé les étapes de configuration de l'opérateur Flink Kubernetes pour Amazon EMR sur EKS.
- Docker installé dans votre environnement. Pour plus d'informations, consultez Obtenir Docker.

Étape 1 : récupérer une image de base depuis Amazon Elastic Container Registry

L'image de base contient le moteur d'exécution Amazon EMR et les connecteurs dont vous avez besoin pour accéder à d'autres. Services AWS Si vous utilisez Amazon EMR sur EKS avec la version 6.14.0 ou supérieure de Flink, vous pouvez obtenir les images de base depuis la galerie publique Amazon ECR. Parcourez la galerie pour trouver le lien de l'image et extrayez l'image dans votre espace de travail local. Par exemple, pour la version 6.14.0 d'Amazon EMR, la `docker pull` commande suivante renvoie la dernière image de base standard. `emr-6.14.0:latest`Remplacez-le par la version finale que vous souhaitez.

```
docker pull public.ecr.aws/emr-on-eks/flink/emr-6.14.0-flink:latest
```

Voici les liens vers l'image de la galerie Flink et l'image de la galerie Fluentd :

- emr-on-eks/flink/emr-6.14.0-flink
- emr-on-eks/fluentd/emr-6.14.0 (

Étape 2 : Personnaliser une image de base

Les étapes suivantes décrivent comment personnaliser l'image de base que vous avez extraite d'Amazon ECR.

1. Créez un nouveau espace de travail `Dockerfile` sur votre espace de travail local.
2. Modifiez le contenu `Dockerfile` et ajoutez-y le contenu suivant. Cela `Dockerfile` utilise l'image du conteneur que vous avez extraite`public.ecr.aws/emr-on-eks/flink/emr-7.12.0-flink:latest`.

   ```
   FROM public.ecr.aws/emr-on-eks/flink/emr-7.12.0-flink:latest
   USER root
   ### Add customization commands here ####
   USER hadoop:hadoop
   ```

   Utilisez la configuration suivante si vous utilisez`Fluentd`.

   ```
   FROM public.ecr.aws/emr-on-eks/fluentd/emr-7.12.0:latest
   USER root
   ### Add customization commands here ####
   USER hadoop:hadoop
   ```

3. Ajoutez des commandes dans le `Dockerfile` pour personnaliser l'image de base. La commande suivante montre comment installer des bibliothèques Python.

   ```
   FROM public.ecr.aws/emr-on-eks/flink/emr-7.12.0-flink:latest
   USER root
   RUN pip3 install --upgrade boto3 pandas numpy // For python 3
   USER hadoop:hadoop
   ```

4. Dans le répertoire où vous l'avez créée`DockerFile`, exécutez la commande suivante pour créer l'image Docker. Le champ que vous fournissez après le `-t` drapeau est le nom personnalisé de l'image.

```
docker build -t <YOUR_ACCOUNT_ID>.dkr.ecr.<YOUR_ECR_REGION>.amazonaws.com/
<ECR_REPO>:<ECR_TAG>
```

Étape 3 : Publiez votre image personnalisée

Vous pouvez désormais publier la nouvelle image Docker dans votre registre Amazon ECR.

1. Exécutez la commande suivante pour créer un référentiel Amazon ECR afin de stocker votre image Docker. Donnez un nom à votre référentiel, par exemple `emr_custom_repo`. Pour plus d'informations, consultez Créer un référentiel dans le guide de l'utilisateur d'Amazon Elastic Container Registry.

   ```
   aws ecr create-repository \
          --repository-name emr_custom_repo \
          --image-scanning-configuration scanOnPush=true \
          --region <AWS_REGION>
   ```

2. Exécutez la commande suivante pour vous authentifier dans votre registre par défaut. Pour plus d'informations, consultez Authentifier auprès de votre registre par défaut dans le guide de l'utilisateur d'Amazon Elastic Container Registry.

   ```
   aws ecr get-login-password --region <AWS_REGION> | docker login --username AWS --
   password-stdin <AWS_ACCOUNT_ID>.dkr.ecr.<YOUR_ECR_REGION>.amazonaws.com
   ```

3. Transmettez l'image. Pour plus d'informations, consultez la section Envoyer une image vers Amazon ECR dans le guide de l'utilisateur d'Amazon Elastic Container Registry.

   ```
   docker push <YOUR_ACCOUNT_ID>.dkr.ecr.<YOUR_ECR_REGION>.amazonaws.com/
   <ECR_REPO>:<ECR_TAG>
   ```

Étape 4 : Soumettre une charge de travail Flink dans Amazon EMR à l'aide d'une image personnalisée

Apportez les modifications suivantes à vos `FlinkDeployment` spécifications pour utiliser une image personnalisée. Pour ce faire, entrez votre propre image dans la `spec.image` ligne de votre spécification de déploiement.

```
apiVersion: flink.apache.org/v1beta1
   kind: FlinkDeployment
   metadata:
     name: basic-example
   spec:
     flinkVersion: v1_18
     image: <YOUR_ACCOUNT_ID>.dkr.ecr.<YOUR_ECR_REGION>.amazonaws.com/
<ECR_REPO>:<ECR_TAG>
     imagePullPolicy: Always
     flinkConfiguration:
           taskmanager.numberOfTaskSlots: "1"
```

Pour utiliser une image personnalisée pour votre tâche Fluentd, entrez votre propre image dans la `monitoringConfiguration.image` ligne de votre spécification de déploiement.

```
  monitoringConfiguration:
       image: <YOUR_ACCOUNT_ID>.dkr.ecr.<YOUR_ECR_REGION>.amazonaws.com/
<ECR_REPO>:<ECR_TAG>
       cloudWatchMonitoringConfiguration:
         logGroupName: flink-log-group
         logStreamNamePrefix: custom-fluentd
```

Surveillance de l'opérateur Kubernetes pour Flink et des tâche Flink

Cette section décrit plusieurs manières de surveiller vos tâches Flink à l'aide d'Amazon EMR on EKS. Il s'agit notamment de l'intégration de Flink au service géré Amazon pour Prometheus, de l'utilisation du tableau de bord Web Flink, qui fournit l'état des tâches et des métriques, ou de l'utilisation d'une configuration de surveillance pour envoyer les données de journal à Amazon S3 et. Amazon CloudWatch

Rubriques

- Utilisez Amazon Managed Service pour Prometheus pour surveiller les jobs Flink
- Utiliser l'interface utilisateur de Flink pour surveiller les tâches Flink
- Utiliser la configuration de surveillance pour surveiller l'opérateur Flink Kubernetes et les tâches Flink

Utilisez Amazon Managed Service pour Prometheus pour surveiller les jobs Flink

Vous pouvez intégrer Apache Flink à Amazon Managed Service for Prometheus (portail de gestion). Amazon Managed Service for Prometheus prend en charge l'ingestion de métriques à partir de serveurs Amazon Managed Service for Prometheus dans des clusters exécutés sur Amazon EKS. Amazon Managed Service for Prometheus fonctionne avec un serveur Prometheus déjà en cours d'exécution sur votre cluster Amazon EKS. L'exécution de l'intégration d'Amazon Managed Service for Prometheus à l'opérateur Flink pour Amazon EMR déploiera et configurera automatiquement un serveur Prometheus pour l'intégrer à Amazon Managed Service for Prometheus.

1. Créez un espace de travail Amazon Managed Service for Prometheus. Cet espace de travail sert de point de terminaison pour l'ingestion. Vous aurez besoin de l'URL d'écriture à distance plus tard.
2. Configurez les rôles IAM pour les comptes de service.

   Pour cette méthode d'intégration, utilisez des rôles IAM pour les comptes de service du cluster Amazon EKS où le serveur Prometheus est exécuté. Ces rôles sont également appelés fonctions du service.

   Si vous ne disposez pas encore de rôles, configurez des rôles de service pour l'ingestion de métriques à partir des clusters Amazon EKS.

   Avant de continuer, créez un rôle IAM appelé `amp-iamproxy-ingest-role`.
3. Installez l'opérateur Flink pour Amazon EMR avec Amazon Managed Service for Prometheus.

Maintenant que vous disposez d'un espace de travail Amazon Managed Service for Prometheus, d'un rôle IAM dédié à Amazon Managed Service for Prometheus et des autorisations nécessaires, vous pouvez installer l'opérateur Flink pour Amazon EMR.

Créez un fichier `enable-amp.yaml`. Ce fichier vous permet d'utiliser une configuration personnalisée pour remplacer les paramètres d'Amazon Managed Service for Prometheus. Assurez-vous d'utiliser vos propres rôles.

```
kube-prometheus-stack:
    prometheus:
    serviceAccount:
        create: true
```

```
        name: "amp-iamproxy-ingest-service-account"
        annotations:
            eks.amazonaws.com/role-arn: "arn:aws:iam::<AWS_ACCOUNT_ID>:role/amp-
iamproxy-ingest-role"
    remoteWrite:
        - url: <AMAZON_MANAGED_PROMETHEUS_REMOTE_WRITE_URL>
        sigv4:
            region: <AWS_REGION>
        queueConfig:
            maxSamplesPerSend: 1000
            maxShards: 200
            capacity: 2500
```

Utilisez la commande Helm Install --set pour transmettre les remplacements au chart `flink-kubernetes-operator`.

```
helm upgrade -n <namespace> flink-kubernetes-operator \
   oci://public.ecr.aws/emr-on-eks/flink-kubernetes-operator \
   --set prometheus.enabled=true
   -f enable-amp.yaml
```

Cette commande installe automatiquement un reporter Prometheus dans l'opérateur sur le port 9999. Tout `FlinkDeployment` futur expose également un port `metrics` sur 9249.

- Les métriques de l'opérateur Flink apparaissent dans Prometheus sous l'étiquette `flink_k8soperator_`.
- Les métriques du gestionnaire de tâches Flink apparaissent dans Prometheus sous l'étiquette `flink_taskmanager_`.
- Les métriques du gestionnaire de tâches Flink apparaissent dans Prometheus sous l'étiquette `flink_jobmanager_`.

Utiliser l'interface utilisateur de Flink pour surveiller les tâches Flink

Pour surveiller l'état et les performances d'une application Flink en cours d'exécution, utilisez le tableau de bord web de Flink. Ce tableau de bord fournit des informations sur le statut de la tâche, le nombre de tâches TaskManagers, les métriques et les journaux associés à la tâche. Il vous permet également de consulter et de modifier la configuration de la tâche Flink, et d'interagir avec le cluster Flink pour soumettre ou annuler des tâches.

Pour accéder au tableau de bord Web de Flink pour une application Flink en cours d'exécution sur Kubernetes :

1. Utilisez la `kubectl port-forward` commande pour transférer un port local vers le port sur lequel le tableau de bord Web Flink s'exécute dans les modules de TaskManager l'application Flink. Par défaut, ce port est 8081. Remplacez *`deployment-name`* par le nom du déploiement de l'application Flink indiqué ci-dessus.

   ```
   kubectl get deployments -n namespace
   ```

   Exemple de sortie :

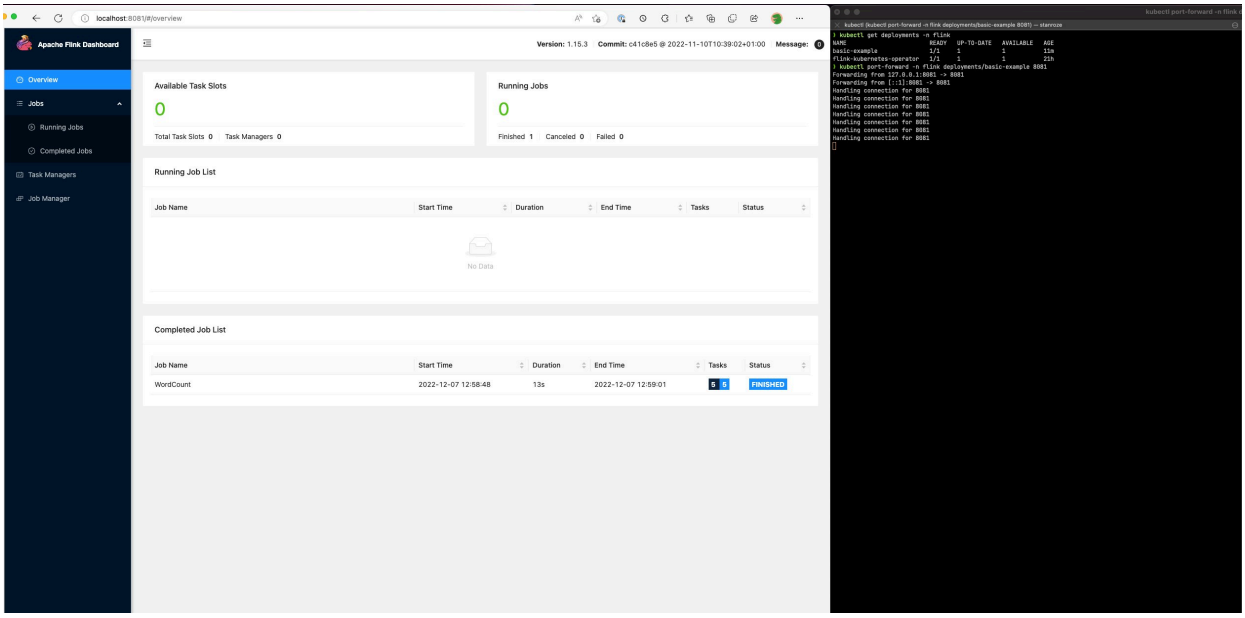
   ```
   kubectl get deployments -n flink-namespace
   NAME                        READY   UP-TO-DATE   AVAILABLE   AGE
   basic-example               1/1       1            1           11m
   flink-kubernetes-operator   1/1       1            1           21h
   ```

   ```
   kubectl port-forward deployments/deployment-name 8081 -n namespace
   ```

2. Si vous souhaitez utiliser un autre port localement, utilisez le paramètre:8081*`local-port`*.

   ```
   kubectl port-forward -n flink deployments/basic-example 8080:8081
   ```

3. Dans un navigateur web, naviguez vers `http://localhost:8081` (ou `http://localhost:`*`local-port`* si vous avez utilisé un port local personnalisé) pour accéder au tableau de bord web de Flink. Ce tableau de bord affiche des informations sur l'application Flink en cours d'exécution, telles que le statut de la tâche, le nombre de tâches TaskManagers, ainsi que les métriques et les journaux associés à la tâche.



Utiliser la configuration de surveillance pour surveiller l'opérateur Flink Kubernetes et les tâches Flink

La configuration de surveillance vous permet de configurer facilement l'archivage des journaux de votre application Flink et des journaux des opérateurs dans S3 and/or CloudWatch (vous pouvez choisir l'un ou les deux). Cela ajoute un sidecar FluentD à vos pods TaskManager et transmet ensuite les JobManager journaux de ces composants à vos récepteurs configurés.

> **Note**
>
> Vous devez configurer les rôles IAM pour le compte de service de votre opérateur Flink et de votre tâche Flink (comptes de service) afin de pouvoir utiliser cette fonctionnalité, car elle nécessite des interactions avec d'autres Services AWS. Vous devez le configurer en utilisant IRSA dans Configuration de l'opérateur Kubernetes pour Flink sur Amazon EMR on EKS.

Journaux des applications Flink

Vous pouvez définir cette configuration de la manière suivante.

```
apiVersion: flink.apache.org/v1beta1
kind: FlinkDeployment
metadata:
```

```
  name: basic-example
spec:
  image: FLINK IMAGE TAG
  imagePullPolicy: Always
  flinkVersion: v1_17
  flinkConfiguration:
    taskmanager.numberOfTaskSlots: "2"
  executionRoleArn: JOB EXECUTION ROLE
  jobManager:
    resource:
      memory: "2048m"
      cpu: 1
  taskManager:
    resource:
      memory: "2048m"
      cpu: 1
  job:
    jarURI: local:///opt/flink/examples/streaming/StateMachineExample.jar
  monitoringConfiguration:
    s3MonitoringConfiguration:
      logUri: S3 BUCKET
    cloudWatchMonitoringConfiguration:
      logGroupName: LOG GROUP NAME
      logStreamNamePrefix: LOG GROUP STREAM PREFIX
    sideCarResources:
      limits:
        cpuLimit: 500m
        memoryLimit: 250Mi
    containerLogRotationConfiguration:
        rotationSize: 2GB
        maxFilesToKeep: 10
```

Les options de configuration sont les suivantes.

- `s3MonitoringConfiguration` : clé de configuration permettant de configurer le transfert vers S3
  - `logUri` (obligatoire) : le chemin du compartiment S3 dans lequel vous souhaitez stocker vos journaux.
  - Le chemin sur S3 une fois les journaux chargés ressemblera à ce qui suit.
    - Aucune rotation des journaux n'est activée :

```
s3://${logUri}/${POD NAME}/STDOUT or STDERR.gz
```

- La rotation des journaux est activée. Vous pouvez utiliser à la fois un fichier en rotation et un fichier actuel (sans horodatage).

```
s3://${logUri}/${POD NAME}/STDOUT or STDERR.gz
```

Le format suivant est un nombre incrémentiel.

```
s3://${logUri}/${POD NAME}/stdout_YYYYMMDD_index.gz
```

- Les autorisations IAM suivantes sont nécessaires pour utiliser ce transféreur.

```
{
    "Effect": "Allow",
    "Action": [
        "s3:PutObject"
    ],
    "Resource": [
       "S3_BUCKET_URI/*",
       "S3_BUCKET_URI"
    ]
}
```

- `cloudWatchMonitoringConfiguration`— clé de configuration vers laquelle configurer le transfert CloudWatch.
  - `logGroupName`(obligatoire) : nom du groupe de CloudWatch journaux auquel vous souhaitez envoyer des journaux (crée automatiquement le groupe s'il n'existe pas).
  - `logStreamNamePrefix` (facultatif) : nom du flux de journaux auquel vous souhaitez envoyer des journaux. La valeur par défaut est une chaîne vide. Le format est le suivant :

```
${logStreamNamePrefix}/${POD NAME}/STDOUT or STDERR
```

  - Les autorisations IAM suivantes sont nécessaires pour utiliser ce transféreur.

```
{
    "Effect": "Allow",
    "Action": [
        "logs:CreateLogStream",
```

```
            "logs:CreateLogGroup",
            "logs:PutLogEvents"
        ],
        "Resource": [
            "arn:aws:logs:REGION:ACCOUNT-ID:log-group:{YOUR_LOG_GROUP_NAME}:*",
            "arn:aws:logs:REGION:ACCOUNT-ID:log-group:{YOUR_LOG_GROUP_NAME}"
        ]
    }
```

- `sideCarResources` (facultatif) : clé de configuration permettant de définir les limites de ressources sur le conteneur sidecar Fluentbit lancé.
  - `memoryLimit` (facultatif) : la valeur par défaut est 512Mi. Ajustez selon vos besoins.
  - `cpuLimit` (facultatif) : cette option n'a pas de valeur par défaut. Ajustez selon vos besoins.
- `containerLogRotationConfiguration` (facultatif) : contrôle le comportement de rotation des journaux du conteneur. Il est activé par défaut.
  - `rotationSize` (obligatoire) : indique la taille du fichier pour la rotation des journaux. La plage de valeurs possibles est comprise entre 2 Ko et 2 Go. La partie unitaire numérique du paramètre rotationSize est transmise sous forme d'entier. Les valeurs décimales n'étant pas prises en charge, vous pouvez indiquer une taille de rotation de 1,5 Go, par exemple, avec la valeur 1500 Mo. La valeur par défaut est 2 Go.
  - `maxFilesToKeep` (obligatoire) : indique le nombre maximum de fichiers à retenir dans le conteneur après la rotation. La valeur minimale est 1 et la valeur maximale est 50. La valeur par défaut est 10.

Journaux de l'opérateur Flink

Il est également possible d'activer l'archivage des journaux de l'opérateur en utilisant les options ci-dessous dans le fichier `values.yaml` de l'installation de vos Charts de Helm. Vous pouvez activer S3 ou CloudWatch les deux.

```
monitoringConfiguration:
  s3MonitoringConfiguration:
    logUri: "S3-BUCKET"
    totalFileSize: "1G"
    uploadTimeout: "1m"
  cloudWatchMonitoringConfiguration:
    logGroupName: "flink-log-group"
    logStreamNamePrefix: "example-job-prefix-test-2"
  sideCarResources:
```

```
    limits:
      cpuLimit: 1
      memoryLimit: 800Mi
  memoryBufferLimit: 700M
```

Les options de configuration disponibles sous `monitoringConfiguration` sont les suivantes.

- `s3MonitoringConfiguration` : définissez cette option pour archiver dans S3.
- `logUri` (obligatoire) : le chemin du compartiment S3 dans lequel vous souhaitez stocker vos journaux.
- Les formats suivants indiquent à quoi peuvent ressembler les chemins des compartiments S3 une fois les journaux chargés.
  - Aucune rotation des journaux n'est activée.

    ```
    s3://${logUri}/${POD NAME}/OPERATOR or WEBHOOK/STDOUT or STDERR.gz
    ```

  - La rotation des journaux est activée. Vous pouvez utiliser à la fois un fichier en rotation et un fichier actuel (sans horodatage).

    ```
    s3://${logUri}/${POD NAME}/OPERATOR or WEBHOOK/STDOUT or STDERR.gz
    ```

    L'index du format suivant est un nombre incrémentiel.

    ```
    s3://${logUri}/${POD NAME}/OPERATOR or WEBHOOK/stdout_YYYYMMDD_index.gz
    ```

- `cloudWatchMonitoringConfiguration`— la clé de configuration vers laquelle configurer le transfert CloudWatch.
  - `logGroupName`(obligatoire) : nom du groupe de CloudWatch journaux auquel vous souhaitez envoyer des journaux. Le groupe est automatiquement créé s'il n'existe pas.
  - `logStreamNamePrefix` (facultatif) : nom du flux de journaux auquel vous souhaitez envoyer des journaux. La valeur par défaut est une chaîne vide. Le format CloudWatch est le suivant :

    ```
    ${logStreamNamePrefix}/${POD NAME}/STDOUT or STDERR
    ```

- `sideCarResources` (facultatif) : clé de configuration permettant de définir les limites de ressources sur le conteneur sidecar Fluentbit lancé.
  - `memoryLimit` (facultatif) : limite de mémoire. Ajustez selon vos besoins. La valeur par défaut est 512Mi.

  - `cpuLimit` : la limite de CPU. Ajustez selon vos besoins. Aucune valeur par défaut.
- `containerLogRotationConfiguration` (facultatif) : contrôle le comportement de rotation des journaux du conteneur. Il est activé par défaut.
  - `rotationSize` (obligatoire) : indique la taille du fichier pour la rotation des journaux. La plage de valeurs possibles est comprise entre 2 Ko et 2 Go. La partie unitaire numérique du paramètre rotationSize est transmise sous forme d'entier. Les valeurs décimales n'étant pas prises en charge, vous pouvez indiquer une taille de rotation de 1,5 Go, par exemple, avec la valeur 1500 Mo. La valeur par défaut est 2 Go.
  - `maxFilesToKeep` (obligatoire) : indique le nombre maximum de fichiers à retenir dans le conteneur après la rotation. La valeur minimale est 1 et la valeur maximale est 50. La valeur par défaut est 10.

Comment Flink favorise la haute disponibilité et la résilience au travail

Les sections suivantes expliquent comment Flink améliore la fiabilité et la disponibilité des offres d'emploi. Pour ce faire, il utilise des fonctionnalités intégrées telles que la haute disponibilité de Flink et diverses fonctionnalités de restauration en cas de panne.

Rubriques

- Utilisation de la haute disponibilité (HA) pour les opérateurs et les applications Flink
- Optimisation des temps de redémarrage des tâches Flink pour la récupération des tâches et la mise à l'échelle des opérations avec Amazon EMR sur EKS
- Mise hors service progressive des instances Spot avec Flink sur Amazon EMR sur EKS

Utilisation de la haute disponibilité (HA) pour les opérateurs et les applications Flink

Cette rubrique explique comment configurer la haute disponibilité et décrit son fonctionnement pour différents cas d'utilisation. Cela inclut lorsque vous utilisez le Job Manager et lorsque vous utilisez les kubernetes natifs de Flink.

Haute disponibilité de l’opérateur Flink

Nous activons la haute disponibilité de l’opérateur Flink afin de pouvoir basculer vers un opérateur Flink de secours et réduire au minimum les temps d’arrêt dans la boucle de contrôle de l’opérateur en cas de défaillance. La haute disponibilité est activée par défaut et le nombre initial par défaut de répliques d’opérateur est de 2. Vous pouvez configurer le champ des répliques dans votre fichier `values.yaml` pour les Charts de Helm.

Les champs suivants sont personnalisables :

- `replicas` (facultatif, la valeur par défaut est 2) : si vous définissez ce nombre sur une valeur supérieure à 1, d'autres opérateurs de secours seront créés et vous pourrez reprendre votre tâche plus rapidement.
- `highAvailabilityEnabled` (facultatif, la valeur par défaut est « true ») : permet d'indiquer si vous souhaitez activer la haute disponibilité (HA). Le fait de définir ce paramètre comme « true » permet de prendre en charge le déploiement multi-AZ et de définir les paramètres `flink-conf.yaml` corrects.

Vous pouvez désactiver la haute disponibilité pour votre opérateur en paramétrant la configuration ci-dessous dans votre fichier `values.yaml`.

```
...
imagePullSecrets: []

replicas: 1

# set this to false if you don't want HA
highAvailabilityEnabled: false
...
```

Déploiement multi-AZ

Nous créons les pods des opérateurs dans plusieurs zones de disponibilité. Il s'agit d'une contrainte souple, et vos pods d'opérateur seront planifiés dans la même zone si vous ne disposez pas de suffisamment de ressources dans une autre zone.

Détermination de la réplique leader

Si HA est activé, les répliques utilisent un bail pour déterminer lequel des deux JMs est le leader et utilisent un bail K8s pour l'élection du leader. Vous pouvez décrire le bail et consulter le champ .Spec.Holder Identity pour déterminer le leader actuel

```
kubectl describe lease <Helm Install Release Name>-<NAMESPACE>-lease -n <NAMESPACE> |
 grep "Holder Identity"
```

Interaction Flink-S3

Configuration des informations d'identification d'accès

Assurez-vous d'avoir configuré IRSA avec les autorisations IAM appropriées pour accéder au compartiment S3.

Récupération des fichiers JAR des tâches à partir du mode d'application S3

L'opérateur Flink prend également en charge la récupération des fichiers JAR des applications à partir de S3. Il vous suffit de fournir l'emplacement S3 du JArUri dans votre FlinkDeployment spécification.

Vous pouvez également utiliser cette fonctionnalité pour télécharger d'autres artefacts tels que PyFlink des scripts. Le script Python résultant est déposé sous le chemin `/opt/flink/usrlib/`.

L'exemple suivant montre comment utiliser cette fonctionnalité pour une PyFlink tâche. Notez les champs jarURI et args.

```
apiVersion: flink.apache.org/v1beta1
kind: FlinkDeployment
metadata:
  name: python-example
spec:
  image: <YOUR CUSTOM PYFLINK IMAGE>
  emrReleaseLabel: "emr-6.12.0-flink-latest"
  flinkVersion: v1_16
  flinkConfiguration:
    taskmanager.numberOfTaskSlots: "1"
  serviceAccount: flink
  jobManager:
    highAvailabilityEnabled: false
    replicas: 1
    resource:
      memory: "2048m"
```

```
      cpu: 1
  taskManager:
    resource:
      memory: "2048m"
      cpu: 1
  job:
    jarURI: "s3://<S3-BUCKET>/scripts/pyflink.py" # Note, this will trigger the
 artifact download process
    entryClass: "org.apache.flink.client.python.PythonDriver"
    args: ["-pyclientexec", "/usr/local/bin/python3", "-py", "/opt/flink/usrlib/
pyflink.py"]
    parallelism: 1
    upgradeMode: stateless
```

Connecteurs S3 pour Flink

Flink est livré avec deux connecteurs S3 (indiqués ci-dessous). Les sections suivantes expliquent quand utiliser quel connecteur.

Point de contrôle : connecteur S3 pour Presto

- Définissez le schéma S3 sur s3p://
- Le connecteur recommandé à utiliser pour le point de contrôle vers S3. Pour plus d'informations, consultez la section spécifique à S3 dans la documentation d'Apache Flink.

Exemple de FlinkDeployment spécification :

```
apiVersion: flink.apache.org/v1beta1
kind: FlinkDeployment
metadata:
  name: basic-example
spec:
  flinkConfiguration:
    taskmanager.numberOfTaskSlots: "2"
    state.checkpoints.dir: s3p://<BUCKET-NAME>/flink-checkpoint/
```

Lecture et écriture sur S3 : connecteur Hadoop S3

- Définissez le schéma S3 sur `s3://` ou (`s3a://`)
- Le connecteur recommandé pour lire et écrire des fichiers à partir de S3 (uniquement le connecteur S3 qui implémente l'interface Filesystem de Flinks).

- Par défaut, nous avons défini `fs.s3a.aws.credentials.provider` le `flink-conf.yaml` fichier, qui est`com.amazonaws.auth.WebIdentityTokenCredentialsProvider`. Si vous remplacez complètement la `flink-conf` par défaut et que vous interagissez avec S3, assurez-vous d'utiliser ce fournisseur.

Exemple de FlinkDeployment spécification

```
apiVersion: flink.apache.org/v1beta1
kind: FlinkDeployment
metadata:
  name: basic-example
spec:
  job:
    jarURI: local:///opt/flink/examples/streaming/WordCount.jar
    args: [ "--input", "s3a://<INPUT BUCKET>/PATH", "--output", "s3a://<OUTPUT BUCKET>/
PATH" ]
    parallelism: 2
    upgradeMode: stateless
```

Gestionnaire de tâches Flink

La haute disponibilité (HA) pour les déploiements de Flink permet aux tâches de continuer à progresser même en cas d'erreur transitoire et de panne. JobManager Les tâches redémarreront, mais à partir du dernier point de contrôle validé lorsque la haute disponibilité est activée. Si la haute disponibilité n'est pas activée, Kubernetes redémarrera votre tâche JobManager, mais votre tâche redémarrera comme une nouvelle tâche et perdra sa progression. Après avoir configuré HA, nous pouvons demander à Kubernetes de stocker les métadonnées HA dans un stockage persistant afin de les référencer en cas de défaillance transitoire du, JobManager puis de reprendre nos tâches à partir du dernier point de contrôle réussi.

La haute disponibilité est activée par défaut pour vos tâches Flink (le nombre de répliques est défini sur 2, ce qui nécessite la mise à disposition d'un emplacement de stockage S3 pour l'enregistrement des métadonnées haute disponibilité).

Configurations de haute disponibilité

```
apiVersion: flink.apache.org/v1beta1
kind: FlinkDeployment
metadata:
  name: basic-example
```

```
spec:
  flinkConfiguration:
    taskmanager.numberOfTaskSlots: "2"
  executionRoleArn: "<JOB EXECUTION ROLE ARN>"
  emrReleaseLabel: "emr-6.13.0-flink-latest"
  jobManager:
    resource:
      memory: "2048m"
      cpu: 1
    replicas: 2
    highAvailabilityEnabled: true
    storageDir: "s3://<S3 PERSISTENT STORAGE DIR>"
  taskManager:
    resource:
      memory: "2048m"
      cpu: 1
```

Voici les descriptions des configurations de haute disponibilité ci-dessus dans le gestionnaire de tâches (définies sous .spec.JobManager) :

- `highAvailabilityEnabled` (facultatif, la valeur par défaut est « true ») : définissez ce paramètre sur `false` si vous ne voulez pas activer la haute disponibilité et si vous ne souhaitez pas utiliser les configurations de haute disponibilité fournies. Vous pouvez toujours manipuler le champ « replicas » pour configurer manuellement la haute disponibilité.
- `replicas`(facultatif, la valeur par défaut est 2) : Si vous définissez ce nombre sur une valeur supérieure à 1, vous créez un autre mode de veille JobManagers et vous pouvez reprendre votre travail plus rapidement. Si vous désactivez la haute disponibilité, vous devez définir le nombre de répliques sur 1, sinon vous continuerez à recevoir des erreurs de validation (une seule réplique est prise en charge si la haute disponibilité n'est pas activée).
- `storageDir` (obligatoire) : étant donné que le nombre de répliques est égal à 2 par défaut, nous devons fournir un StorageDir persistant. Actuellement, ce champ n'accepte que les chemins S3 comme emplacement de stockage.

Placement des pods

Si vous activez la haute disponibilité, nous essayons également de colocaliser les pods dans la même zone, ce qui améliore les performances (réduction de la latence du réseau grâce à la présence de pods dans les mêmes zones AZs). Ceci est un processus réalisé au mieux des possibilités, signifiant que si vous ne disposez pas de ressources suffisantes dans la zone de disponibilité où la

majorité de vos pods sont planifiés, les pods restants seront tout de même planifiés, mais pourraient se retrouver sur un nœud situé en dehors de cette zone de disponibilité.

Détermination de la réplique leader

Si HA est activé, les répliques utilisent un bail pour déterminer lequel d'entre eux JMs est le leader et utilisent un K8s Configmap comme banque de données pour stocker ces métadonnées. Si vous souhaitez identifier le leader, vous pouvez consulter le contenu de la Configmap et la clé `org.apache.flink.k8s.leader.restserver` sous les données pour trouver le pod K8s correspondant à l'adresse IP indiquée. Vous pouvez également utiliser les commandes bash ci-dessous.

```
ip=$(kubectl get configmap -n <NAMESPACE> <JOB-NAME>-cluster-config-map -o json | jq -
r ".data[\"org.apache.flink.k8s.leader.restserver\"]" | awk -F: '{print $2}' | awk -F
 '/' '{print $3}')
kubectl get pods -n NAMESPACE  -o json | jq -r ".items[] | select(.status.podIP ==
 \"$ip\") | .metadata.name"
```

Tâche Flink – Native Kubernetes

À partir de la version 6.13.0, Amazon EMR prend en charge Flink Native Kubernetes pour l'exécution d'applications Flink en mode haute disponibilité sur un cluster Amazon EKS.

> **Note**
>
> Il est nécessaire de disposer d'un compartiment Amazon S3 préalablement créé pour conserver les métadonnées de haute disponibilité lorsque vous soumettez votre tâche Flink. Si vous ne souhaitez pas utiliser cette fonctionnalité, vous pouvez la désactiver. Elle est activée par défaut.

Pour activer la fonctionnalité haute disponibilité de Flink, spécifiez les paramètres Flink suivants lorsque vous exécutez la commande `run-application` dans la CLI. Les paramètres sont définis dans l'exemple ci-dessous.

```
-Dhigh-availability.type=kubernetes \
-Dhigh-availability.storageDir=S3://DOC-EXAMPLE-STORAGE-BUCKET \
-
Dfs.s3a.aws.credentials.provider="com.amazonaws.auth.WebIdentityTokenCredentialsProvider"
 \
```

```
-Dkubernetes.jobmanager.replicas=3 \
-Dkubernetes.cluster-id=example-cluster
```

- **`Dhigh-availability.storageDir`** : compartiment Amazon S3 où vous souhaitez stocker les métadonnées de haute disponibilité pour votre tâche

  `Dkubernetes.jobmanager.replicas` : nombre de pods Job Manager à créer sous la forme d'un entier supérieur à 1

  `Dkubernetes.cluster-id` : ID unique qui identifie le cluster Flink

Optimisation des temps de redémarrage des tâches Flink pour la récupération des tâches et la mise à l'échelle des opérations avec Amazon EMR sur EKS

Lorsqu'une tâche échoue ou qu'une opération de mise à l'échelle a lieu, Flink tente de réexécuter la tâche à partir du dernier point de contrôle terminé. L'exécution du processus de redémarrage peut durer une minute ou plus, en fonction de la taille de l'état du point de contrôle et du nombre de tâches parallèles. Pendant la période de redémarrage, les tâches en attente peuvent s'accumuler pour la tâche. Flink peut cependant permettre d'optimiser la vitesse de récupération et de redémarrage des graphes d'exécution afin d'améliorer la stabilité des tâches.

Cette page décrit certaines des manières dont Amazon EMR Flink peut améliorer le temps de redémarrage des tâches lors de la reprise des tâches ou des opérations de dimensionnement sur des instances ponctuelles. Les instances Spot sont des capacités de calcul inutilisées disponibles à prix discount. Ses comportements sont uniques, notamment des interruptions occasionnelles. Il est donc important de comprendre comment Amazon EMR on EKS les gère, notamment comment Amazon EMR on EKS procède à la mise hors service et au redémarrage des tâches.

Rubriques

Récupération locale des tâches

> **Note**
>
> La récupération locale des tâches est prise en charge par Flink sur Amazon EMR sur EKS 6.14.0 et versions ultérieures.

Avec les points de contrôle Flink, chaque tâche produit un instantané de son état que Flink écrit sur un stockage distribué comme Amazon S3. En cas de récupération, les tâches restaurent leur état à partir du stockage distribué. Le stockage distribué offre une tolérance aux pannes et peut redistribuer l'état lors de la mise à l'échelle, car tous les nœuds peuvent y accéder.

Cependant, un magasin distribué à distance présente également un inconvénient : toutes les tâches doivent lire leur état depuis un emplacement distant sur le réseau, ce qui peut entraîner l'augmentation du temps de récupération pour les états importants lors des opérations de récupération ou de mise à l'échelle de tâches.

La récupération locale des tâches permet de résoudre ce problème. Les tâches enregistrent leur état au point de contrôle sur un stockage secondaire local à la tâche, par exemple sur un disque local. Elles stockent également leur état sur le stockage principal, à savoir Amazon S3 dans notre cas. Lors de la récupération, le planificateur planifie les tâches sur le même Task Manager que celui dans lequel les tâches ont été exécutées précédemment afin qu'elles puissent être récupérées depuis le magasin d'états local au lieu de lire depuis le magasin d'état distant. Pour plus d'informations, voir la rubrique Récupération locale des tâches de la documentation Apache Flink.

Nos tests d'évaluation avec des exemples de tâches ont montré que le temps de récupération était passé de quelques minutes à quelques secondes grâce à l'activation de la récupération locale des tâches.

Pour activer la récupération locale des tâches, définissez les configurations suivantes dans votre fichier `flink-conf.yaml`. Spécifiez la valeur de l'intervalle de point de contrôle en millisecondes.

```
state.backend.local-recovery: true
state.backend: hasmap or rocksdb
state.checkpoints.dir: s3://STORAGE-BUCKET-PATH/checkpoint
execution.checkpointing.interval: 15000
```

Récupération locale des tâches par montage de volume Amazon EBS

> **Note**
>
> La récupération locale des tâches par Amazon EBS est prise en charge par Flink sur Amazon EMR sur EKS 6.15.0 et versions ultérieures.

Avec Flink sur Amazon EMR sur EKS, vous pouvez automatiquement provisionner des volumes Amazon EBS sur les pods TaskManager pour la restauration locale des tâches. Le montage de superposition par défaut comprend un volume de 10 Go, ce qui est suffisant pour les tâches dont l'état est moins important. Les tâches dont les états sont importants peuvent activer l'option de montage automatique de volume EBS. Les pods TaskManager sont automatiquement créés et montés lors de la création du pod et supprimés lors de sa suppression.

Procédez comme suit pour activer le montage automatique de volume EBS pour Flink dans Amazon EMR sur EKS :

1. Exportez les valeurs des variables suivantes que vous utiliserez lors des étapes suivantes.

```
export AWS_REGION=aa-example-1
export FLINK_EKS_CLUSTER_NAME=my-cluster
export AWS_ACCOUNT_ID=111122223333
```

2. Créez ou mettez à jour un fichier YAML `kubeconfig` pour votre cluster.

```
aws eks update-kubeconfig --name $FLINK_EKS_CLUSTER_NAME --region $AWS_REGION
```

3. Créez un compte de service IAM pour le pilote CSI Amazon EBS sur votre cluster Amazon EKS.

```
eksctl create iamserviceaccount \
   --name ebs-csi-controller-sa \
   --namespace kube-system \
   --region $AWS_REGION \
   --cluster $FLINK_EKS_CLUSTER_NAME\
   --role-name TLR_${AWS_REGION}_${FLINK_EKS_CLUSTER_NAME} \
   --role-only \
   --attach-policy-arn arn:aws:iam::aws:policy/service-role/
AmazonEBSCSIDriverPolicy \
   --approve
```

4. Créez le pilote CSI Amazon EBS à l'aide de la commande suivante :

```
eksctl create addon \
   --name aws-ebs-csi-driver \
   --region $AWS_REGION \
   --cluster $FLINK_EKS_CLUSTER_NAME \
   --service-account-role-arn arn:aws:iam::${AWS_ACCOUNT_ID}:role/TLR_
${AWS_REGION}_${FLINK_EKS_CLUSTER_NAME}
```

5. Créez la classe de stockage Amazon EBS à l'aide de la commande suivante :

```
cat # EOF # storage-class.yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ebs-sc
provisioner: ebs.csi.aws.com
volumeBindingMode: WaitForFirstConsumer
EOF
```

   Appliquez ensuite la classe :

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

6. Helm installe l'opérateur Kubernetes pour Flink sur Amazon EMR avec des options permettant de créer un compte de service. Cela crée le `emr-containers-sa-flink` à utiliser dans le déploiement de Flink.

```
helm install flink-kubernetes-operator flink-kubernetes-operator/ \
   --set jobServiceAccount.create=true \
   --set rbac.jobRole.create=true \
   --set rbac.jobRoleBinding.create=true
```

7. Pour soumettre la tâche Flink et activer le provisionnement automatique des volumes EBS pour la récupération locale des tâches, définissez les configurations suivantes dans votre fichier `flink-conf.yaml`. Ajustez la limite de taille d'état pour la tâche. Définissez `serviceAccount` sur `emr-containers-sa-flink`. Spécifiez la valeur de l'intervalle de point de contrôle en millisecondes. Et omettez le `executionRoleArn`.

```
flinkConfiguration:
    task.local-recovery.ebs.enable: true
```

```
    kubernetes.taskmanager.local-recovery.persistentVolumeClaim.sizeLimit: 10Gi
    state.checkpoints.dir: s3://BUCKET-PATH/checkpoint
    state.backend.local-recovery: true
    state.backend: hasmap or rocksdb
    state.backend.incremental: "true"
    execution.checkpointing.interval: 15000
  serviceAccount: emr-containers-sa-flink
```

Lorsque vous êtes prêt à supprimer le plug-in de pilote CSI Amazon EBS, utilisez les commandes suivantes :

```
  # Detach Attached Policy
  aws iam detach-role-policy --role-name TLR_${$AWS_REGION}_${FLINK_EKS_CLUSTER_NAME}
 --policy-arn arn:aws:iam::aws:policy/service-role/AmazonEBSCSIDriverPolicy
  # Delete the created Role
  aws iam delete-role --role-name TLR_${$AWS_REGION}_${FLINK_EKS_CLUSTER_NAME}
  # Delete the created service account
  eksctl delete iamserviceaccount --name ebs-csi-controller-sa --namespace kube-system
 --cluster $FLINK_EKS_CLUSTER_NAME --region $AWS_REGION
  # Delete Addon
  eksctl delete addon --name aws-ebs-csi-driver --cluster $FLINK_EKS_CLUSTER_NAME --
region $AWS_REGION
  # Delete the EBS storage class
  kubectl delete -f storage-class.yaml
```

Point de contrôle incrémentiel générique basé sur les journaux

> **Note**
>
> Les points de contrôle incrémentiels génériques basés sur les journaux sont pris en charge avec Flink sur Amazon EMR sur EKS 6.14.0 et versions ultérieures.

Les points de contrôle incrémentiels génériques basés sur les journaux ont été ajoutés à Flink 1.16 pour rendre les points de contrôle plus fréquents. Un intervalle de point de contrôle plus court entraîne souvent une réduction du travail de récupération, car moins d'événements doivent être traités de nouveau après la récupération. Pour plus d'informations, accédez à la page Improving speed and stability of checkpointing with generic log-based incremental checkpoints sur le blog Apache Flink.

Sur base de quelques exemples de tâches, nos tests d'évaluation ont montré que le temps de contrôle était passé de quelques minutes à quelques secondes grâce au point de contrôle incrémentiel générique basé sur les journaux.

Pour activer les points de contrôle incrémentiels génériques basés sur les journaux, définissez les configurations suivantes dans votre fichier `flink-conf.yaml`. Spécifiez la valeur de l'intervalle de point de contrôle en millisecondes.

```
state.backend.changelog.enabled: true
state.backend.changelog.storage: filesystem
dstl.dfs.base-path: s3://bucket-path/changelog
state.backend.local-recovery: true
state.backend: rocksdb
state.checkpoints.dir: s3://bucket-path/checkpoint
execution.checkpointing.interval: 15000
```

Récupération précise

> **Note**
>
> La récupération précise du planificateur par défaut est prise en charge avec Flink sur Amazon EMR sur EKS 6.14.0 et versions ultérieures. La récupération précise du planificateur adaptatif est prise en charge avec Flink sur Amazon EMR sur EKS 6.15.0 et versions ultérieures.

Lorsqu'une tâche échoue pendant son exécution, Flink réinitialise l'intégralité du graphe d'exécution et déclenche une réexécution complète depuis le dernier point de contrôle terminé. Cette opération est plus chère qu'une simple réexécution des tâches qui ont échoué. La récupération précise redémarre uniquement le composant connecté au pipeline de la tâche ayant échoué. Dans l'exemple suivant, le graphe de tâches présente 5 sommets (`A` à `E`). Toutes les connexions entre les sommets sont en pipeline avec une distribution ponctuelle, et la valeur de `parallelism.default` pour la tâche est définie sur 2.

```
A # B # C # D # E
```

Dans cet exemple, 10 tâches sont en cours d'exécution au total. Le premier pipeline (`a1` à `e1`) s'exécute sur un TaskManager (`TM1`), et le deuxième pipeline (`a2` à `e2`) s'exécute sur un autre TaskManager (`TM2`).

```
a1 # b1 # c1 # d1 # e1
a2 # b2 # c2 # d2 # e2
```

Deux composants sont connectés en pipeline : `a1 # e1` et `a2 # e2`. En cas d'échec de `TM1` ou de `TM2`, l'échec affecte uniquement les 5 tâches du pipeline dans lequel TaskManager était en cours d'exécution. La stratégie de redémarrage démarre uniquement le composant en pipeline concerné.

La récupération précise ne fonctionne qu'avec des tâches Flink parfaitement parallèles. Elle n'est pas prise en charge par `keyBy()` ou par les opérations `redistribute()`. Pour plus d'informations, accédez à la page FLIP-1 : Fine Grained Recovery from Task Failures du projet Jira Flink Improvement Proposal.

Pour activer la récupération précise, définissez les configurations suivantes dans votre fichier `flink-conf.yaml`.

```
jobmanager.execution.failover-strategy: region
restart-strategy: exponential-delay or fixed-delay
```

Mécanisme de redémarrage combiné dans le planificateur adaptatif

> **Note**
>
> Le mécanisme de redémarrage combiné du planificateur adaptatif est pris en charge avec Flink sur Amazon EMR 6.15.0 et versions ultérieures.

Le planificateur adaptatif peut ajuster le parallélisme de la tâche en fonction des emplacements disponibles. Si le nombre d'emplacements disponibles est insuffisant, le planificateur réduit automatiquement le parallélisme pour s'adapter au parallélisme des tâches configuré. Si de nouveaux emplacements sont disponibles, la tâche fait l'objet d'une augmentation d'échelle selon le parallélisme des tâches configuré. Un planificateur adaptatif permet d'éviter les temps d'arrêt de la tâche lorsque les ressources disponibles sont insuffisantes. Il s'agit du planificateur pris en charge par l'outil de mise à l'échelle automatique Flink. Nous recommandons donc l'utilisation d'un planificateur adaptatif avec Amazon EMR Flink. Les planificateurs adaptatifs peuvent toutefois effectuer plusieurs redémarrages en peu de temps, à raison d'un redémarrage pour chaque nouvelle ressource ajoutée, ce qui peut entraîner une baisse des performances de la tâche.

Avec Amazon EMR 6.15.0 et versions ultérieures, Flink dispose d'un mécanisme de redémarrage combiné dans le planificateur adaptatif qui ouvre une fenêtre de redémarrage lorsque la première

ressource est ajoutée, puis attend l'intervalle de fenêtre configuré de 1 minute par défaut. Un seul redémarrage est effectué lorsque les ressources disponibles sont suffisantes pour exécuter la tâche avec un parallélisme configuré ou lorsque l'intervalle expire.

Grâce à quelques exemples de tâches, nos tests d'évaluation ont montré que cette fonctionnalité traite 10 % d'enregistrements supplémentaires par rapport au comportement par défaut lorsque vous utilisez le planificateur adaptatif et l'outil de mise à l'échelle automatique Flink.

Pour activer le mécanisme de redémarrage combiné, définissez les configurations suivantes dans votre fichier `flink-conf.yaml`.

```
jobmanager.adaptive-scheduler.combined-restart.enabled: true
jobmanager.adaptive-scheduler.combined-restart.window-interval: 1m
```

Mise hors service progressive des instances Spot avec Flink sur Amazon EMR sur EKS

L'utilisation de Flink avec Amazon EMR sur EKS peut améliorer le temps de redémarrage des tâches lors des opérations de récupération ou de mise à l'échelle des tâches.

Présentation de

La version 6.15.0 et les versions ultérieures d'Amazon EMR prennent en charge la mise hors service progressive des Task Managers sur les instances Spot dans Amazon EMR sur EKS avec Apache Flink. Dans le cadre de cette fonctionnalité, Amazon EMR sur EKS avec Flink fournit les fonctionnalités suivantes :

- Just-in-time pointage de contrôle — Les tâches de streaming Flink peuvent répondre à une interruption d'une instance Spot, effectuer un point de contrôle just-in-time (JIT) des tâches en cours d'exécution et empêcher la planification de tâches supplémentaires sur ces instances Spot. Les points de contrôle juste à temps sont pris en charge avec le planificateur par défaut et le planificateur adaptatif.
- Mécanisme de redémarrage combiné : un mécanisme de redémarrage combiné tente de redémarrer la tâche une fois que le parallélisme des ressources cibles est atteint ou que la fenêtre actuellement configurée est terminée. Cela permet également d'éviter des redémarrages successifs susceptibles d'être provoqués par l'arrêt de plusieurs instances Spot. Le mécanisme de redémarrage combiné est uniquement disponible avec le planificateur adaptatif.

Ces fonctionnalités offrent les avantages suivants :

- Vous pouvez tirer parti des instances Spot pour exécuter des Task Managers et réduire les dépenses liées aux clusters.
- L'amélioration de la réactivité du Task Manager sur les instances Spot se traduit par une résilience accrue et une planification des tâches plus efficace.
- Le temps de fonctionnement de vos tâches Flink sera plus élevé, car les redémarrages après l'arrêt d'une instance Spot seront moins nombreux.

Comment fonctionne une mise hors service progressive

Prenons l'exemple suivant : vous provisionnez un cluster Amazon EMR sur EKS exécutant Apache Flink, et vous spécifiez des nœuds à la demande pour le Job Manager et des nœuds d'instance Spot pour le Task Manager. Deux minutes avant l'arrêt, le Task Manager reçoit un avis d'interruption.

Dans ce scénario, le Job Manager gère le signal d'interruption d'instance Spot, empêche la planification de tâches supplémentaires sur l'instance Spot et crée un point de contrôle juste à temps pour la tâche de streaming.

Le Job Manager ne redémarre le graphe des tâches que lorsque les nouvelles ressources disponibles sont suffisantes pour permettre le parallélisme des tâches actuelles dans la fenêtre d'intervalle de redémarrage actuelle. L'intervalle de fenêtre de redémarrage est déterminée en fonction du temps nécessaire au remplacement de l'instance Spot, à la création de nouveaux pods du Task Manager et à l'enregistrement auprès du Job Manager.

Conditions préalables

Pour utiliser la mise hors service progressive, créez et exécutez une tâche de streaming sur un cluster Amazon EMR on EKS exécutant Apache Flink. Activez le planificateur adaptatif et les Task Managers planifiés sur au moins une instance Spot, comme illustré dans l'exemple suivant. Vous devez utiliser des nœuds à la demande pour le Job Manager, et vous pouvez utiliser des nœuds à la demande pour les Task Managers à condition qu'il existe également au moins une instance Spot.

```
apiVersion: flink.apache.org/v1beta1
kind: FlinkDeployment
metadata:
  name: deployment_name
spec:
```

```
  flinkVersion: v1_17
  flinkConfiguration:
    taskmanager.numberOfTaskSlots: "2"
    cluster.taskmanager.graceful-decommission.enabled: "true"
    execution.checkpointing.interval: "240s"
    jobmanager.adaptive-scheduler.combined-restart.enabled: "true"
    jobmanager.adaptive-scheduler.combined-restart.window-interval : "1m"
  serviceAccount: flink
  jobManager:
    resource:
      memory: "2048m"
      cpu: 1
    nodeSelector:
      'eks.amazonaws.com/capacityType': 'ON_DEMAND'
  taskManager:
    resource:
      memory: "2048m"
      cpu: 1
    nodeSelector:
      'eks.amazonaws.com/capacityType': 'SPOT'
  job:
    jarURI: flink_job_jar_path
```

Configuration

Cette section couvre la plupart des configurations que vous pouvez spécifier pour vos besoins de mise hors service.

Clé	Description	Valeur par défaut	Valeurs acceptables
`cluster.taskmanager.graceful-decommission.enabled`	Active la mise hors service progressive du Task Manager.	`true`	`true`, `false`
`jobmanager.adaptiv`	Active le mécanisme de redémarrage combiné dans le planificateur adaptatif.	`false`	`true`, `false`

Clé	Description	Valeur par défaut	Valeurs acceptables
`e-scheduler.combined-restart.enabled`			
`jobmanager.adaptive-scheduler.combined-restart.window-interval`	Intervalle de fenêtre de redémarrage combiné pour l'exécution de redémarrages combinés de la tâche. Un entier sans unité est interprété comme une milliseconde.	`1m`	Exemples : `30`, `60s`, `3m`, `1h`

Utilisation d'Autoscaler pour les applications Flink

L'outil de mise à l'échelle automatique de l'opérateur peut contribuer à réduire la surcharge en recueillant des métriques des tâches Flink et en modifiant automatiquement le parallélisme au niveau du sommet de la tâche. Voici un exemple de ce à quoi pourrait ressembler votre configuration :

```
apiVersion: flink.apache.org/v1beta1
kind: FlinkDeployment
metadata:
  ...
spec:
  ...
  flinkVersion: v1_18
  flinkConfiguration:
    job.autoscaler.enabled: "true"
    job.autoscaler.stabilization.interval: 1m
    job.autoscaler.metrics.window: 5m
    job.autoscaler.target.utilization: "0.6"
    job.autoscaler.target.utilization.boundary: "0.2"
    job.autoscaler.restart.time: 2m
    job.autoscaler.catch-up.duration: 5m
    pipeline.max-parallelism: "720"
  ...
```

Cette configuration utilise les valeurs par défaut de la dernière version d'Amazon EMR. Si vous utilisez d'autres versions, vous pouvez avoir des valeurs différentes.

> **Note**
>
> Depuis Amazon EMR 7.2.0, il n'est pas nécessaire d'inclure le préfixe `kubernetes.operator` dans votre configuration. Si vous utilisez la version 7.1.0 ou une version antérieure, vous devez utiliser le préfixe avant chaque configuration. Par exemple, vous devez spécifier`kubernetes.operator.job.autoscaler.scaling.enabled`.

Les options de configuration de l'outil de mise à l'échelle automatique sont les suivantes.

- `job.autoscaler.scaling.enabled`— indique s'il faut activer l'exécution de la mise à l'échelle des sommets par l'autoscaler. La valeur par défaut est `true`. Si vous désactivez cette configuration, l'autoscaler collecte uniquement les métriques et évalue le parallélisme suggéré pour chaque sommet, mais ne met pas à niveau les tâches.
- `job.autoscaler.stabilization.interval` : la période de stabilisation au cours de laquelle aucune nouvelle mise à l'échelle ne sera exécutée. La valeur par défaut est de 5 minutes.
- `job.autoscaler.metrics.window` : la taille de la fenêtre d'agrégation des métriques de mise à l'échelle. Une fenêtre de taille plus grande rend le processus plus fluide et stable, toutefois, cela peut ralentir la réactivité de l'outil de mise à l'échelle automatique face à des variations brusques de charge. La valeur par défaut est de 15 minutes. Nous vous recommandons d'expérimenter en utilisant une valeur comprise entre 3 et 60 minutes.
- `job.autoscaler.target.utilization` : l'utilisation du sommet cible pour assurer des performances stables de la tâche et une marge pour les fluctuations de charge. Le `0.7` ciblage par défaut est de 70 % utilization/load pour les sommets des tâches.
- `job.autoscaler.target.utilization.boundary` : la limite d'utilisation du sommet cible qui sert de tampon supplémentaire pour éviter une mise à l'échelle immédiate en cas de fluctuations de charge. La valeur par défaut est`0.3`, ce qui signifie qu'un écart de 30 % par rapport à l'utilisation cible est autorisé avant de déclencher une action de dimensionnement.
- `ob.autoscaler.restart.time` : l'heure prévue pour redémarrer l'application. La valeur par défaut est de 5 minutes.
- `job.autoscaler.catch-up.duration` : le temps estimé pour rattraper le retard, c'est-à-dire pour traiter entièrement tout retard accumulé après l'achèvement d'une opération de mise à l'échelle. La valeur par défaut est de 5 minutes. En réduisant la durée de rattrapage, l'outil de mise

à l'échelle automatique doit réserver une plus grande capacité supplémentaire pour les actions de mise à l'échelle.

- `pipeline.max-parallelism` : le parallélisme maximal que l'outil de mise à l'échelle automatique peut utiliser. L'outil de mise à l'échelle automatique ignore cette limite si elle est supérieure au parallélisme maximal configuré dans la configuration Flink ou directement sur chaque opérateur. La valeur par défaut est -1. Notez que l'outil de mise à l'échelle automatique calcule le parallélisme comme un diviseur du parallélisme maximum. Il est donc recommandé de sélectionner des paramètres de parallélisme maximum qui offrent une large gamme de diviseurs potentiels, plutôt que de se baser uniquement sur les valeurs par défaut proposées par Flink. Nous recommandons d'utiliser des multiples de 60 pour cette configuration, tels que 120, 180, 240, 360, 720, etc.

Pour une page de référence plus détaillée sur la configuration, consultez la rubrique Configuration de l'outil de mise à l'échelle automatique.

Réglage automatique des paramètres de l'Autoscaler

Cette section décrit le comportement de réglage automatique pour les différentes versions d'Amazon EMR. Il décrit également en détail les différentes configurations d'auto-scaling.

> **Note**
>
> Amazon EMR 7.2.0 et versions ultérieures utilisent la configuration open source `job.autoscaler.restart.time-tracking.enabled` pour permettre l'estimation du temps de redimensionnement. L'estimation du temps de redimensionnement possède les mêmes fonctionnalités que le réglage automatique d'Amazon EMR. Vous n'avez donc pas à attribuer manuellement des valeurs empiriques à l'heure de redémarrage.
> Vous pouvez toujours utiliser le réglage automatique d'Amazon EMR si vous utilisez Amazon EMR 7.1.0 ou une version antérieure.

7.2.0 and higher

Amazon EMR 7.2.0 et versions ultérieures mesurent le temps de redémarrage réel requis pour appliquer les décisions de dimensionnement automatique. Dans les versions 7.1.0 et antérieures, vous deviez utiliser la configuration `job.autoscaler.restart.time` pour configurer manuellement le temps de redémarrage maximal estimé. En utilisant la

configuration`job.autoscaler.restart.time-tracking.enabled`, il vous suffit de saisir une heure de redémarrage pour la première mise à l'échelle. Ensuite, l'opérateur enregistre l'heure de redémarrage réelle et l'utilisera pour les redimensionnements ultérieurs.

Pour activer ce suivi, utilisez la commande suivante :

```
job.autoscaler.restart.time-tracking.enabled: true
```

Les configurations associées pour l'estimation du temps de redimensionnement sont les suivantes.

Configuration	Obligatoire	Par défaut	Description
job.autoscaler.restart.time-tracking.enabled	Non	False	Indique si le Flink Autoscaler doit ajuster automatiquement les configurations au fil du temps afin d'optimiser les décisions de dimensionnement. Notez que l'Autoscaler peut uniquement régler automatiquement le paramètre Autoscaler. `restart.time`
job.autoscaler.restart.time	Non	5 min	Temps de redémarrage prévu utilisé par Amazon EMR on EKS jusqu'à ce que l'opérateur puisse déterminer le temps de redémarrage réel à partir des mises à l'échelle précédentes.
job.autoscaler.restart.time-tracking.limit	Non	15 min	Durée de redémarrage maximale observée lorsque le `job.autoscaler.restart.time-tracking.enabled` paramètre est réglé sur`true`.

Voici un exemple de spécification de déploiement que vous pouvez utiliser pour essayer d'estimer le temps de redimensionnement :

```
apiVersion: flink.apache.org/v1beta1
kind: FlinkDeployment
metadata:
  name: autoscaling-example
spec:
  flinkVersion: v1_18
  flinkConfiguration:

    # Autoscaler parameters
    job.autoscaler.enabled: "true"
    job.autoscaler.scaling.enabled: "true"
    job.autoscaler.stabilization.interval: "5s"
    job.autoscaler.metrics.window: "1m"

    job.autoscaler.restart.time-tracking.enabled: "true"
    job.autoscaler.restart.time: "2m"
    job.autoscaler.restart.time-tracking.limit: "10m"

    jobmanager.scheduler: adaptive
    taskmanager.numberOfTaskSlots: "1"
    pipeline.max-parallelism: "12"

  executionRoleArn: <JOB ARN>
  emrReleaseLabel: emr-7.12.0-flink-latest
  jobManager:
    highAvailabilityEnabled: false
    storageDir: s3://<s3_bucket>/flink/autoscaling/ha/
    replicas: 1
    resource:
      memory: "1024m"
      cpu: 0.5
  taskManager:
    resource:
      memory: "1024m"
      cpu: 0.5
  job:
    jarURI: s3://<s3_bucket>/some-job-with-back-pressure
    parallelism: 1
    upgradeMode: stateless
```

Pour simuler la contre-pression, utilisez les spécifications de déploiement suivantes.

```
job:
    jarURI: s3://<s3_bucket>/pyflink-script.py
    entryClass: "org.apache.flink.client.python.PythonDriver"
    args: ["-py", "/opt/flink/usrlib/pyflink-script.py"]
    parallelism: 1
    upgradeMode: stateless
```

Téléchargez le script Python suivant dans votre compartiment S3.

```
import logging
import sys
import time
import random

from pyflink.datastream import StreamExecutionEnvironment
from pyflink.table import StreamTableEnvironment

TABLE_NAME="orders"
QUERY=f"""
CREATE TABLE {TABLE_NAME} (
  id INT,
  order_time AS CURRENT_TIMESTAMP,
  WATERMARK FOR order_time AS order_time - INTERVAL '5' SECONDS
)
WITH (
  'connector' = 'datagen',
  'rows-per-second'='10',
  'fields.id.kind'='random',
  'fields.id.min'='1',
  'fields.id.max'='100'
);
"""

def create_backpressure(i):
    time.sleep(2)
    return i

def autoscaling_demo():
    env = StreamExecutionEnvironment.get_execution_environment()
    t_env = StreamTableEnvironment.create(env)
    t_env.execute_sql(QUERY)
```

```
    res_table = t_env.from_path(TABLE_NAME)

    stream =  t_env.to_data_stream(res_table) \
      .shuffle().map(lambda x: create_backpressure(x))\
      .print()
    env.execute("Autoscaling demo")

if __name__ == '__main__':
    logging.basicConfig(stream=sys.stdout, level=logging.INFO, format="%(message)s")
    autoscaling_demo()
```

Pour vérifier que l'estimation du temps de redimensionnement fonctionne, assurez-vous que l'enregistrement des DEBUG niveaux par l'opérateur Flink est activé. L'exemple ci-dessous montre comment mettre à jour le fichier de diagramme de barre`values.yaml`. Réinstallez ensuite le tableau de bord mis à jour et réexécutez votre tâche Flink.

```
log4j-operator.properties: |+
  # Flink Operator Logging Overrides
  rootLogger.level = DEBUG
```

Obtenez le nom de votre module leader.

```
ip=$(kubectl get configmap -n $NAMESPACE <job-name>-cluster-config-map -o json | jq
 -r ".data[\"org.apache.flink.k8s.leader.restserver\"]" | awk -F: '{print $2}' | awk
 -F '/' '{print $3}')

kubectl get pods -n $NAMESPACE -o json | jq -r ".items[] | select(.status.podIP ==
 \"$ip\") | .metadata.name"
```

Exécutez la commande suivante pour obtenir le temps de redémarrage réel utilisé dans les évaluations des métriques.

```
kubectl logs <FLINK-OPERATOR-POD-NAME> -c flink-kubernetes-operator -n <OPERATOR-
NAMESPACE> -f | grep "Restart time used in scaling summary computation"
```

Vous devriez voir des journaux similaires aux suivants. Notez que seule la première mise à l'échelle l'utilise `job.autoscaler.restart.time`. Les mises à l'échelle suivantes utilisent le temps de redémarrage observé.

```
2024-05-16 17:17:32,590 o.a.f.a.ScalingExecutor        [DEBUG][default/autoscaler-
example] Restart time used in scaling summary computation: PT2M
```

```
2024-05-16 17:19:03,787 o.a.f.a.ScalingExecutor        [DEBUG][default/autoscaler-
example] Restart time used in scaling summary computation: PT14S
2024-05-16 17:19:18,976 o.a.f.a.ScalingExecutor        [DEBUG][default/autoscaler-
example] Restart time used in scaling summary computation: PT14S
2024-05-16 17:20:50,283 o.a.f.a.ScalingExecutor        [DEBUG][default/autoscaler-
example] Restart time used in scaling summary computation: PT14S
2024-05-16 17:22:21,691 o.a.f.a.ScalingExecutor        [DEBUG][default/autoscaler-
example] Restart time used in scaling summary computation: PT14S
```

7.0.0 and 7.1.0

Le logiciel open source intégré Flink Autoscaler utilise de nombreux indicateurs pour prendre les meilleures décisions en matière de dimensionnement. Cependant, les valeurs par défaut qu'il utilise pour ses calculs sont censées s'appliquer à la plupart des charges de travail et peuvent ne pas être optimales pour une tâche donnée. La fonction de réglage automatique ajoutée à la version Amazon EMR on EKS du Flink Operator examine les tendances historiques observées sur des métriques capturées spécifiques, puis tente de calculer la valeur la plus optimale adaptée à la tâche donnée.

Configuration	Obligatoire	Par défaut	Description
kubernetes.operator.job.autoscaler.autotune.enable	Non	False	Indique si le Flink Autoscaler doit ajuster automatiquement les configurations au fil du temps afin d'optimiser les décisions des autoscalers. Actuellement, l'Autoscaler peut uniquement régler automatiquement le paramètre Autoscaler. `restart.time`
kubernetes.operator.job.autoscaler.autotune.metrics.history.max.count	Non	3	Indique le nombre de métriques historiques Amazon EMR sur EKS que l'Autoscaler conserve dans la carte de configuration des

Configuration	Obligatoire	Par défaut	Description
			métriques Amazon EMR on EKS.
kubernetes.operator.job.autoscaler.autotune.metrics.restart.count	Non	3	Indique le nombre de redémarrages effectués par Autoscaler avant de commencer à calculer le temps de redémarrage moyen pour une tâche donnée.

Pour activer le réglage automatique, vous devez avoir effectué les opérations suivantes :

- Définissez `kubernetes.operator.job.autoscaler.autotune.enable:` sur `true`
- Définissez `metrics.job.status.enable:` sur `TOTAL_TIME`
- A suivi la configuration de l'utilisation d'Autoscaler pour les applications Flink pour activer la mise à l'échelle automatique.

Voici un exemple de spécification de déploiement que vous pouvez utiliser pour essayer le réglage automatique.

```
apiVersion: flink.apache.org/v1beta1
kind: FlinkDeployment
metadata:
  name: autoscaling-example
spec:
  flinkVersion: v1_18
  flinkConfiguration:

    # Autotuning parameters
    kubernetes.operator.job.autoscaler.autotune.enable: "true"
    kubernetes.operator.job.autoscaler.autotune.metrics.history.max.count: "2"
    kubernetes.operator.job.autoscaler.autotune.metrics.restart.count: "1"
    metrics.job.status.enable: TOTAL_TIME

    # Autoscaler parameters
    kubernetes.operator.job.autoscaler.enabled: "true"
```

```
    kubernetes.operator.job.autoscaler.scaling.enabled: "true"
    kubernetes.operator.job.autoscaler.stabilization.interval: "5s"
    kubernetes.operator.job.autoscaler.metrics.window: "1m"

    jobmanager.scheduler: adaptive

    taskmanager.numberOfTaskSlots: "1"
    state.savepoints.dir: s3://<S3_bucket>/autoscaling/savepoint/
    state.checkpoints.dir: s3://<S3_bucket>/flink/autoscaling/checkpoint/
    pipeline.max-parallelism: "4"

  executionRoleArn: <JOB ARN>
  emrReleaseLabel: emr-6.14.0-flink-latest
  jobManager:
    highAvailabilityEnabled: true
    storageDir: s3://<S3_bucket>/flink/autoscaling/ha/
    replicas: 1
    resource:
      memory: "1024m"
      cpu: 0.5
  taskManager:
    resource:
      memory: "1024m"
      cpu: 0.5
  job:
    jarURI: s3://<S3_bucket>/some-job-with-back-pressure
    parallelism: 1
    upgradeMode: last-state
```

Pour simuler la contre-pression, utilisez les spécifications de déploiement suivantes.

```
  job:
    jarURI: s3://<S3_bucket>/pyflink-script.py
    entryClass: "org.apache.flink.client.python.PythonDriver"
    args: ["-py", "/opt/flink/usrlib/pyflink-script.py"]
    parallelism: 1
    upgradeMode: last-state
```

Téléchargez le script Python suivant dans votre compartiment S3.

```
import logging
import sys
import time
```

```
import random

from pyflink.datastream import StreamExecutionEnvironment
from pyflink.table import StreamTableEnvironment

TABLE_NAME="orders"
QUERY=f"""
CREATE TABLE {TABLE_NAME} (
  id INT,
  order_time AS CURRENT_TIMESTAMP,
  WATERMARK FOR order_time AS order_time - INTERVAL '5' SECONDS
)
WITH (
  'connector' = 'datagen',
  'rows-per-second'='10',
  'fields.id.kind'='random',
  'fields.id.min'='1',
  'fields.id.max'='100'
);
"""

def create_backpressure(i):
    time.sleep(2)
    return i

def autoscaling_demo():
    env = StreamExecutionEnvironment.get_execution_environment()
    t_env = StreamTableEnvironment.create(env)
    t_env.execute_sql(QUERY)
    res_table = t_env.from_path(TABLE_NAME)

    stream =  t_env.to_data_stream(res_table) \
      .shuffle().map(lambda x: create_backpressure(x))\
      .print()
    env.execute("Autoscaling demo")

if __name__ == '__main__':
    logging.basicConfig(stream=sys.stdout, level=logging.INFO, format="%(message)s")
    autoscaling_demo()
```

Pour vérifier que votre autotuner fonctionne, utilisez les commandes suivantes. Notez que vous devez utiliser les informations de votre propre module leader pour le Flink Operator.

Obtenez d'abord le nom de votre module leader.

```
ip=$(kubectl get configmap -n $NAMESPACE <job-name>-cluster-config-map -o json | jq
 -r ".data[\"org.apache.flink.k8s.leader.restserver\"]" | awk -F: '{print $2}' | awk
 -F '/' '{print $3}')

kubectl get pods -n $NAMESPACE -o json | jq -r ".items[] | select(.status.podIP ==
 \"$ip\") | .metadata.name"
```

Une fois que vous avez le nom de votre module leader, vous pouvez exécuter la commande suivante.

```
kubectl logs -n $NAMESPACE  -c flink-kubernetes-operator --follow <YOUR-FLINK-
OPERATOR-POD-NAME>  | grep -E 'EmrEks|autotun|calculating|restart|autoscaler'
```

Vous devriez voir des journaux similaires aux suivants.

```
[m[33m2023-09-13 20:10:35,941[m [36mc.a.c.f.k.o.a.EmrEksMetricsAutotuner[m
 [36m[DEBUG][flink/autoscaling-example] Using the latest
 Emr Eks Metric for calculating restart.time for autotuning:
 EmrEksMetrics(restartMetric=RestartMetric(restartingTime=65, numRestarts=1))

[m[33m2023-09-13 20:10:35,941[m [36mc.a.c.f.k.o.a.EmrEksMetricsAutotuner[m
 [32m[INFO ][flink/autoscaling-example] Calculated average restart.time metric via
 autotuning to be: PT0.065S
```

Maintenance et résolution des problèmes liés aux tâches Flink sur Amazon EMR sur EKS

Les sections suivantes expliquent comment gérer vos tâches Flink de longue durée et fournissent des conseils sur la manière de résoudre certains problèmes courants liés aux tâches Flink.

Maintenance des applications Flink

Rubriques

- Modes de mise à niveau

Les applications Flink sont généralement conçues pour fonctionner pendant de longues périodes (plusieurs semaines, mois, voire plusieurs années). Comme pour tous les services de longue durée, les applications de streaming Flink doivent faire l'objet d'une maintenance. Cela inclut des corrections de bogues, des améliorations et la migration vers un cluster Flink exécutant une version plus récente.

L'évolution des spécifications pour les ressources `FlinkDeployment` et `FlinkSessionJob` implique de mettre à niveau l'application en cours d'exécution. Pour ce faire, l'opérateur arrête la tâche en cours d'exécution (à moins qu'elle soit déjà suspendue) et la redéploie avec les dernières spécifications et, pour les applications avec état, l'état de l'exécution précédente.

Les utilisateurs choisissent comment gérer l'état lorsque les applications avec état s'arrêtent et sont restaurées avec le paramètre `upgradeMode` défini sur `JobSpec`.

Modes de mise à niveau

Introduction optionnelle

Applications sans état

Les applications sans état sont mises à niveau à partir de l'état vide.

Dernier état

Les mises à niveau rapides, quel que soit l'état de l'application (même pour les tâches ayant échoué), ne nécessitent pas une tâche saine, car elles utilisent toujours le dernier point de contrôle réussi. Une récupération manuelle peut être nécessaire en cas de perte de métadonnées de haute disponibilité. Pour limiter l'ancienneté du dernier point de contrôle utilisé pour la récupération, configurez le paramètre `kubernetes.operator.job.upgrade.last-state.max.allowed.checkpoint.age`. Si le point de contrôle est plus ancien que la valeur configurée, un point de sauvegarde sera utilisé à la place pour les tâches saines. Cette fonctionnalité n'est pas prise en charge en mode session.

Points de sauvegarde

Utilisez le point de sauvegarde pour la mise à niveau, offrant ainsi une sécurité maximale et la possibilité de servir de backup/fork point. Le point de sauvegarde sera créé lors de la mise à niveau. Notez que la tâche Flink doit être en cours d'exécution pour que le point de sauvegarde puisse être créé. Si la tâche n'est pas fonctionnelle, le dernier point de contrôle sera utilisé (sauf kubernetes.operator.job.upgrade). last-state-fallback.enabled est défini sur false). Si le dernier point de contrôle n'est pas disponible, la mise à niveau de la tâche échouera.

Résolution des problèmes

Cette section explique comment résoudre les problèmes liés à Amazon EMR on EKS. Pour plus d'informations sur la manière de résoudre les problèmes généraux liés à Amazon EMR, consultez la rubrique Résolution des problèmes liés à un cluster dans le Guide de gestion d'Amazon EMR.

- Résolution des problèmes liés à l'utilisation de PersistentVolumeClaims (PVC)
- Résolution des problèmes de mise à l'échelle automatique verticale d'Amazon EMR on EKS
- Résolution des problèmes liés à l'opérateur Spark d'Amazon EMR on EKS

Résoudre les problèmes liés à Apache Flink sur Amazon EMR sur EKS

Le mappage des ressources est introuvable lors de l'installation des Charts de Helm

Le message d'erreur suivant peut s'afficher lorsque vous installez les Charts de Helm.

```
Error: INSTALLATION FAILED: pulling from host 1234567890.dkr.ecr.us-
west-2.amazonaws.com failed with status code [manifests 6.13.0]: 403 Forbidden Error:
 INSTALLATION FAILED: unable to build kubernetes objects from release manifest:
 [resource mapping not found for name: "flink-operator-serving-cert" namespace: "<the
 namespace to install your operator>" from "": no matches for kind "Certificate" in
 version "cert-manager.io/v1"

ensure CRDs are installed first, resource mapping not found for name: "flink-operator-
selfsigned-issuer" namespace: "<the namespace to install your operator>" " from "": no
 matches for kind "Issuer" in version "cert-manager.io/v1"

ensure CRDs are installed first].
```

Pour résoudre cette erreur, installez cert-manager afin de pouvoir intégrer le composant webhook. Vous devez installer cert-manager sur chaque cluster Amazon EKS que vous utilisez.

```
kubectl apply -f https://github.com/cert-manager/cert-manager/releases/download/v1.12.0
```

Service AWS erreur d'accès refusé

Si un message d'erreur access denied s'affiche, vérifiez que le rôle IAM pour `operatorExecutionRoleArn` dans le fichier `values.yaml` de Charts de Helm dispose des

autorisations appropriées. Vérifiez également que le rôle IAM sous `executionRoleArn` dans votre spécification `FlinkDeployment` dispose des autorisations appropriées.

`FlinkDeployment` est bloqué

Si votre `FlinkDeployment` est bloqué à l'état arrêté, suivez les étapes suivantes pour forcer la suppression du déploiement :

1. Modifiez l'exécution du déploiement.

```
kubectl edit -n Flink Namespace flinkdeployments/App Name
```

2. Supprimez ce finalisateur.

```
finalizers:
  - flinkdeployments.flink.apache.org/finalizer
```

3. Supprimez le déploiement.

```
kubectl delete -n Flink Namespace flinkdeployments/App Name
```

AWSBadRequestException problème s3a lors de l'exécution d'une application Flink dans un opt-in Région AWS

Si vous exécutez une application Flink dans le cadre d'un opt-in Région AWS, les erreurs suivantes peuvent s'afficher :

```
Caused by: org.apache.hadoop.fs.s3a.AWSBadRequestException: getFileStatus on
s3://flink.txt: com.amazonaws.services.s3.model.AmazonS3Exception: Bad Request
(Service: Amazon S3; Status Code: 400; Error Code: 400 Bad Request; Request ID:
 ABCDEFGHIJKL; S3 Extended Request ID:
ABCDEFGHIJKLMNOP=; Proxy: null), S3 Extended Request ID: ABCDEFGHIJKLMNOP=:400 Bad
 Request: Bad Request
(Service: Amazon S3; Status Code: 400; Error Code: 400 Bad Request; Request ID:
 ABCDEFGHIJKL; S3 Extended Request ID: ABCDEFGHIJKLMNOP=; Proxy: null)
```

```
Caused by: org.apache.hadoop.fs.s3a.AWSBadRequestException: getS3Region on flink-
application: software.amazon.awssdk.services.s3.model.S3Exception: null
(Service: S3, Status Code: 400, Request ID: ABCDEFGHIJKLMNOP, Extended Request ID:
 ABCDEFGHIJKLMNOPQRST==):null: null
```

```
(Service: S3, Status Code: 400, Request ID: ABCDEFGHIJKLMNOP, Extended Request ID:
 AHl42uDNaTUFOus/5IIVNvSakBcMjMCH7dd37ky0vE6jhABCDEFGHIJKLMNOPQRST==)
```

Pour corriger ces erreurs, utilisez la configuration suivante dans votre fichier de `FlinkDeployment` définition.

```
spec:
  flinkConfiguration:
    taskmanager.numberOfTaskSlots: "2"
    fs.s3a.endpoint.region: OPT_IN_AWS_REGION_NAME
```

Nous vous recommandons également d'utiliser le fournisseur SDKv2 d'informations d'identification :

```
fs.s3a.aws.credentials.provider:
 software.amazon.awssdk.auth.credentials.WebIdentityTokenFileCredentialsProvider
```

Si vous souhaitez utiliser le fournisseur SDKv1 d'informations d'identification, assurez-vous que votre SDK prend en charge votre région d'adhésion. Pour plus d'informations, consultez le aws-sdk-java GitHub référentiel.

`S3 AWSBadRequestException`Si vous exécutez des instructions SQL Flink dans une région optionnelle, assurez-vous de définir la configuration `fs.s3a.endpoint.region: OPT_IN_AWS_REGION_NAME` dans vos spécifications de configuration Flink.

S3A AWSBad RequestException lors de l'exécution d'une tâche de session Flink dans les régions CN

Pour les versions 6.15.0 à 7.2.0 d'Amazon EMR, les messages d'erreur suivants peuvent s'afficher lorsque vous exécutez une tâche de session Flink dans les régions CN. Il s'agit notamment de la Chine (Pékin) et de la Chine (Ningxia) :

```
Error:
 {"type":"org.apache.flink.kubernetes.operator.exception.ReconciliationException","message":"or
                    getFileStatus on s3://ABCDPath:
 software.amazon.awssdk.services.s3.model.S3Exception: null (Service: S3, Status Code:
 400, Request ID: ABCDEFGH, Extended Request ID:
                    ABCDEFGH:null: null (Service: S3, Status Code: 400, Request ID:
 ABCDEFGH, Extended Request ID: ABCDEFGH","additionalMetadata":{},"throwableList":

 [{"type":"org.apache.hadoop.fs.s3a.AWSBadRequestException","message":"getFileStatus on
 s3://ABCDPath: software.amazon.awssdk.services.s3.model.S3Exception:
```

```
                    null (Service: S3, Status Code: 400, Request ID: ABCDEFGH, Extended
 Request ID: ABCDEFGH:null: null (Service: S3, Status Code: 400, Request ID: ABCDEFGH,
                    Extended Request ID: ABCDEFGH","additionalMetadata":{}},
{"type":"software.amazon.awssdk.services.s3.model.S3Exception","message":"null
 (Service: S3, Status Code: 400,
                    Request ID: ABCDEFGH, Extended Request ID:
 ABCDEFGH","additionalMetadata":{}}]}
```

Il existe une prise de conscience de ce problème. L'équipe travaille à la correction des opérateurs Flink pour toutes ces versions. Toutefois, avant de terminer le correctif, pour corriger cette erreur, vous devez télécharger le diagramme de barre de l'opérateur Flink, le décompresser (extraire le fichier compressé) et apporter des modifications de configuration dans le graphique de barre.

Les étapes spécifiques sont les suivantes :

1. Accédez, en particulier, aux répertoires dans votre dossier local pour le graphique de barre, et exécutez la ligne de commande suivante pour extraire le graphique de barre et le décompresser (extraire).

   ```
   helm pull oci://public.ecr.aws/emr-on-eks/flink-kubernetes-operator \
   --version $VERSION \
   --namespace $NAMESPACE
   ```

   ```
   tar -zxvf flink-kubernetes-operator-$VERSION.tgz
   ```

2. Accédez au dossier Helm Chart et trouvez le `templates/flink-operator.yaml` fichier.
3. Recherchez `flink-operator-config` ConfigMap et ajoutez la `fs.s3a.endpoint.region` configuration suivante dans le`flink-conf.yaml`. Par exemple :

   ```
   {{- if .Values.defaultConfiguration.create }}
   apiVersion: v1
   kind: ConfigMap
   metadata:
     name: flink-operator-config
     namespace: {{ .Release.Namespace }}
     labels:
       {{- include "flink-operator.labels" . | nindent 4 }}
   data:
     flink-conf.yaml: |+
   fs.s3a.endpoint.region: {{ .Values.emrContainers.awsRegion }}
   ```

4. Installez le tableau de bord local et exécutez votre tâche.

Versions prises en charge pour Amazon EMR sur EKS avec Apache Flink

Apache Flink est disponible avec les versions suivantes d'Amazon EMR sur EKS. Pour plus d'informations sur toutes les versions disponibles, consultez Versions Amazon EMR on EKS.

Étiquette de version	Java	Flink	Opérateur Flink
emr-7.2.0-flink-latest	17	1.18.1	-
emr-7.2.0-flink-k8s-operator-latest	11	-	1.8.0
emr-7.1.0-flink-latest	17	1.18.1	-
emr-7.1.0-flink-k8s-operator-latest	11	-	1.6.1
emr-7.0.0-flink-latest	11	1.18.0	-
emr-7.0.0-flink-k8s-operator-latest	11	-	1.6.1
emr-6.15.0-flink-latest	11	1,17.1	-
emr-6.15.0-flink-k8s-operator-latest	11	-	1.6.0
emr-6.14.0-flink-latest	11	1,17.1	-
emr-6.14.0-flink-k8s-operator-latest	11	-	1.6.0
emr-6.13.0-flink-latest	11	1.17.0	-
emr-6.13.0-flink-k8s-operator-latest	11	-	1.5.0

Exécution de tâches Spark avec Amazon EMR sur EKS

Une exécution de tâche est une unité de travail, telle qu'un fichier JAR, un PySpark script ou une requête SparkSQL Spark, que vous soumettez à Amazon EMR sur EKS. Cette rubrique fournit une vue d'ensemble de la gestion des exécutions de tâches à l'aide de la console Amazon EMR AWS CLI, de l'affichage des exécutions de tâches à l'aide de la console Amazon EMR et de la résolution des erreurs courantes d'exécution de tâches.

Notez que vous ne pouvez pas exécuter de tâches IPv6 Spark sur Amazon EMR sur EKS

> **Note**
>
> Avant de soumettre une tâche exécutée à l'aide d'Amazon EMR on EKS, vous devez effectuer les étapes décrites dans la rubrique Configuration d'Amazon EMR on EKS.

Rubriques

Exécution de tâches Spark avec `StartJobRun`

Cette section inclut des étapes de configuration détaillées pour préparer votre environnement à exécuter des tâches Spark, puis fournit des step-by-step instructions pour soumettre une exécution de tâche avec des paramètres spécifiés.

Rubriques

Configuration d'Amazon EMR on EKS

Effectuez les tâches suivantes pour vous préparer à utiliser Amazon EMR on EKS. Si vous êtes déjà inscrit à Amazon Web Services (AWS) et que vous utilisez Amazon EKS, vous êtes presque prêt à utiliser Amazon EMR on EKS. Ignorez toutes les tâches que vous avez déjà effectuées.

> **Note**
>
> Vous pouvez également suivre l'atelier Amazon EMR on EKS pour configurer toutes les ressources nécessaires à l'exécution des tâches Spark sur Amazon EMR on EKS. L'atelier fournit également une automatisation en utilisant des CloudFormation modèles pour créer les ressources nécessaires à votre démarrage. Pour d'autres modèles et meilleures pratiques, consultez notre guide des meilleures pratiques en matière de conteneurs EMR sur. GitHub

1. Installez ou mettez à jour vers la dernière version du AWS CLI
2. Configurez kubectl et eksctl.
3. Commencez avec Amazon EKS — eksctl
4. Activer l'accès au cluster pour Amazon EMR sur EKS
5. Activer les rôles IAM pour le cluster EKS
6. Autorisation des utilisateurs à accéder à Amazon EMR on EKS
7. Enregistrement du cluster Amazon EKS dans Amazon EMR

Activation de l'accès aux clusters pour Amazon EMR on EKS

Les sections suivantes présentent deux manières d'activer l'accès au cluster. La première consiste à utiliser la gestion de l'accès au cluster (CAM) d'Amazon EKS et la seconde montre comment effectuer des étapes manuelles pour activer l'accès au cluster.

Activer l'accès au cluster à l'aide d'EKS Access Entry (recommandé)

> **Note**
>
> `aws-auth` ConfigMap est obsolète. La méthode recommandée pour gérer l'accès à Kubernetes APIs est celle des entrées d'accès.

Amazon EMR est intégré à la gestion des accès aux clusters (CAM) d'Amazon EKS, ce qui vous permet d'automatiser la configuration des politiques AuthN et AuthZ nécessaires pour exécuter des tâches Amazon EMR Spark dans les espaces de noms des clusters Amazon EKS. Lorsque vous créez un cluster virtuel à partir d'un espace de noms de cluster Amazon EKS, Amazon EMR configure automatiquement toutes les autorisations nécessaires, de sorte que vous n'avez pas besoin d'ajouter d'étapes supplémentaires à vos flux de travail actuels.

> **Note**
>
> L'intégration d'Amazon EMR à Amazon EKS CAM n'est prise en charge que pour les nouveaux clusters virtuels Amazon EMR on EKS. Vous ne pouvez pas migrer des clusters virtuels existants pour utiliser cette intégration.

Conditions préalables

- Assurez-vous que vous utilisez la version 2.15.3 ou supérieure du AWS CLI
- Votre cluster Amazon EKS doit être doté de la version 1.23 ou supérieure.

Configuration

Pour configurer l'intégration entre Amazon EMR et les opérations d' AccessEntry API d'Amazon EKS, assurez-vous d'avoir effectué les étapes suivantes :

- Assurez-vous que celui `authenticationMode` de votre cluster Amazon EKS est défini sur`API_AND_CONFIG_MAP`.

  ```
  aws eks describe-cluster --name <eks-cluster-name>
  ```

  Si ce n'est pas déjà le cas, réglez-le `authenticationMode` sur`API_AND_CONFIG_MAP`.

```
aws eks update-cluster-config
    --name <eks-cluster-name>
    --access-config authenticationMode=API_AND_CONFIG_MAP
```

Pour plus d'informations sur les modes d'authentification, consultez la section Modes d'authentification de cluster.

- Assurez-vous que le rôle IAM que vous utilisez pour exécuter les opérations `CreateVirtualCluster` et `DeleteVirtualCluster` API dispose également des autorisations suivantes :

```
{
  "Effect": "Allow",
  "Action": [
    "eks:CreateAccessEntry"
  ],
  "Resource":
 "arn:<AWS_PARTITION>:eks:<AWS_REGION>:<AWS_ACCOUNT_ID>:cluster/<EKS_CLUSTER_NAME>"
},
{
  "Effect": "Allow",
  "Action": [
    "eks:DescribeAccessEntry",
    "eks:DeleteAccessEntry",
    "eks:ListAssociatedAccessPolicies",
    "eks:AssociateAccessPolicy",
    "eks:DisassociateAccessPolicy"
  ],
  "Resource": "arn:<AWS_PARTITION>:eks:<AWS_REGION>:<AWS_ACCOUNT_ID>:access-entry/
<EKS_CLUSTER_NAME>/role/<AWS_ACCOUNT_ID>/AWSServiceRoleForAmazonEMRContainers/*"
}
```

Concepts et terminologie

Vous trouverez ci-dessous une liste de terminologies et de concepts liés à Amazon EKS CAM.

- Cluster virtuel (VC) : représentation logique de l'espace de noms créé dans Amazon EKS. Il s'agit d'un lien 1:1 vers un espace de noms de cluster Amazon EKS. Vous pouvez l'utiliser pour exécuter des charges de travail Amazon EMR sur un cluster Amazon EKS au sein de l'espace de noms spécifié.

- Namespace : mécanisme permettant d'isoler des groupes de ressources au sein d'un seul cluster EKS.
- Politique d'accès : autorisations qui accordent l'accès et les actions à un rôle IAM au sein d'un cluster EKS.
- Entrée d'accès : entrée créée à l'aide d'un rôle arn. Vous pouvez lier l'entrée d'accès à une politique d'accès pour attribuer des autorisations spécifiques dans le cluster Amazon EKS.
- Cluster virtuel intégré d'entrée d'accès EKS : cluster virtuel créé à l'aide des opérations d'API d'entrée d'accès d'Amazon EKS.

Activez l'accès au cluster en utilisant **`aws-auth`**

Vous devez autoriser Amazon EMR on EKS à accéder à un espace de noms spécifique de votre cluster en effectuant les actions suivantes : créer un rôle Kubernetes, lier le rôle à un utilisateur Kubernetes et associer l'utilisateur Kubernetes au rôle lié au service `AWSServiceRoleForAmazonEMRContainers`. Ces actions sont automatisées dans `eksctl` lorsque la commande de mappage d'identité IAM est utilisée avec `emr-containers` comme nom de service. Vous pouvez effectuer ces opérations facilement à l'aide de la commande suivante.

```
eksctl create iamidentitymapping \
    --cluster my_eks_cluster \
    --namespace kubernetes_namespace \
    --service-name "emr-containers"
```

*`my_eks_cluster`*Remplacez-le par le nom de votre cluster Amazon EKS et remplacez-le par l'espace *`kubernetes_namespace`* de noms Kubernetes créé pour exécuter les charges de travail Amazon EMR.

> **Important**
>
> Vous devez télécharger la dernière version d'eksctl en suivant l'étape précédente. Configurez kubectl et eksctl pour utiliser cette fonctionnalité.

Étapes manuelles à suivre pour activer l'accès aux clusters pour Amazon EMR on EKS

Vous pouvez également suivre les étapes manuelles ci-dessous pour activer l'accès aux clusters pour Amazon EMR on EKS.

1. Création d'un rôle Kubernetes dans un espace de noms spécifique

   Amazon EKS 1.22 - 1.29

   Avec Amazon EKS 1.22 - 1.29, exécutez la commande suivante pour créer un rôle Kubernetes dans un espace de noms spécifique. Ce rôle accorde les autorisations RBAC nécessaires à Amazon EMR on EKS.

```
namespace=my-namespace
cat - >>EOF | kubectl apply -f - >>namespace "${namespace}"
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: emr-containers
  namespace: ${namespace}
rules:
  - apiGroups: [""]
    resources: ["namespaces"]
    verbs: ["get"]
  - apiGroups: [""]
    resources: ["serviceaccounts", "services", "configmaps", "events", "pods",
 "pods/log"]
    verbs: ["get", "list", "watch", "describe", "create", "edit", "delete",
 "deletecollection", "annotate", "patch", "label"]
  - apiGroups: [""]
    resources: ["secrets"]
    verbs: ["create", "patch", "delete", "watch"]
  - apiGroups: ["apps"]
    resources: ["statefulsets", "deployments"]
    verbs: ["get", "list", "watch", "describe", "create", "edit", "delete",
 "annotate", "patch", "label"]
  - apiGroups: ["batch"]
    resources: ["jobs"]
    verbs: ["get", "list", "watch", "describe", "create", "edit", "delete",
 "annotate", "patch", "label"]
  - apiGroups: ["extensions", "networking.k8s.io"]
    resources: ["ingresses"]
    verbs: ["get", "list", "watch", "describe", "create", "edit", "delete",
 "annotate", "patch", "label"]
  - apiGroups: ["rbac.authorization.k8s.io"]
    resources: ["roles", "rolebindings"]
    verbs: ["get", "list", "watch", "describe", "create", "edit", "delete",
 "deletecollection", "annotate", "patch", "label"]
```

```
  - apiGroups: [""]
    resources: ["persistentvolumeclaims"]
    verbs: ["get", "list", "watch", "describe", "create", "edit", "delete",
 "deletecollection", "annotate", "patch", "label"]
EOF
```

Amazon EKS 1.21 and below

À l'aide d'Amazon EKS en version 1.21 et inférieure, exécutez la commande suivante pour créer un rôle Kubernetes dans un espace de noms spécifique. Ce rôle accorde les autorisations RBAC nécessaires à Amazon EMR on EKS.

```
namespace=my-namespace
cat - >>EOF | kubectl apply -f - >>namespace "${namespace}"
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: emr-containers
  namespace: ${namespace}
rules:
  - apiGroups: [""]
    resources: ["namespaces"]
    verbs: ["get"]
  - apiGroups: [""]
    resources: ["serviceaccounts", "services", "configmaps", "events", "pods",
 "pods/log"]
    verbs: ["get", "list", "watch", "describe", "create", "edit", "delete",
 "deletecollection", "annotate", "patch", "label"]
  - apiGroups: [""]
    resources: ["secrets"]
    verbs: ["create", "patch", "delete", "watch"]
  - apiGroups: ["apps"]
    resources: ["statefulsets", "deployments"]
    verbs: ["get", "list", "watch", "describe", "create", "edit", "delete",
 "annotate", "patch", "label"]
  - apiGroups: ["batch"]
    resources: ["jobs"]
    verbs: ["get", "list", "watch", "describe", "create", "edit", "delete",
 "annotate", "patch", "label"]
  - apiGroups: ["extensions"]
    resources: ["ingresses"]
```

```
    verbs: ["get", "list", "watch", "describe", "create", "edit", "delete",
 "annotate", "patch", "label"]
  - apiGroups: ["rbac.authorization.k8s.io"]
    resources: ["roles", "rolebindings"]
    verbs: ["get", "list", "watch", "describe", "create", "edit", "delete",
 "deletecollection", "annotate", "patch", "label"]
  - apiGroups: [""]
    resources: ["persistentvolumeclaims"]
    verbs: ["get", "list", "watch", "describe", "create", "edit", "delete",
 "deletecollection", "annotate", "patch", "label"]
EOF
```

2. Création d'une liaison de rôle Kubernetes adaptée à l'espace de noms

   Exécutez la commande suivante pour créer une liaison de rôle Kubernetes dans l'espace de noms donné. Cette liaison de rôle accorde les autorisations définies dans le rôle créé à l'étape précédente à un utilisateur nommé `emr-containers`. Cet utilisateur identifie les rôles liés au service pour Amazon EMR on EKS et permet ainsi à Amazon EMR on EKS d'effectuer les actions définies par le rôle que vous avez créé.

   ```
   namespace=my-namespace

   cat - <<EOF | kubectl apply -f - --namespace "${namespace}"
   apiVersion: rbac.authorization.k8s.io/v1
   kind: RoleBinding
   metadata:
     name: emr-containers
     namespace: ${namespace}
   subjects:
   - kind: User
     name: emr-containers
     apiGroup: rbac.authorization.k8s.io
   roleRef:
     kind: Role
     name: emr-containers
     apiGroup: rbac.authorization.k8s.io
   EOF
   ```

3. Mise à jour la carte de configuration **`aws-auth`** de Kubernetes

   Vous pouvez utiliser l'une des options suivantes pour associer le rôle lié au service Amazon EMR on EKS à l'utilisateur `emr-containers` associé au rôle Kubernetes à l'étape précédente.

Option 1 : utilisation de l'outil **`eksctl`**

Exécutez la commande `eksctl` suivante pour associer le rôle lié au service Amazon EMR on EKS à l'utilisateur `emr-containers`.

```
eksctl create iamidentitymapping \
    --cluster my-cluster-name \
    --arn "arn:aws:iam::my-account-id:role/AWSServiceRoleForAmazonEMRContainers" \
    --username emr-containers
```

Option 2 : sans utiliser eksctl

1. Exécutez la commande suivante pour ouvrir la carte de configuration `aws-auth` dans l'éditeur de texte.

   ```
   kubectl edit -n kube-system configmap/aws-auth
   ```

   > **Note**
   >
   > Si vous recevez un message d'erreur`Error from server (NotFound): configmaps "aws-auth" not found`, consultez les étapes décrites dans la section Ajouter des rôles d'utilisateur dans le guide de l'utilisateur Amazon EKS pour appliquer le stock ConfigMap.

2. Ajoutez les détails du rôle lié au service Amazon EMR on EKS dans la section `mapRoles` de la `ConfigMap`, sous `data`. Ajoutez cette section si elle n'existe pas déjà dans le fichier. La section `mapRoles` mise à jour sous les données ressemble à l'exemple suivant.

   ```
   apiVersion: v1
   data:
     mapRoles: |
       - rolearn: arn:aws:iam::<your-account-id>:role/
   AWSServiceRoleForAmazonEMRContainers
         username: emr-containers
       - ... <other previously existing role entries, if there's any>.
   ```

3. Enregistrez le fichier et quittez votre éditeur de texte.

Activer les rôles IAM pour le cluster EKS

Les rubriques suivantes décrivent les options permettant d'activer les rôles IAM.

Rubriques

Option 1 : activer l'identité du pod EKS sur le cluster 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.

Amazon EMR on EKS commence à prendre en charge l'identité des pods EKS depuis la version emr-7.3.0 pour le modèle de soumission. StartJobRun

Pour plus d'informations sur les identités des pods EKS, reportez-vous à la section Comprendre le fonctionnement d'EKS Pod Identity.

Pourquoi choisir EKS Pod Identities ?

Dans le cadre de la configuration d'EMR, le rôle Job Execution doit établir des limites de confiance entre un rôle IAM et des comptes de service dans un espace de noms spécifique (des clusters virtuels EMR). Avec l'IRSA, cela a été réalisé en mettant à jour la politique de confiance du rôle EMR Job Execution. Cependant, en raison de la limite stricte de 4 096 caractères imposée à la politique de confiance IAM, il était nécessaire de partager un seul rôle IAM d'exécution de tâches sur un maximum de douze (12) clusters EKS.

Grâce au support d'EMR pour Pod Identities, la limite de confiance entre les rôles IAM et les comptes de service est désormais gérée par l'équipe EKS via l'association d'EKS pod identity. APIs

> **Note**
>
> La limite de sécurité pour l'identité du pod EKS est toujours au niveau du compte de service, et non au niveau du pod.

Considérations relatives à l'identité des

Pour plus d'informations sur les limites relatives à l'identité des pods, consultez la section Considérations relatives à l'identité des pods d'EKS.

Préparer l'identité du pod EKS dans le cluster EKS

Vérifiez si l'autorisation requise existe dans NodeInstanceRole

Le rôle de nœud `NodeInstanceRole` nécessite une autorisation pour que l'agent puisse effectuer l'`AssumeRoleForPodIdentity`action dans l'API EKS Auth. Vous pouvez ajouter les éléments suivants à l'Amazon EKSWorker NodePolicy, qui est défini dans le guide de l'utilisateur Amazon EKS, ou utiliser une politique personnalisée.

Si votre cluster EKS a été créé avec une version eksctl supérieure à 0.181.0, Amazon EKSWorkerNodePolicy, y compris les `AssumeRoleForPodIdentity` autorisations requises, sera automatiquement attaché au rôle de nœud. Si l'autorisation n'est pas présente, ajoutez manuellement l'autorisation suivante à Amazon, EKSWorker NodePolicy qui permet d'assumer un rôle dans l'identité du pod. L'agent d'identité des pods EKS a besoin de cette autorisation pour récupérer les informations d'identification des pods.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "eks-auth:AssumeRoleForPodIdentity"
      ],
      "Resource": [
        "*"
      ],
      "Sid": "AllowEKSAUTHAssumeroleforpodidentity"
    }
  ]
}
```

Créer un module complémentaire pour l'agent d'identité EKS pod

Utilisez la commande suivante pour créer le module complémentaire EKS Pod Identity Agent avec la dernière version :

```
aws eks create-addon --cluster-name cluster-name --addon-name eks-pod-identity-agent

kubectl get pods -n kube-system | grep 'eks-pod-identity-agent'
```

Suivez les étapes suivantes pour créer le module complémentaire EKS Pod Identity Agent à partir de la console Amazon EKS :

1. Ouvrez la console Amazon EKS : 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 des 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. Vérifiez que les pods d'agent s'exécutent sur votre cluster via la CLI.

   ```
   kubectl get pods -n kube-system | grep 'eks-pod-identity-agent'
   ```

Voici un exemple de sortie :

```
NAME                              READY   STATUS    RESTARTS      AGE
eks-pod-identity-agent-gmqp7      1/1     Running   1 (24h ago)   24h
```

```
eks-pod-identity-agent-prnsh      1/1     Running   1 (24h ago)   24h
```

Cela crée un nouveau nom DaemonSet dans l'espace de `kube-system` noms. L'agent Amazon EKS Pod Identity, qui s'exécute sur chaque nœud EKS, utilise cette AssumeRoleForPodIdentityaction pour récupérer des informations d'identification temporaires à partir de l'API EKS Auth. Ces informations d'identification sont ensuite mises à disposition pour AWS SDKs celles que vous exécutez dans vos conteneurs.

Pour plus d'informations, consultez la condition préalable dans le document public : Configurer l'agent d'identité Amazon EKS Pod.

Création d'un rôle d'exécution de Job

Création ou mise à jour du rôle d'exécution des tâches qui autorise EKS Pod Identity

Pour exécuter des charges de travail avec Amazon EMR sur EKS, vous devez créer un rôle IAM. Dans cette documentation, nous appelons ce rôle le rôle d'exécution des tâches. Pour plus d'informations sur la création du rôle IAM, consultez la section Création de rôles IAM dans le guide de l'utilisateur.

En outre, vous devez créer une politique IAM qui spécifie les autorisations nécessaires pour le rôle d'exécution de la tâche, puis associer cette politique au rôle pour activer EKS Pod Identity.

Par exemple, vous avez le rôle d'exécution de tâches suivant. Pour plus d'informations, consultez la section Création d'un rôle d'exécution de tâche.

```
arn:aws:iam::111122223333:role/PodIdentityJobExecutionRole
```

> **Important**
>
> Amazon EMR on EKS crée automatiquement des comptes de service Kubernetes, en fonction du nom de votre rôle d'exécution de tâches. Assurez-vous que le nom du rôle n'est pas trop long, car votre tâche risque d'échouer si la combinaison de `cluster_namepod_name`, et `service_account_name` dépasse la limite de longueur.

Configuration du rôle d'exécution des tâches : assurez-vous que le rôle d'exécution des tâches est créé avec l'autorisation de confiance ci-dessous pour EKS Pod Identity. Pour mettre à jour un rôle d'exécution de tâche existant, configurez-le pour qu'il approuve le principal de service EKS suivant

en tant qu'autorisation supplémentaire dans la politique de confiance. Cette autorisation de confiance peut coexister avec les politiques de confiance existantes de l'IRSA.

```
cat >trust-relationship.json <<EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowEksAuthToAssumeRoleForPodIdentity",
            "Effect": "Allow",
            "Principal": {
                "Service": "pods.eks.amazonaws.com"
            },
            "Action": [
                "sts:AssumeRole",
                "sts:TagSession"
            ]
        }
    ]
}
EOF
```

Autorisation utilisateur : les utilisateurs doivent être `iam:PassRole` autorisés à exécuter des appels d'`StartJobRun`API ou à soumettre des tâches. Cette autorisation permet aux utilisateurs de transmettre le rôle d'exécution des tâches à EMR sur EKS. Les administrateurs de tâches doivent avoir l'autorisation par défaut.

Vous trouverez ci-dessous l'autorisation requise pour un utilisateur :

```
{
    "Effect": "Allow",
    "Action": "iam:PassRole",
    "Resource": "arn:aws:iam::111122223333:role/PodIdentityJobExecutionRole",
    "Condition": {
        "StringEquals": {
            "iam:PassedToService": "pods.eks.amazonaws.com"
        }
    }
}
```

Pour restreindre davantage l'accès des utilisateurs à des clusters EKS spécifiques, ajoutez le filtre AssociatedResourceArn d'attributs à la politique IAM. Il limite l'attribution des rôles aux clusters EKS autorisés, renforçant ainsi vos contrôles de sécurité au niveau des ressources.

```
"Condition": {
        "ArnLike": {
            "iam:AssociatedResourceARN": [
                "arn:aws:eks:us-west-2:111122223333:cluster/*"
            ]
        }
```

Configurer les associations d'identité des pods EKS

Prérequis

Assurez-vous que l'identité IAM qui crée l'association d'identité du pod, par exemple un utilisateur administrateur EKS, dispose de l'autorisation `eks:CreatePodIdentityAssociation` et`iam:PassRole`.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "eks:CreatePodIdentityAssociation"
      ],
      "Resource": [
        "arn:aws:eks:*:*:cluster/*"
      ],
      "Sid": "AllowEKSCreatepodidentityassociation"
    },
    {
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": [
        "arn:aws:iam::*:role/*"
```

```
      ],
      "Condition": {
        "StringEquals": {
          "iam:PassedToService": "pods.eks.amazonaws.com"
        }
      },
      "Sid": "AllowIAMPassrole"
    }
  ]
}
```

Créer des associations pour le rôle et le compte de service EMR

Create EMR role associations through the AWS CLI

Lorsque vous soumettez une tâche à un espace de noms Kubernetes, un administrateur doit créer des associations entre le rôle d'exécution de la tâche et l'identité du compte de service géré EMR. Notez que le compte de service géré EMR est automatiquement créé lors de la soumission de la tâche, dans la limite de l'espace de noms dans lequel la tâche est soumise.

Avec la version 2.24.0 AWS CLI (supérieure), exécutez la commande suivante pour créer des associations de rôles avec l'identité du pod.

Exécutez la commande suivante pour créer des associations de rôles avec pod identity :

```
aws emr-containers create-role-associations \
        --cluster-name mycluster \
        --namespace mynamespace \
        --role-name JobExecutionRoleIRSAv2
```

Remarque :

- Chaque cluster peut avoir une limite de 1 000 associations. Le mappage des rôles et des espaces de noms de chaque tâche nécessitera 3 associations pour les modules émetteur de tâches, pilote et exécuteur.
- Vous ne pouvez associer que des rôles appartenant au même AWS compte que le cluster. Vous pouvez déléguer l'accès d'un autre compte au rôle de ce compte que vous configurez pour que les identités du pod EKS utilisent. Pour un didacticiel sur la délégation d'accès`AssumeRole`, voir Tutoriel IAM : déléguer l'accès entre AWS comptes à l'aide de rôles IAM.

Create EMR role associations through Amazon EKS

EMR crée un compte de service avec un certain modèle de dénomination lorsqu'une tâche est soumise. Pour créer des associations manuelles ou intégrer ce flux de travail au AWS SDK, procédez comme suit :

Nom du compte de service de construction :

```
emr-containers-sa-spark-%(SPARK_ROLE)s-%(AWS_ACCOUNT_ID)s-
%(BASE36_ENCODED_ROLE_NAME)s
```

Les exemples ci-dessous créent une association de rôles pour un exemple de rôle d'exécution de Job JobExecutionRoleIRSAv2.

Exemples d'associations de rôles :

```
RoleName: JobExecutionRoleIRSAv2
Base36EncodingOfRoleName: 2eum5fah1jc1kwyjc19ikdhdkdegh1n26vbe
```

Exemple de commande CLI :

```
# setup for the client service account (used by job runner pod)
# emr-containers-sa-spark-client-111122223333-2eum5fah1jc1kwyjc19ikdhdkdegh1n26vbe
aws eks create-pod-identity-association --cluster-name mycluster
 --role-arn arn:aws:iam::111122223333:role/JobExecutionRoleIRSAv2
 --namespace mynamespace --service-account emr-containers-sa-spark-
client-111122223333-2eum5fah1jc1kwyjc19ikdhdkdegh1n26vbe

# driver service account
# emr-containers-sa-spark-driver-111122223333-2eum5fah1jc1kwyjc19ikdhdkdegh1n26vbe

aws eks create-pod-identity-association --cluster-name mycluster
 --role-arn arn:aws:iam::111122223333:role/JobExecutionRoleIRSAv2
 --namespace mynamespace --service-account emr-containers-sa-spark-
driver-111122223333-2eum5fah1jc1kwyjc19ikdhdkdegh1n26vbe

# executor service account
# emr-containers-sa-spark-executor-111122223333-2eum5fah1jc1kwyjc19ikdhdkdegh1n26vbe
aws eks create-pod-identity-association --cluster-name mycluster
 --role-arn arn:aws:iam::111122223333:role/JobExecutionRoleIRSAv2
 --namespace mynamespace --service-account emr-containers-sa-spark-
executor-111122223333-2eum5fah1jc1kwyjc19ikdhdkdegh1n26vbe
```

Une fois que vous avez effectué toutes les étapes requises pour l'identification du pod EKS, vous pouvez ignorer les étapes suivantes pour la configuration de l'IRSA :

- Activer les rôles IAM pour les comptes de service (IRSA) sur le cluster EKS
- Création d'un rôle d'exécution de tâches
- Mettre à jour la politique de confiance du rôle d'exécution des tâches

Vous pouvez passer directement à l'étape suivante : Accorder aux utilisateurs l'accès à Amazon EMR sur EKS

Supprimer les associations de rôles

Chaque fois que vous supprimez un cluster virtuel ou un rôle d'exécution de tâches et que vous ne souhaitez plus donner accès à EMR à ses comptes de service, vous devez supprimer les associations associées au rôle. Cela est dû au fait qu'EKS autorise les associations avec des ressources inexistantes (espace de noms et compte de service). Amazon EMR on EKS recommande de supprimer les associations si l'espace de noms est supprimé ou si le rôle n'est plus utilisé, afin de libérer de l'espace pour d'autres associations.

> **Note**
>
> Les associations persistantes peuvent avoir un impact sur votre capacité à évoluer si vous ne les supprimez pas, car EKS limite le nombre d'associations que vous pouvez créer (limite souple : 1 000 associations par cluster). Vous pouvez répertorier les associations d'identité des pods dans un espace de noms donné pour vérifier s'il existe des associations persistantes qui doivent être nettoyées :

```
aws eks list-pod-identity-associations --cluster-name mycluster --namespace mynamespace
```

Avec la AWS CLI version 2.24.0 ou supérieure, exécutez la commande emr-containers suivante pour supprimer les associations de rôles d'EMR :

```
aws emr-containers delete-role-associations \
        --cluster-name mycluster \
        --namespace mynamespace \
        --role-name JobExecutionRoleIRSAv2
```

Migrer automatiquement l'IRSA existant vers Pod Identity

Vous pouvez utiliser l'outil eksctl pour migrer les rôles IAM existants pour les comptes de service (IRSA) vers des associations d'identité de pod :

```
eksctl utils migrate-to-pod-identity \
    --cluster mycluster \
    --remove-oidc-provider-trust-relationship \
    --approve
```

L'exécution de la commande sans l'`--approve`indicateur produira uniquement un plan reflétant les étapes de migration, et aucune migration réelle n'aura lieu.

Résolution des problèmes

Ma tâche a échoué avec NoClassDefinitionFound ou ClassNotFound Exception pour le fournisseur d'informations d'identification, ou je n'ai pas réussi à obtenir le fournisseur d'informations d'identification.

EKS Pod Identity utilise le fournisseur d'informations d'identification du conteneur pour récupérer les informations d'identification nécessaires. Si vous avez spécifié un fournisseur d'identifiants personnalisés, assurez-vous qu'il fonctionne correctement. Vous pouvez également vous assurer que vous utilisez une version du AWS SDK correcte qui prend en charge l'EKS Pod Identity. Pour plus d'informations, consultez la section Commencer avec Amazon EKS.

La tâche a échoué en raison de l'erreur « Impossible de récupérer les informations d'identification en raison de la limite de taille [x] » affichée dans le eks-pod-identity-agent journal.

EMR sur EKS crée des comptes de service Kubernetes en fonction du nom du rôle d'exécution de la tâche. Si le nom du rôle est trop long, EKS Auth ne parviendra pas à récupérer les informations d'identification car la combinaison de `cluster_namepod_name`, et `service_account_name` dépasse la limite de longueur. Identifiez le composant qui occupe le plus d'espace et ajustez la taille en conséquence.

La tâche a échoué avec l'erreur « Failed to Retrieve Credentials xxx » affichée dans le eks-pod-identity journal.

Ce problème peut être dû au fait que le cluster EKS est configuré sous des sous-réseaux privés sans être correctement configuré PrivateLink pour le cluster. Vérifiez si votre cluster se trouve dans un réseau privé et configurez-le AWS PrivateLink pour résoudre le problème. Pour obtenir des instructions détaillées, reportez-vous à la section Commencer avec Amazon EKS.

Option 2 : activer les rôles IAM pour les comptes de service (IRSA) sur le cluster EKS

La fonctionnalité des rôles IAM pour les comptes de service est disponible sur Amazon EKS en version 1.14 ou ultérieure et pour les clusters EKS mis à jour vers la version 1.13 ou ultérieure à partir du 3 septembre 2019. Pour utiliser cette fonctionnalité, vous pouvez mettre à jour les clusters EKS existants vers la version 1.14 ou une version ultérieure. Pour plus d'informations, consultez la rubrique Mise à jour de la version Kubernetes d'un cluster Amazon EKS.

Si votre cluster prend en charge les rôles IAM pour les comptes de service, une URL d'émetteur OpenID Connect lui est associée. Vous pouvez consulter cette URL dans la console Amazon EKS ou utiliser la AWS CLI commande suivante pour la récupérer.

> **Important**
>
> Vous devez utiliser la dernière version de AWS CLI pour obtenir le résultat approprié de cette commande.

```
aws eks describe-cluster --name cluster_name --query "cluster.identity.oidc.issuer" --
output text
```

Le résultat attendu est le suivant.

```
https://oidc.eks.<region-code>.amazonaws.com/id/EXAMPLED539D4633E53DE1B716D3041E
```

Pour utiliser les rôles IAM pour les comptes de service dans votre cluster, vous devez créer un fournisseur d'identité OIDC à l'aide d'eksctl ou de la AWS Management Console.

Pour créer un fournisseur d'identité OIDC IAM pour votre cluster avec **`eksctl`**

Vous pouvez vérifier la version de votre `eksctl` à l'aide de la commande suivante. Cette procédure suppose que vous avez installé `eksctl` et que votre version `eksctl` est 0.32.0 ou une version ultérieure.

```
eksctl version
```

Pour plus d'informations sur l'installation ou la mise à niveau d'eksctl, consultez la rubrique Installation ou mise à niveau d'eksctl.

Créez votre fournisseur d'identité OIDC pour votre cluster avec la commande suivante. Remplacez *cluster_name* par votre propre valeur.

```
eksctl utils associate-iam-oidc-provider --cluster cluster_name --approve
```

Pour créer un fournisseur d'identité IAM OIDC pour votre cluster à l'aide du AWS Management Console

Récupérez l'URL de l'émetteur OIDC dans la description de votre cluster sur la console Amazon EKS, ou utilisez la commande suivante AWS CLI .

Utilisez la commande suivante pour récupérer l'URL de l'émetteur OIDC à partir de la AWS CLI.

```
aws eks describe-cluster --name <cluster_name> --query "cluster.identity.oidc.issuer"
 --output text
```

Suivez les étapes ci-dessous pour récupérer l'URL de l'émetteur OIDC à partir de la console Amazon EKS.

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Dans le volet de navigation, choisissez Fournisseurs d'identité, puis Créer un fournisseur.
    1. Sous Provider Type (Type de fournisseur), choisissez Choose a provider type (Choisir un type de fournisseur), puis choisissez OpenID Connect.
    2. Pour Provider URL (URL du fournisseur, collez l'URL de l'émetteur OIDC pour votre cluster.
    3. Pour Public ciblé, saisissez sts.amazonaws.com et choisissez Étape suivante.
3. Vérifiez que les informations du fournisseur sont correctes, puis choisissez Créer (Create) pour créer votre fournisseur d'identité.

Création d'un rôle d'exécution des tâches

Pour exécuter des charges de travail sur Amazon EMR on EKS, vous devez créer un rôle IAM. Dans cette documentation, nous appelons ce rôle le rôle d'exécution des tâches. Pour plus d'informations sur la création de rôles IAM, consultez Création de rôles IAM dans le Guide de l'utilisateur IAM.

Vous devez également créer une politique IAM qui spécifie les autorisations pour le rôle d'exécution des tâches, puis associer la politique IAM au rôle d'exécution des tâches.

La politique suivante pour le rôle d'exécution des tâches autorise l'accès aux cibles de ressources, Amazon S3 et CloudWatch. Ces autorisations sont nécessaires pour surveiller les tâches et les journaux d'accès. Pour suivre le même processus en utilisant AWS CLI :

Créer un rôle IAM pour l'exécution des tâches : créons le rôle qu'EMR utilisera pour l'exécution des tâches. C'est le rôle que les tâches EMR assumeront lorsqu'elles s'exécuteront sur EKS.

```
cat <<EoF > ~/environment/emr-trust-policy.json
 {
   "Version": "2012-10-17",
   "Statement": [
     {
       "Effect": "Allow",
       "Principal": {
         "Service": "elasticmapreduce.amazonaws.com"
       },
       "Action": "sts:AssumeRole"
     }
   ]
 }
 EoF

 aws iam create-role --role-name EMRContainers-JobExecutionRole --assume-role-policy-
document file://~/environment/emr-trust-policy.json
```

Ensuite, nous devons associer les politiques IAM requises au rôle afin qu'il puisse écrire des journaux sur s3 et Cloudwatch.

```
cat <<EoF > ~/environment/EMRContainers-JobExecutionRole.json
 {
     "Version": "2012-10-17",
     "Statement": [
         {
             "Effect": "Allow",
             "Action": [
                 "s3:PutObject",
                 "s3:GetObject",
                 "s3:ListBucket"
             ],
             "Resource": "arn:aws:s3:::amzn-s3-demo-bucket"
         },
```

```
        {
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents",
                "logs:CreateLogStream",
              "logs:DescribeLogGroups",
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:*:*:*"
            ]
        }
    ]
}
EoF
aws iam put-role-policy --role-name EMRContainers-JobExecutionRole --policy-name
EMR-Containers-Job-Execution --policy-document file://~/environment/EMRContainers-
JobExecutionRole.json
```

> **Note**
>
> L'accès doit être défini de manière appropriée et ne pas être accordé à tous les objets S3 du rôle d'exécution des tâches.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket"
      ],
      "Sid": "AllowS3Putobject"
```

```
    },
    {
      "Effect": "Allow",
      "Action": [
        "logs:PutLogEvents",
        "logs:CreateLogStream",
        "logs:DescribeLogGroups",
        "logs:DescribeLogStreams"
      ],
      "Resource": [
        "arn:aws:logs:*:*:*"
      ],
      "Sid": "AllowLOGSPutlogevents"
    }
  ]
}
```

Pour plus d'informations, consultez les sections Utilisation des rôles d'exécution de tâches, Configuration d'une exécution de tâche pour utiliser les journaux S3 et Configuration d'une exécution de tâche pour utiliser CloudWatch les journaux.

Mise à jour la politique d'approbation du rôle d'exécution des tâches

Lorsque vous utilisez les rôles IAM pour les comptes de service (IRSA) pour exécuter des tâches sur un espace de noms Kubernetes, un administrateur doit créer une relation d'approbation entre le rôle d'exécution des tâches et l'identité du compte de service géré EMR. La relation d'approbation peut être créée en mettant à jour la politique d'approbation du rôle d'exécution des tâches. Notez que le compte de service géré EMR est automatiquement créé lors de la soumission de la tâche, dans la limite de l'espace de noms dans lequel la tâche est soumise.

Pour mettre à jour la politique d'approbation, exécutez la commande suivante.

```
aws emr-containers update-role-trust-policy \
      --cluster-name cluster \
      --namespace namespace \
      --role-name iam_role_name_for_job_execution
```

Pour de plus amples informations, veuillez consulter Utilisation des rôles d'exécution de tâches avec Amazon EMR on EKS.

> **Important**
>
> L'opérateur exécutant la commande ci-dessus doit disposer des autorisations suivantes : `eks:DescribeCluster`, `iam:GetRole`, `iam:UpdateAssumeRolePolicy`.

Autorisation des utilisateurs à accéder à Amazon EMR on EKS

Pour toute action que vous effectuez sur Amazon EMR on EKS, vous avez besoin d'une autorisation IAM correspondant à cette action. Vous devez créer une politique IAM qui vous permet d'exécuter les actions Amazon EMR on EKS et l'attacher à l'utilisateur IAM ou au rôle que vous utilisez.

Cette rubrique décrit les étapes à suivre pour créer une nouvelle politique et l'associer à un utilisateur. Elle couvre également les autorisations de base dont vous avez besoin pour configurer votre environnement Amazon EMR on EKS. Nous vous recommandons d'affiner les autorisations relatives à des ressources spécifiques dans la mesure du possible en fonction des besoins de votre entreprise.

Création d'une nouvelle politique IAM et association de celle-ci à un utilisateur dans la console IAM

Création d'une nouvelle politique IAM

1. Connectez-vous à la console IAM AWS Management Console et ouvrez-la à https://console.aws.amazon.com/iam/l'adresse.
2. Dans le volet de navigation gauche de la console IAM, choisissez Politiques.
3. Sur la page Politiques, choisissez Créer une politique.
4. Dans la fenêtre Créer une politique, accédez à l'onglet Modifier JSON. Créez un document de politique avec une ou plusieurs instructions JSON, comme indiqué dans les exemples suivant cette procédure. Ensuite, choisissez Examiner la politique.
5. Sur l'écran Review policy (Examiner la politique), saisissez votre Policy Name (Nom de politique), par exemple, `AmazonEMROnEKSPolicy`. Saisissez une description facultative, puis sélectionnez Créer une politique.

Attacher la politique à un utilisateur ou à un rôle

1. Connectez-vous à la console IAM AWS Management Console et ouvrez-la à https://console.aws.amazon.com/iam/

2. Dans le panneau de navigation, choisissez Politiques.
3. Dans la liste des politiques, cochez la case en regard de la politique créée dans la section précédente. Vous pouvez utiliser le menu Filtre et la zone de recherche pour filtrer la liste de politiques.
4. Sélectionnez Policy Actions (Actions de politique), puis sélectionnez Attach (Attacher).
5. Choisissez l'utilisateur ou le rôle auquel attacher la politique. Vous pouvez utiliser le menu Filtre et la zone de recherche pour filtrer la liste des entités du principal. Après avoir choisi l'utilisateur auquel attacher la politique, sélectionnez Attacher la politique.

Autorisations pour la gestion des clusters virtuels

Pour gérer les clusters virtuels dans votre AWS compte, créez une politique IAM avec les autorisations suivantes. Ces autorisations vous permettent de créer, de répertorier, de décrire et de supprimer des clusters virtuels dans votre AWS compte.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iam:CreateServiceLinkedRole"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "StringLike": {
          "iam:AWSServiceName": "emr-containers.amazonaws.com"
        }
      },
      "Sid": "AllowIAMCreateservicelinkedrole"
    },
    {
      "Effect": "Allow",
      "Action": [
        "emr-containers:CreateVirtualCluster",
        "emr-containers:ListVirtualClusters",
```

```
        "emr-containers:DescribeVirtualCluster",
        "emr-containers:DeleteVirtualCluster"
      ],
      "Resource": [
        "*"
      ],
      "Sid": "AllowEMRCONTAINERSCreatevirtualcluster"
    }
  ]
}
```

Amazon EMR est intégré à la gestion des accès aux clusters (CAM) d'Amazon EKS, ce qui vous permet d'automatiser la configuration des politiques AuthN et AuthZ nécessaires pour exécuter des tâches Amazon EMR Spark dans les espaces de noms des clusters Amazon EKS. Pour ce faire, vous devez disposer des autorisations suivantes :

```
{
  "Effect": "Allow",
  "Action": [
    "eks:CreateAccessEntry"
  ],
  "Resource":
 "arn:<AWS_PARTITION>:eks:<AWS_REGION>:<AWS_ACCOUNT_ID>:cluster/<EKS_CLUSTER_NAME>"
},
{
  "Effect": "Allow",
  "Action": [
    "eks:DescribeAccessEntry",
    "eks:DeleteAccessEntry",
    "eks:ListAssociatedAccessPolicies",
    "eks:AssociateAccessPolicy",
    "eks:DisassociateAccessPolicy"
  ],
  "Resource": "arn:<AWS_PARTITION>:eks:<AWS_REGION>:<AWS_ACCOUNT_ID>:access-
entry/<EKS_CLUSTER_NAME>/role/<AWS_ACCOUNT_ID>/AWSServiceRoleForAmazonEMRContainers/*"
}
```

Pour plus d'informations, consultez Automatiser l'activation de l'accès au cluster pour Amazon EMR sur EKS.

Lorsque l'`CreateVirtualCluster`opération est invoquée pour la première fois depuis un AWS compte, vous devez également disposer des `CreateServiceLinkedRole` autorisations nécessaires pour créer le rôle lié à un service pour Amazon EMR sur EKS. Pour de plus amples informations, veuillez consulter Utilisation des rôles liés à un service pour Amazon EMR on EKS.

Autorisations pour la soumission de tâches

Pour soumettre des tâches sur les clusters virtuels de votre AWS compte, créez une politique IAM avec les autorisations suivantes. Ces autorisations vous permettent de démarrer, de répertorier, de décrire et d'annuler des exécutions de tâches pour tous les clusters virtuels de votre compte. Vous devriez envisager d'ajouter des autorisations pour répertorier ou décrire les clusters virtuels, ce qui vous permet de vérifier l'état du cluster virtuel avant de soumettre des tâches.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "emr-containers:StartJobRun",
        "emr-containers:ListJobRuns",
        "emr-containers:DescribeJobRun",
        "emr-containers:CancelJobRun"
      ],
      "Resource": [
        "*"
      ],
      "Sid": "AllowEMRCONTAINERSStartjobrun"
    }
  ]
}
```

Autorisations pour le débogage et la surveillance

Pour accéder aux journaux transmis à Amazon S3 et/ou pour consulter les CloudWatch journaux des événements de l'application dans la console Amazon EMR, créez une politique IAM avec les autorisations suivantes. Nous vous recommandons d'affiner les autorisations relatives à des ressources spécifiques dans la mesure du possible en fonction des besoins de votre entreprise.

> **Important**
>
> Si vous n'avez pas créé de compartiment Amazon S3, vous devez ajouter une autorisation `s3:CreateBucket` à la déclaration de politique. Si vous n'avez pas créé de groupe de journaux, vous devez ajouter `logs:CreateLogGroup` à la déclaration de politique.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "emr-containers:DescribeJobRun",
        "elasticmapreduce:CreatePersistentAppUI",
        "elasticmapreduce:DescribePersistentAppUI",
        "elasticmapreduce:GetPersistentAppUIPresignedURL"
      ],
      "Resource": [
        "*"
      ],
      "Sid": "AllowEMRCONTAINERSDescribejobrun"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": [
        "*"
      ],
      "Sid": "AllowS3Getobject"
    },
    {
      "Effect": "Allow",
      "Action": [
        "logs:Get*",
```

```
        "logs:DescribeLogGroups",
        "logs:DescribeLogStreams"
      ],
      "Resource": [
        "*"
      ],
      "Sid": "AllowLOGSGet"
    }
  ]
}
```

Pour plus d'informations sur la façon de configurer l'exécution d'une tâche pour envoyer des journaux vers Amazon S3 CloudWatch, consultez Configurer une exécution de tâche pour utiliser les journaux S3 et Configurer une exécution de tâche pour utiliser CloudWatch les journaux.

Enregistrement du cluster Amazon EKS dans Amazon EMR

L'enregistrement de votre cluster est la dernière étape nécessaire pour configurer Amazon EMR on EKS afin d'exécuter des charges de travail.

Utilisez la commande suivante pour créer un cluster virtuel portant le nom de votre choix pour le cluster Amazon EKS et l'espace de noms que vous avez configurés aux étapes précédentes.

> **Note**
>
> Chaque cluster virtuel doit avoir un nom unique pour tous les clusters EKS. Si deux clusters virtuels portent le même nom, le processus de déploiement échouera même s'ils appartiennent à des clusters EKS différents.

```
aws emr-containers create-virtual-cluster \
--name virtual_cluster_name \
--container-provider '{
    "id": "cluster_name",
    "type": "EKS",
    "info": {
        "eksInfo": {
            "namespace": "namespace_name"
        }
    }
```

```
}'
```

Vous pouvez également créer un fichier JSON qui inclut les paramètres requis pour le cluster virtuel, puis exécuter la commande `create-virtual-cluster` avec le chemin d'accès au fichier JSON. Pour de plus amples informations, veuillez consulter Gestion des clusters virtuels.

> **Note**
>
> Pour valider la création réussie d'un cluster virtuel, consultez l'état des clusters virtuels à l'aide de l'opération `list-virtual-clusters` ou en accédant à la page Clusters virtuels de la console Amazon EMR.

Soumission d'une tâche exécutée avec **`StartJobRun`**

Soumission d'une tâche exécutée à l'aide d'un fichier JSON avec les paramètres spécifiés

1. Créez un fichier `start-job-run-request.json` et indiquez les paramètres requis pour l'exécution de votre tâche, comme le montre l'exemple de fichier JSON ci-dessous. Pour plus d'informations sur les paramètres, consultez Options de configuration d'une exécution de tâche.

```
{
  "name": "myjob",
  "virtualClusterId": "123456",
  "executionRoleArn": "iam_role_name_for_job_execution",
  "releaseLabel": "emr-6.2.0-latest",
  "jobDriver": {
    "sparkSubmitJobDriver": {
      "entryPoint": "entryPoint_location",
      "entryPointArguments": ["argument1", "argument2", ...],
       "sparkSubmitParameters": "--class <main_class> --conf
 spark.executor.instances=2 --conf spark.executor.memory=2G --conf
 spark.executor.cores=2 --conf spark.driver.cores=1"
    }
  },
  "configurationOverrides": {
    "applicationConfiguration": [
      {
        "classification": "spark-defaults",
        "properties": {
          "spark.driver.memory":"2G"
```

```
        }
      }
    ],
    "monitoringConfiguration": {
      "persistentAppUI": "ENABLED",
      "cloudWatchMonitoringConfiguration": {
        "logGroupName": "my_log_group",
        "logStreamNamePrefix": "log_stream_prefix"
      },
      "s3MonitoringConfiguration": {
        "logUri": "s3://my_s3_log_location"
      }
    }
  }
}
```

2. Utilisez la commande `start-job-run` avec un chemin d'accès au fichier `start-job-run-request.json` stocké localement.

```
aws emr-containers start-job-run \
--cli-input-json file://./start-job-run-request.json
```

Démarrage de l'exécution d'une tâche à l'aide la commande **`start-job-run`**

1. Fournissez tous les paramètres spécifiés dans la commande `StartJobRun`, comme le montre l'exemple ci-dessous.

```
aws emr-containers start-job-run \
--virtual-cluster-id 123456 \
--name myjob \
--execution-role-arn execution-role-arn \
--release-label emr-6.2.0-latest \
--job-driver '{"sparkSubmitJobDriver": {"entryPoint": "entryPoint_location",
 "entryPointArguments": ["argument1", "argument2", ...], "sparkSubmitParameters":
 "--class <main_class> --conf spark.executor.instances=2 --conf
 spark.executor.memory=2G --conf spark.executor.cores=2 --conf
 spark.driver.cores=1"}}' \
--configuration-overrides '{"applicationConfiguration": [{"classification":
 "spark-defaults", "properties": {"spark.driver.memory": "2G"}}],
 "monitoringConfiguration": {"cloudWatchMonitoringConfiguration":
 {"logGroupName": "log_group_name", "logStreamNamePrefix": "log_stream_prefix"},
```

```
 "persistentAppUI":"ENABLED",  "s3MonitoringConfiguration": {"logUri":
 "s3://my_s3_log_location" }}}'
```

2. Pour Spark SQL, fournissez tous les paramètres spécifiés dans la commande `StartJobRun`, comme le montre l'exemple ci-dessous.

```
aws emr-containers start-job-run \
--virtual-cluster-id 123456 \
--name myjob \
--execution-role-arn execution-role-arn \
--release-label emr-6.7.0-latest \
--job-driver '{"sparkSqlJobDriver": {"entryPoint": "entryPoint_location",
 "sparkSqlParameters": "--conf spark.executor.instances=2 --conf
 spark.executor.memory=2G --conf spark.executor.cores=2 --conf
 spark.driver.cores=1"}}' \
--configuration-overrides '{"applicationConfiguration": [{"classification":
 "spark-defaults", "properties": {"spark.driver.memory": "2G"}}],
 "monitoringConfiguration": {"cloudWatchMonitoringConfiguration":
 {"logGroupName": "log_group_name", "logStreamNamePrefix": "log_stream_prefix"},
 "persistentAppUI":"ENABLED",  "s3MonitoringConfiguration": {"logUri":
 "s3://my_s3_log_location" }}}'
```

Utilisation de la classification des soumissionnaires de tâches

Présentation de

La demande `StartJobRun` d'Amazon EMR on EKS crée un pod soumissionnaire de tâche (également connu sous le nom de pod exécuteur de tâche) pour lancer le pilote Spark. Vous pouvez utiliser la `emr-job-submitter` classification pour configurer les sélecteurs de nœuds, ajouter des tolérances, personnaliser la journalisation et apporter d'autres modifications au module d'envoi des tâches.

Les paramètres suivants sont disponibles dans la `emr-job-submitter` classification :

`jobsubmitter.node.selector.[selectorKey]`

S'ajoute au sélecteur de nœuds du module d'envoi de tâches, avec la clé *`selectorKey`* et la valeur comme valeur de configuration. Par exemple, vous pouvez `jobsubmitter.node.selector.identifier` définir sur `myIdentifier` et le module de soumission des tâches comportera un sélecteur de nœuds avec une clé `identifier` et

une valeur. `myIdentifier` Cela peut être utilisé pour spécifier sur quels nœuds le pod de soumission de tâches peut être placé. Pour ajouter plusieurs clés de sélection de nœuds, définissez plusieurs configurations avec ce préfixe.

`jobsubmitter.label.[labelKey]`

S'ajoute aux étiquettes du module d'envoi de tâches, avec la clé *`labelKey`* et la valeur comme valeur de configuration. Pour ajouter plusieurs étiquettes, définissez plusieurs configurations avec ce préfixe.

`jobsubmitter.annotation.[annotationKey]`

S'ajoute aux annotations du module d'envoi des tâches, avec la clé *`annotationKey`* et la valeur comme valeur de configuration. Pour ajouter plusieurs annotations, définissez plusieurs configurations avec ce préfixe.

`jobsubmitter.node.toleration.[tolerationKey]`

Ajoute des tolérances au module d'envoi des tâches. Par défaut, aucune tolérance n'est ajoutée au module. La clé de la tolérance sera *`tolerationKey`* et la valeur de la tolérance sera la valeur de configuration. Si la valeur de configuration est définie sur une chaîne non vide, l'opérateur le sera`Equals`. Si la valeur de configuration est définie sur`""`, l'opérateur le sera`Exists`.

`jobsubmitter.node.toleration.[tolerationKey].[effect]`

Ajoute un effet de tolérance au préfixe*`tolerationKey`*. Ce champ est obligatoire lors de l'ajout de tolérances. Les valeurs autorisées pour le champ d'effet sont `NoExecute`, `NoSchedule`, et`PreferNoSchedule`.

`jobsubmitter.node.toleration.[tolerationKey].[tolerationSeconds]`

Ajoute TolerationSeconds au préfixe. *`tolerationKey`* Champ facultatif. Applicable uniquement lorsque l'effet est présent`NoExecute`.

`jobsubmitter.scheduler.name`

Définit un SchedulerName personnalisé pour le module d'envoi des tâches.

`jobsubmitter.logging`

Active ou désactive la journalisation sur le module d'envoi des tâches. Lorsque ce paramètre est défini sur, `DISABLED` le conteneur de journalisation est supprimé du module d'envoi des tâches, ce qui désactive toute journalisation pour ce module spécifiée

dans le`monitoringConfiguration`, tel que `s3MonitoringConfiguration` ou. `cloudWatchMonitoringConfiguration` Lorsque ce paramètre n'est pas défini ou qu'il est défini sur une autre valeur, la connexion au module d'envoi des tâches est activée.

`jobsubmitter.logging.image`

Définit une image personnalisée à utiliser pour le conteneur de journalisation dans le module d'envoi des tâches.

`jobsubmitter.logging.request.cores`

Définit une valeur personnalisée pour le nombre CPUs, en unités de processeur, du conteneur de journalisation sur le module d'envoi des tâches. Par défaut, ce paramètre est réglé sur 100 m.

`jobsubmitter.logging.request.memory`

Définit une valeur personnalisée pour la quantité de mémoire, en octets, pour le conteneur de journalisation sur le module d'envoi des tâches. Par défaut, ce paramètre est réglé sur 200 Mi. Un mébioctet est une unité de mesure similaire à un mégaoctet.

`jobsubmitter.container.image`

Définit une image personnalisée pour le conteneur du module d'envoi de `job-runner` tâches.

`jobsubmitter.container.image.pullPolicy`

Définit imagePullPolicyles conteneurs du module de soumission des tâches.

Nous vous recommandons de placer les modules d'envoi de tâches sur des instances à la demande. Le fait de placer des modules d'envoi de tâches sur des instances Spot peut entraîner l'échec d'une tâche si l'instance sur laquelle s'exécute le module d'envoi de tâches est sujette à une interruption d'instance Spot. Vous pouvez également placer le module d'envoi des tâches dans une seule zone de disponibilité ou utiliser les étiquettes Kubernetes appliquées aux nœuds.

Exemples de classification des soumissionnaires de tâches

Dans cette section

- Demande StartJobRun avec placement de nœuds à la demande pour le pod soumissionnaire de tâches
- StartJobRundemande avec placement d'un nœud mono-AZ et placement du type d'instance Amazon EC2 pour le module d'envoi des tâches

- StartJobRundemande avec des étiquettes, des annotations et un planificateur personnalisé pour le module d'envoi des tâches
- StartJobRundemande avec une tolérance appliquée au module d'envoi de la tâche avec clédedicated, valeur graviton_machinesNoExecute, effet et un tolerationSeconds délai de 60 secondes
- StartJobRundemande avec journalisation désactivée pour le module d'envoi de tâches
- StartJobRundemande avec image du conteneur de journalisation personnalisé, processeur et mémoire pour le module d'envoi des tâches
- StartJobRundemande avec une image de conteneur d'envoi de tâches personnalisée et une politique d'extraction

Demande **`StartJobRun`** avec placement de nœuds à la demande pour le pod soumissionnaire de tâches

```
cat >spark-python-in-s3-nodeselector-job-submitter.json << EOF
{
  "name": "spark-python-in-s3-nodeselector",
  "virtualClusterId": "virtual-cluster-id",
  "executionRoleArn": "execution-role-arn",
  "releaseLabel": "emr-6.11.0-latest",
  "jobDriver": {
    "sparkSubmitJobDriver": {
      "entryPoint": "s3://S3-prefix/trip-count.py",
      "sparkSubmitParameters": "--conf spark.driver.cores=5  --conf
 spark.executor.memory=20G --conf spark.driver.memory=15G --conf
 spark.executor.cores=6"
    }
  },
  "configurationOverrides": {
    "applicationConfiguration": [
      {
        "classification": "spark-defaults",
        "properties": {
          "spark.dynamicAllocation.enabled":"false"
        }
      },
      {
        "classification": "emr-job-submitter",
        "properties": {
          "jobsubmitter.node.selector.eks.amazonaws.com/capacityType": "ON_DEMAND"
```

```
        }
      }
    ],
    "monitoringConfiguration": {
      "cloudWatchMonitoringConfiguration": {
        "logGroupName": "/emr-containers/jobs",
        "logStreamNamePrefix": "demo"
      },
      "s3MonitoringConfiguration": {
        "logUri": "s3://joblogs"
      }
    }
  }
}
EOF
aws emr-containers start-job-run --cli-input-json file:///spark-python-in-s3-
nodeselector-job-submitter.json
```

`StartJobRun`demande avec placement d'un nœud mono-AZ et placement du type d'instance Amazon EC2 pour le module d'envoi des tâches

```
"configurationOverrides": {
  "applicationConfiguration": [
    {
      "classification": "emr-job-submitter",
      "properties": {
        "jobsubmitter.node.selector.topology.kubernetes.io/zone": "Availability Zone",
        "jobsubmitter.node.selector.node.kubernetes.io/instance-type":"m5.4xlarge"
      }
    }
  ]
}
```

`StartJobRun`demande avec des étiquettes, des annotations et un planificateur personnalisé pour le module d'envoi des tâches

```
"configurationOverrides": {
  "applicationConfiguration": [
    {
      "classification": "emr-job-submitter",
      "properties": {
        "jobsubmitter.label.label1": "value1",
```

```
        "jobsubmitter.label.label2": "value2",
        "jobsubmitter.annotation.ann1": "value1",
        "jobsubmitter.annotation.ann2": "value2",
        "jobsubmitter.scheduler.name": "custom-scheduler"
      }
    }
  ]
}
```

`StartJobRun`demande avec une tolérance appliquée au module d'envoi de la tâche avec clé**`dedicated`**, valeur **`graviton_machinesNoExecute`**, effet et un **`tolerationSeconds`** délai de 60 secondes

```
"configurationOverrides": {
  "applicationConfiguration": [
    {
      "classification": "emr-job-submitter",
      "properties": {
        "jobsubmitter.node.toleration.dedicated":"graviton_machines",
        "jobsubmitter.node.toleration.dedicated.effect":"NoExecute",
        "jobsubmitter.node.toleration.dedicated.tolerationSeconds":"60"
      }
    }
  ]
}
```

`StartJobRun`demande avec journalisation désactivée pour le module d'envoi de tâches

```
"configurationOverrides": {
  "applicationConfiguration": [
    {
      "classification": "emr-job-submitter",
      "properties": {
        "jobsubmitter.logging": "DISABLED"
      }
    }
  ],
  "monitoringConfiguration": {
    "cloudWatchMonitoringConfiguration": {
      "logGroupName": "/emr-containers/jobs",
      "logStreamNamePrefix": "demo"
    },
```

```
    "s3MonitoringConfiguration": {
      "logUri": "s3://joblogs"
    }
  }
}
```

`StartJobRun`demande avec image du conteneur de journalisation personnalisé, processeur et mémoire pour le module d'envoi des tâches

```
"configurationOverrides": {
  "applicationConfiguration": [
    {
      "classification": "emr-job-submitter",
      "properties": {
        "jobsubmitter.logging.image": "YOUR_ECR_IMAGE_URL",
        "jobsubmitter.logging.request.memory": "200Mi",
        "jobsubmitter.logging.request.cores": "0.5"
      }
    }
  ],
  "monitoringConfiguration": {
    "cloudWatchMonitoringConfiguration": {
      "logGroupName": "/emr-containers/jobs",
      "logStreamNamePrefix": "demo"
    },
    "s3MonitoringConfiguration": {
      "logUri": "s3://joblogs"
    }
  }
}
```

`StartJobRun`demande avec une image de conteneur d'envoi de tâches personnalisée et une politique d'extraction

```
"configurationOverrides": {
  "applicationConfiguration": [
    {
      "classification": "emr-job-submitter",
      "properties": {
        "jobsubmitter.container.image": "123456789012.dkr.ecr.us-west-2.amazonaws.com/
emr6.11_custom_repo",
        "jobsubmitter.container.image.pullPolicy": "kubernetes pull policy"
```

```
      }
    }
  ]
}
```

Utilisation de la classification par défaut des conteneurs Amazon EMR

Présentation de

Les paramètres suivants sont disponibles dans la `emr-containers-defaults` classification :

`job-start-timeout`

Par défaut, une tâche expirera si elle ne peut pas démarrer et qu'elle attend dans son `SUBMITTED` état pendant 15 minutes. Cette configuration modifie le nombre de secondes à attendre avant l'expiration de la tâche.

`executor.logging`

Active ou désactive la journalisation sur les pods de l'exécuteur. Lorsque ce paramètre est défini sur, `DISABLED` le conteneur de journalisation est supprimé des pods de l'exécuteur, ce qui désactivera toute journalisation pour ces pods spécifiés dans le`monitoringConfiguration`, tel que `s3MonitoringConfiguration` ou`cloudWatchMonitoringConfiguration`. Lorsque ce paramètre n'est pas défini ou qu'il est défini sur une autre valeur, la connexion aux pods de l'exécuteur est activée.

`logging.image`

Définit une image personnalisée à utiliser pour le conteneur de journalisation sur les modules pilote et exécuteur.

`logging.request.cores`

Définit une valeur personnalisée pour le nombre CPUs, en unités de processeur, du conteneur de journalisation sur les pods pilote et exécuteur. Par défaut, ce paramètre n'est pas défini.

`logging.request.memory`

Définit une valeur personnalisée pour la quantité de mémoire, en octets, pour le conteneur de journalisation sur les pods pilote et exécuteur. Par défaut, ce paramètre est réglé sur 512 Mi. Un mébioctet est une unité de mesure similaire à un mégaoctet.

Exemples de classification des soumissionnaires de tâches

Dans cette section

StartJobRundemande avec délai d'expiration de tâche personnalisé

```
{
  "name": "spark-python",
  "virtualClusterId": "virtual-cluster-id",
  "executionRoleArn": "execution-role-arn",
  "releaseLabel": "emr-6.11.0-latest",
  "jobDriver": {
    "sparkSubmitJobDriver": {
      "entryPoint": "s3://S3-prefix/trip-count.py"
    }
  },
  "configurationOverrides": {
    "applicationConfiguration": [
      {
        "classification": "emr-containers-defaults",
        "properties": {
          "job-start-timeout": "1800"
        }
      }
    ],
    "monitoringConfiguration": {
      "cloudWatchMonitoringConfiguration": {
        "logGroupName": "/emr-containers/jobs",
        "logStreamNamePrefix": "demo"
      },
      "s3MonitoringConfiguration": {
        "logUri": "s3://joblogs"
      }
    }
  }
}
```

`StartJobRun`demande avec journalisation désactivée pour les modules d'exécution

```
"configurationOverrides": {
  "applicationConfiguration": [
    {
      "classification": "emr-containers-defaults",
      "properties": {
        "executor.logging": "DISABLED"
      }
    }
  ],
  "monitoringConfiguration": {
    "cloudWatchMonitoringConfiguration": {
      "logGroupName": "/emr-containers/jobs",
      "logStreamNamePrefix": "demo"
    },
    "s3MonitoringConfiguration": {
      "logUri": "s3://joblogs"
    }
  }
}
```

`StartJobRun`demande avec image du conteneur de journalisation personnalisé, processeur et mémoire pour les pods pilote et exécuteur

```
"configurationOverrides": {
  "applicationConfiguration": [
    {
      "classification": "emr-containers-defaults",
      "properties": {
        "logging.image": "YOUR_ECR_IMAGE_URL",
        "logging.request.memory": "200Mi",
        "logging.request.cores": "0.5"
      }
    }
  ],
  "monitoringConfiguration": {
    "cloudWatchMonitoringConfiguration": {
      "logGroupName": "/emr-containers/jobs",
      "logStreamNamePrefix": "demo"
    },
    "s3MonitoringConfiguration": {
      "logUri": "s3://joblogs"
```

```
    }
  }
}
```

Exécution de tâches Spark à l'aide de l'opérateur Spark

Les versions 6.10.0 et supérieures d'Amazon EMR prennent en charge l'opérateur Kubernetes pour Apache Spark, ou l'opérateur Spark, en tant que modèle de soumission de tâches pour Amazon EMR on EKS. Grâce à l'opérateur Spark, vous pouvez déployer et gérer des applications Spark à l'aide du moteur d'exécution Amazon EMR sur vos propres clusters Amazon EKS. Une fois que vous avez déployé l'opérateur Spark dans votre cluster Amazon EKS, vous pouvez directement soumettre des applications Spark à l'aide de cet opérateur. L'opérateur gère le cycle de vie des applications Spark.

> **Note**
>
> Amazon EMR calcule les tarifs sur Amazon EKS en fonction du vCPU et de la consommation de mémoire. Ce calcul s'applique aux modules pilote et exécuteur. Ce calcul commence à partir du moment où vous téléchargez l'image de votre application Amazon EMR jusqu'à ce que le module Amazon EKS se termine et est arrondi à la seconde près.

Rubriques

- Configuration de l'opérateur Spark pour Amazon EMR on EKS
- Les premiers pas avec l'opérateur Spark pour Amazon EMR on EKS
- Utiliser l'autoscaling vertical avec l'opérateur Spark pour Amazon EMR sur EKS
- Désinstallation de l'opérateur Spark pour Amazon EMR on EKS
- Utilisation de la configuration de surveillance pour surveiller l'opérateur Spark Kubernetes et les tâches Spark
- Sécurité et opérateur Spark avec Amazon EMR on EKS

Configuration de l'opérateur Spark pour Amazon EMR on EKS

Effectuez les tâches suivantes pour vous préparer avant d'installer l'opérateur Spark sur Amazon EKS. Si vous êtes déjà inscrit à Amazon Web Services (AWS) et que vous avez utilisé Amazon EKS, vous êtes presque prêt à utiliser Amazon EMR on EKS. Effectuez les tâches suivantes pour vous

préparer pour l'utilisation de l'opérateur Spark sur Amazon EKS. Si vous avez déjà rempli l'une des conditions préalables, vous pouvez l'ignorer et passer à la suivante.

- Installation ou mise à jour vers la dernière version du AWS CLI — Si vous avez déjà installé le AWS CLI, vérifiez que vous disposez de la dernière version.
- Configurer kubectl et eksctl — eksctl est un outil de ligne de commande que vous utilisez pour communiquer avec Amazon EKS.
- Installer Helm – Le gestionnaire de packages Helm pour Kubernetes vous aide à installer et à gérer des applications sur votre cluster Kubernetes.
- Commencez avec Amazon EKS — eksctl — Suivez les étapes pour créer un nouveau cluster Kubernetes avec des nœuds dans Amazon EKS.
- Sélectionner l'URI d'une image de base Amazon EMR (version 6.10.0 ou supérieure) – L'opérateur Spark est pris en charge par les versions 6.10.0 et supérieures d'Amazon EMR.

Les premiers pas avec l'opérateur Spark pour Amazon EMR on EKS

Cette rubrique vous aide à commencer à utiliser l'opérateur Spark sur Amazon EKS en déployant une application Spark et une application Schedule Spark.

Installation de l'opérateur Spark

Procédez comme suit pour installer l'opérateur Kubernetes pour Apache Spark.

1. Si vous ne l'avez pas déjà fait, suivez les étapes de Configuration de l'opérateur Spark pour Amazon EMR on EKS.
2. Authentifiez votre client Helm dans le registre Amazon ECR. Dans la commande suivante, remplacez les *region-id* valeurs par vos valeurs préférées Région AWS et par la *ECR-registry-account* valeur correspondante pour la région Comptes de registre Amazon ECR par région sur la page.

   ```
   aws ecr get-login-password \
   --region region-id | helm registry login \
   --username AWS \
   --password-stdin ECR-registry-account.dkr.ecr.region-id.amazonaws.com
   ```

3. Installez l'opérateur Spark à l'aide de la commande suivante.

Pour le paramètre `--version` des Charts de Helm, utilisez votre étiquette de version Amazon EMR avec le préfixe `emr-` et le suffixe de date supprimés. Par exemple, pour la version `emr-6.12.0-java17-latest`, spécifiez `6.12.0-java17`. L'exemple de la commande ci-dessous utilise la version `emr-7.12.0-latest`, elle spécifie donc `7.12.0` pour les Charts de Helm `--version`.

```
helm install spark-operator-demo \
  oci://895885662937.dkr.ecr.region-id.amazonaws.com/spark-operator \
  --set emrContainers.awsRegion=region-id \
  --version 7.12.0 \
  --namespace spark-operator \
  --create-namespace
```

Par défaut, la commande crée un compte de service `emr-containers-sa-spark-operator` pour l'opérateur Spark. Pour utiliser un autre compte de service, saisissez l'argument `serviceAccounts.sparkoperator.name`. Par exemple :

```
--set serviceAccounts.sparkoperator.name my-service-account-for-spark-operator
```

Si vous souhaitez utiliser l'autoscaling vertical avec l'opérateur Spark, ajoutez la ligne suivante à la commande d'installation pour autoriser les webhooks pour l'opérateur :

```
--set webhook.enable=true
```

4. Vérifiez que vous avez installé les Charts de Helm à l'aide de la commande `helm list` :

```
helm list --namespace spark-operator -o yaml
```

La commande `helm list` doit vous renvoyer les informations relatives à la version des Charts de Helm qui vient d'être déployée :

```
app_version: v1beta2-1.3.8-3.1.1
chart: spark-operator-7.12.0
name: spark-operator-demo
namespace: spark-operator
revision: "1"
status: deployed
updated: 2023-03-14 18:20:02.721638196 +0000 UTC
```

5. Terminez l'installation avec toutes les options supplémentaires dont vous avez besoin. Pour plus d'informations, consultez la `spark-on-k8s-operator`documentation sur GitHub.

Exécution d'une application Spark

L'opérateur Spark est pris en charge avec Amazon EMR en version 6.10.0 ou supérieure. Lorsque vous installez l'opérateur Spark, il crée le compte de service `emr-containers-sa-spark` pour exécuter les applications Spark par défaut. Suivez les étapes ci-dessous pour exécuter une application Spark avec l'opérateur Spark sur Amazon EMR on EKS en version 6.10.0 ou supérieure.

1. Pour pouvoir exécuter une application Spark à l'aide de l'opérateur Spark, suivez les étapes indiquées dans Configuration de l'opérateur Spark pour Amazon EMR on EKS et Installation de l'opérateur Spark.
2. Créez un fichier de définition `SparkApplication` `spark-pi.yaml` avec le contenu suivant :

```
apiVersion: "sparkoperator.k8s.io/v1beta2"
kind: SparkApplication
metadata:
  name: spark-pi
  namespace: spark-operator
spec:
  type: Scala
  mode: cluster
  image: "895885662937.dkr.ecr.us-west-2.amazonaws.com/spark/emr-6.10.0:latest"
  imagePullPolicy: Always
  mainClass: org.apache.spark.examples.SparkPi
  mainApplicationFile: "local:///usr/lib/spark/examples/jars/spark-examples.jar"
  sparkVersion: "3.3.1"
  restartPolicy:
    type: Never
  volumes:
    - name: "test-volume"
      hostPath:
        path: "/tmp"
        type: Directory
  driver:
    cores: 1
    coreLimit: "1200m"
    memory: "512m"
    labels:
      version: 3.3.1
```

```
    serviceAccount: emr-containers-sa-spark
    volumeMounts:
      - name: "test-volume"
        mountPath: "/tmp"
  executor:
    cores: 1
    instances: 1
    memory: "512m"
    labels:
      version: 3.3.1
    volumeMounts:
      - name: "test-volume"
        mountPath: "/tmp"
```

3. Maintenant, soumettez l'application Spark à l'aide de la commande suivante. Cela créera également un objet `SparkApplication` nommé `spark-pi` :

```
kubectl apply -f spark-pi.yaml
```

4. Vérifiez les événements de l'objet `SparkApplication` à l'aide de la commande suivante :

```
kubectl describe sparkapplication spark-pi --namespace spark-operator
```

Pour plus d'informations sur l'envoi d'applications à Spark via l'opérateur Spark, consultez la section Utiliser un `SparkApplication` dans la `spark-on-k8s-operator` documentation sur GitHub.

Utiliser Amazon S3 pour le stockage

Pour utiliser Amazon S3 comme option de stockage de fichiers, ajoutez les configurations suivantes à votre fichier YAML.

```
hadoopConf:
# EMRFS filesystem
  fs.s3.customAWSCredentialsProvider:
 com.amazonaws.auth.WebIdentityTokenCredentialsProvider
  fs.s3.impl: com.amazon.ws.emr.hadoop.fs.EmrFileSystem
  fs.AbstractFileSystem.s3.impl: org.apache.hadoop.fs.s3.EMRFSDelegate
  fs.s3.buffer.dir: /mnt/s3
  fs.s3.getObject.initialSocketTimeoutMilliseconds: "2000"
  mapreduce.fileoutputcommitter.algorithm.version.emr_internal_use_only.EmrFileSystem:
 "2"
```

```
  mapreduce.fileoutputcommitter.cleanup-
failures.ignored.emr_internal_use_only.EmrFileSystem: "true"
sparkConf:
 # Required for EMR Runtime
 spark.driver.extraClassPath: /usr/lib/hadoop-lzo/lib/*:/usr/lib/hadoop/hadoop-
aws.jar:/usr/share/aws/aws-java-sdk/*:/usr/share/aws/emr/emrfs/conf:/usr/share/aws/
emr/emrfs/lib/*:/usr/share/aws/emr/emrfs/auxlib/*:/usr/share/aws/emr/security/conf:/
usr/share/aws/emr/security/lib/*:/usr/share/aws/hmclient/lib/aws-glue-datacatalog-
spark-client.jar:/usr/share/java/Hive-JSON-Serde/hive-openx-serde.jar:/usr/share/aws/
sagemaker-spark-sdk/lib/sagemaker-spark-sdk.jar:/home/hadoop/extrajars/*
 spark.driver.extraLibraryPath: /usr/lib/hadoop/lib/native:/usr/lib/hadoop-lzo/lib/
native:/docker/usr/lib/hadoop/lib/native:/docker/usr/lib/hadoop-lzo/lib/native
 spark.executor.extraClassPath: /usr/lib/hadoop-lzo/lib/*:/usr/lib/hadoop/hadoop-
aws.jar:/usr/share/aws/aws-java-sdk/*:/usr/share/aws/emr/emrfs/conf:/usr/share/aws/
emr/emrfs/lib/*:/usr/share/aws/emr/emrfs/auxlib/*:/usr/share/aws/emr/security/conf:/
usr/share/aws/emr/security/lib/*:/usr/share/aws/hmclient/lib/aws-glue-datacatalog-
spark-client.jar:/usr/share/java/Hive-JSON-Serde/hive-openx-serde.jar:/usr/share/aws/
sagemaker-spark-sdk/lib/sagemaker-spark-sdk.jar:/home/hadoop/extrajars/*
 spark.executor.extraLibraryPath: /usr/lib/hadoop/lib/native:/usr/lib/hadoop-lzo/lib/
native:/docker/usr/lib/hadoop/lib/native:/docker/usr/lib/hadoop-lzo/lib/native
```

Si vous utilisez les versions 7.2.0 et supérieures d'Amazon EMR, les configurations sont incluses par défaut. Dans ce cas, vous pouvez définir le chemin du fichier sur `s3://<bucket_name>/<file_path>` plutôt que `local://<file_path>` dans le fichier YAML de l'application Spark.

Soumettez ensuite l'application Spark comme d'habitude.

Utiliser l'autoscaling vertical avec l'opérateur Spark pour Amazon EMR sur EKS

À partir d'Amazon EMR 7.0, vous pouvez utiliser la mise à l'échelle automatique verticale d'Amazon EMR on EKS pour simplifier la gestion des ressources. Elle permet d'adapter automatiquement les ressources de mémoire et de CPU aux besoins de la charge de travail que vous fournissez aux applications Spark sur Amazon EMR. Pour de plus amples informations, veuillez consulter Utilisation de la mise à l'échelle automatique verticale avec les tâches Spark sur Amazon EMR.

Cette section décrit comment configurer l'opérateur Spark pour utiliser l'autoscaling vertical.

Conditions préalables

Avant de configurer la surveillance, veillez à effectuer les tâches de configuration suivantes :

- Suivez les étapes de Configuration de l'opérateur Spark pour Amazon EMR on EKS.
- (facultatif) Si vous avez déjà installé une ancienne version de l'opérateur Spark, supprimez le SparkApplication/ScheduledSparkApplication CRD.

```
kubectl delete crd sparkApplication
kubectl delete crd scheduledSparkApplication
```

- Suivez les étapes de Installation de l'opérateur Spark. À l'étape 3, ajoutez la ligne suivante à la commande d'installation pour autoriser les webhooks pour l'opérateur :

```
--set webhook.enable=true
```

- Suivez les étapes de Configuration de la mise à l'échelle automatique verticale pour Amazon EMR on EKS.
- Donnez accès aux fichiers se trouvant dans votre emplacement Amazon S3 :

1. Annotez le compte de service de votre chauffeur et de votre opérateur avec `JobExecutionRole` les autorisations S3.

```
kubectl annotate serviceaccount -n spark-operator emr-containers-sa-spark
 eks.amazonaws.com/role-arn=JobExecutionRole
kubectl annotate serviceaccount -n spark-operator emr-containers-sa-spark-
operator eks.amazonaws.com/role-arn=JobExecutionRole
```

2. Mettez à jour la politique de confiance de votre rôle d'exécution des tâches dans cet espace de noms.

```
aws emr-containers update-role-trust-policy \
--cluster-name cluster \
--namespace ${Namespace}\
--role-name iam_role_name_for_job_execution
```

3. Modifiez la politique de confiance du rôle IAM de votre rôle d'exécution des tâches et mettez à jour le `serviceaccount` formulaire `emr-containers-sa-spark-*-*-xxxx` vers`emr-containers-sa-*`.

```
{
    "Effect": "Allow",
    "Principal": {
        "Federated": "OIDC-provider"
```

```
    },
    "Action": "sts:AssumeRoleWithWebIdentity",
    "Condition": {
        "StringLike": {
            "OIDC": "system:serviceaccount:${Namespace}:emr-containers-sa-*"
        }
    }
}
```

4. Si vous utilisez Amazon S3 comme espace de stockage de fichiers, ajoutez les valeurs par défaut suivantes à votre fichier yaml.

```
hadoopConf:
# EMRFS filesystem
  fs.s3.customAWSCredentialsProvider:
 com.amazonaws.auth.WebIdentityTokenCredentialsProvider
  fs.s3.impl: com.amazon.ws.emr.hadoop.fs.EmrFileSystem
  fs.AbstractFileSystem.s3.impl: org.apache.hadoop.fs.s3.EMRFSDelegate
  fs.s3.buffer.dir: /mnt/s3
  fs.s3.getObject.initialSocketTimeoutMilliseconds: "2000"

 mapreduce.fileoutputcommitter.algorithm.version.emr_internal_use_only.EmrFileSystem:
 "2"
  mapreduce.fileoutputcommitter.cleanup-
failures.ignored.emr_internal_use_only.EmrFileSystem: "true"
sparkConf:
 # Required for EMR Runtime
 spark.driver.extraClassPath: /usr/lib/hadoop-lzo/lib/*:/usr/lib/hadoop/hadoop-
aws.jar:/usr/share/aws/aws-java-sdk/*:/usr/share/aws/emr/emrfs/conf:/usr/share/
aws/emr/emrfs/lib/*:/usr/share/aws/emr/emrfs/auxlib/*:/usr/share/aws/emr/
security/conf:/usr/share/aws/emr/security/lib/*:/usr/share/aws/hmclient/lib/aws-
glue-datacatalog-spark-client.jar:/usr/share/java/Hive-JSON-Serde/hive-openx-
serde.jar:/usr/share/aws/sagemaker-spark-sdk/lib/sagemaker-spark-sdk.jar:/home/
hadoop/extrajars/*
 spark.driver.extraLibraryPath: /usr/lib/hadoop/lib/native:/usr/lib/hadoop-lzo/
lib/native:/docker/usr/lib/hadoop/lib/native:/docker/usr/lib/hadoop-lzo/lib/
native
 spark.executor.extraClassPath: /usr/lib/hadoop-lzo/lib/*:/usr/lib/hadoop/
hadoop-aws.jar:/usr/share/aws/aws-java-sdk/*:/usr/share/aws/emr/emrfs/conf:/usr/
share/aws/emr/emrfs/lib/*:/usr/share/aws/emr/emrfs/auxlib/*:/usr/share/aws/emr/
security/conf:/usr/share/aws/emr/security/lib/*:/usr/share/aws/hmclient/lib/aws-
glue-datacatalog-spark-client.jar:/usr/share/java/Hive-JSON-Serde/hive-openx-
```

```
serde.jar:/usr/share/aws/sagemaker-spark-sdk/lib/sagemaker-spark-sdk.jar:/home/
hadoop/extrajars/*
 spark.executor.extraLibraryPath: /usr/lib/hadoop/lib/native:/usr/lib/hadoop-
lzo/lib/native:/docker/usr/lib/hadoop/lib/native:/docker/usr/lib/hadoop-lzo/lib/
native
```

Exécuter une tâche avec l'autoscaling vertical sur l'opérateur Spark

Pour pouvoir exécuter une application Spark à l'aide de l'opérateur Spark, vous devez suivre les étapes indiquées dans Conditions préalables.

Pour utiliser la mise à l'échelle automatique verticale avec l'opérateur Spark, ajoutez la configuration suivante au pilote correspondant aux spécifications de votre application Spark afin d'activer la mise à l'échelle automatique verticale :

```
dynamicSizing:
  mode: Off
  signature: "my-signature"
```

Cette configuration permet une mise à l'échelle automatique verticale et constitue une configuration de signature obligatoire qui vous permet de choisir une signature pour votre tâche.

Pour plus d'informations sur les configurations et les valeurs des paramètres, consultez Configuration de l'autoscaling vertical pour Amazon EMR sur EKS. Par défaut, votre tâche est soumise en mode Désactivé, réservé à la surveillance uniquement, de la mise à l'échelle automatique verticale. Cet état de surveillance vous permet de calculer et de consulter les recommandations en matière de ressources sans procéder à la mise à l'échelle automatique. Pour plus d'informations, consultez la section Modes de mise à l'échelle automatique verticaux.

Voici un exemple de fichier de `SparkApplication` définition nommé `spark-pi.yaml` avec les configurations requises pour utiliser l'autoscaling vertical.

```
apiVersion: "sparkoperator.k8s.io/v1beta2"
kind: SparkApplication
metadata:
  name: spark-pi
  namespace: spark-operator
spec:
  type: Scala
```

```
  mode: cluster
  image: "895885662937.dkr.ecr.us-west-2.amazonaws.com/spark/emr-7.12.0:latest"
  imagePullPolicy: Always
  mainClass: org.apache.spark.examples.SparkPi
  mainApplicationFile: "local:///usr/lib/spark/examples/jars/spark-examples.jar"
  sparkVersion: "3.4.1"
  dynamicSizing:
    mode: Off
    signature: "my-signature"
  restartPolicy:
    type: Never
  volumes:
    - name: "test-volume"
      hostPath:
        path: "/tmp"
        type: Directory
  driver:
    cores: 1
    coreLimit: "1200m"
    memory: "512m"
    labels:
      version: 3.4.1
    serviceAccount: emr-containers-sa-spark
    volumeMounts:
      - name: "test-volume"
        mountPath: "/tmp"
  executor:
    cores: 1
    instances: 1
    memory: "512m"
    labels:
      version: 3.4.1
    volumeMounts:
      - name: "test-volume"
        mountPath: "/tmp"
```

Maintenant, soumettez l'application Spark à l'aide de la commande suivante. Cela créera également un objet `SparkApplication` nommé `spark-pi` :

```
kubectl apply -f spark-pi.yaml
```

Pour plus d'informations sur l'envoi d'applications à Spark via l'opérateur Spark, consultez la section Utiliser un `SparkApplication` dans la `spark-on-k8s-operator` documentation sur GitHub.

Vérification de la fonctionnalité de mise à l'échelle automatique verticale

Pour vérifier que la mise à l'échelle automatique verticale fonctionne correctement pour la tâche soumise, utilisez kubectl pour obtenir la ressource personnalisée `verticalpodautoscaler` et consulter vos recommandations de mise à l'échelle.

```
kubectl get verticalpodautoscalers --all-namespaces \
-l=emr-containers.amazonaws.com/dynamic.sizing.signature=my-signature
```

Le résultat de cette requête devrait ressembler à ce qui suit :

```
NAMESPACE        NAME                                                          MODE
 CPU   MEM          PROVIDED   AGE
spark-operator   ds-p73j6mkosvc4xeb3gr7x4xol2bfcw5evqimzqojrlysvj3giozuq-vpa   Off
     580026651   True       15m
```

Si votre résultat ne ressemble pas à cela ou contient un code d'erreur, consultez Résolution des problèmes de mise à l'échelle automatique verticale d'Amazon EMR on EKS pour des étapes permettant de résoudre le problème.

Pour supprimer les modules et les applications, exécutez la commande suivante :

```
kubectl delete sparkapplication spark-pi
```

Désinstallation de l'opérateur Spark pour Amazon EMR on EKS

Procédez comme suit pour désinstaller l'opérateur Spark.

1. Supprimez l'opérateur Spark en utilisant le bon espace de noms. Dans cet exemple, l'espace de noms est `spark-operator-demo`.

   ```
   helm uninstall spark-operator-demo -n spark-operator
   ```

2. Supprimez le compte de service de l'opérateur Spark :

   ```
   kubectl delete sa emr-containers-sa-spark-operator -n spark-operator
   ```

3. Supprimez l'opérateur Spark `CustomResourceDefinitions` (CRDs) :

   ```
   kubectl delete crd sparkapplications.sparkoperator.k8s.io
   ```

```
kubectl delete crd scheduledsparkapplications.sparkoperator.k8s.io
```

Utilisation de la configuration de surveillance pour surveiller l'opérateur Spark Kubernetes et les tâches Spark

La configuration de surveillance vous permet de configurer facilement l'archivage des journaux de votre application Spark et des journaux des opérateurs sur Amazon S3 ou vers Amazon CloudWatch. Vous pouvez choisir l'un ou les deux. Cela ajoute un sidecar d'agent de journalisation à vos modules d'opérateur, de pilote et d'exécuteur Spark, puis transmet les journaux de ces composants aux récepteurs que vous avez configurés.

Conditions préalables

Avant de configurer la surveillance, veillez à effectuer les tâches de configuration suivantes :

1. (Facultatif) Si vous avez déjà installé une ancienne version de l'opérateur Spark, supprimez le SparkApplication/ScheduledSparkApplicationCRD.

   ```
   kubectl delete crd scheduledsparkapplications.sparkoperator.k8s.io
   kubectl delete crd sparkapplications.sparkoperator.k8s.io
   ```

2. Créez un rôle operator/job d'exécution dans IAM si vous n'en avez pas déjà un.
3. Exécutez la commande suivante pour mettre à jour la politique de confiance du rôle operator/job d'exécution que vous venez de créer :

   ```
   aws emr-containers update-role-trust-policy \
   --cluster-name cluster \
   --namespace namespace \
   --role-name iam_role_name_for_operator/job_execution_role
   ```

4. Modifiez la politique de confiance du rôle IAM de votre rôle operator/job d'exécution comme suit :

   ```
   {
       "Effect": "Allow",
       "Principal": {
           "Federated": "${OIDC-provider}"
       },
       "Action": "sts:AssumeRoleWithWebIdentity",
       "Condition": {
   ```

```
        "StringLike": {
            "OIDC_PROVIDER:sub": "system:serviceaccount:${Namespace}:emr-
containers-sa-*"
        }
    }
}
```

5. Créez une politique de configuration de surveillance dans IAM avec les autorisations suivantes :

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogStreams",
        "logs:CreateLogStream",
        "logs:CreateLogGroup",
        "logs:PutLogEvents"
      ],
      "Resource": [
        "arn:aws:logs:*:*:log-group:log_group_name",
        "arn:aws:logs:*:*:log-group:log_group_name:*"
      ],
      "Sid": "AllowLOGSDescribelogstreams"
    },
    {
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogGroups"
      ],
      "Resource": [
        "*"
      ],
      "Sid": "AllowLOGSDescribeloggroups"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:GetObject",
```

```
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::bucket_name",
        "arn:aws:s3:::bucket_name/*"
      ],
      "Sid": "AllowS3Putobject"
    }
  ]
}
```

6. Associez la politique ci-dessus à votre rôle operator/job d'exécution.

Journaux des opérateurs Spark

Vous pouvez définir la configuration de surveillance de la manière suivante lors de cette opération `helm install` :

```
helm install spark-operator spark-operator \
--namespace namespace \
--set emrContainers.awsRegion=aws_region \
--set emrContainers.monitoringConfiguration.image=log_agent_image_url \
--set
 emrContainers.monitoringConfiguration.s3MonitoringConfiguration.logUri=S3_bucket_uri \
--set
 emrContainers.monitoringConfiguration.cloudWatchMonitoringConfiguration.logGroupName=log_group
 \
--set
 emrContainers.monitoringConfiguration.cloudWatchMonitoringConfiguration.logStreamNamePrefix=lo
 \
--set emrContainers.monitoringConfiguration.sideCarResources.limits.cpuLimit=500m \
--set emrContainers.monitoringConfiguration.sideCarResources.limits.memoryLimit=512Mi \
--set
 emrContainers.monitoringConfiguration.containerLogRotationConfiguration.rotationSize=2GB
 \
--set
 emrContainers.monitoringConfiguration.containerLogRotationConfiguration.maxFilesToKeep=10
 \
--set webhook.enable=true \
--set emrContainers.operatorExecutionRoleArn=operator_execution_role_arn
```

Configuration de surveillance

Les options de configuration disponibles sous MonitoringConfiguration sont les suivantes.

- Image (facultatif) — Enregistrez l'URL de l'image de l'agent. Je viendrai le chercher emrReleaseLabel s'il n'est pas fourni.
- s3 MonitoringConfiguration — Définissez cette option pour archiver sur Amazon S3.
  - LoGURI — (obligatoire) — Le chemin du compartiment Amazon S3 dans lequel vous souhaitez stocker vos journaux.
  - Vous trouverez ci-dessous des exemples de formats pour les chemins de compartiment Amazon S3, une fois les journaux chargés. Le premier exemple montre qu'aucune rotation des journaux n'est activée.

    ```
    s3://${logUri}/${POD NAME}/operator/stdout.gz
    s3://${logUri}/${POD NAME}/operator/stderr.gz
    ```

    La rotation des journaux est activée par défaut. Vous pouvez voir à la fois un fichier pivoté, avec un index incrémenté, et un fichier en cours, identique à l'exemple précédent.

    ```
    s3://${logUri}/${POD NAME}/operator/stdout_YYYYMMDD_index.gz
    s3://${logUri}/${POD NAME}/operator/stderr_YYYYMMDD_index.gz
    ```
- cloudWatchMonitoringConfiguration : clé de configuration vers laquelle configurer le transfert Amazon CloudWatch.
  - logGroupName(obligatoire) — Nom du groupe de Amazon CloudWatch journaux auquel vous souhaitez envoyer des journaux. Le groupe est automatiquement créé s'il n'existe pas.
  - logStreamNamePréfixe (facultatif) : nom du flux de journaux auquel vous souhaitez envoyer des journaux. La valeur par défaut est une chaîne vide. Le format Amazon CloudWatch est le suivant :

    ```
    ${logStreamNamePrefix}/${POD NAME}/STDOUT or STDERR
    ```
- sideCarResources(facultatif) — La clé de configuration pour définir les limites de ressources sur le conteneur latéral Fluentd lancé.
  - Limite de mémoire (facultatif) — Limite de mémoire. Ajustez selon vos besoins. La valeur par défaut est 512Mi.
  - CPULimit (facultatif) — Limite du processeur. Ajustez selon vos besoins. La valeur par défaut est de 500 m.

- containerLogRotationConfiguration (facultatif) — Contrôle le comportement de rotation du journal du conteneur. Il est activé par défaut.
  - RotationSize (obligatoire) — Spécifie la taille du fichier pour la rotation du journal. La plage de valeurs possibles est comprise entre 2 Ko et 2 Go. La partie unitaire numérique du paramètre rotationSize est transmise sous forme d'entier. Les valeurs décimales n'étant pas prises en charge, vous pouvez indiquer une taille de rotation de 1,5 Go, par exemple, avec la valeur 1500 Mo. La valeur par défaut est 2 Go.
  - maxFilesToConserver (obligatoire) — Spécifie le nombre maximum de fichiers à conserver dans le conteneur après la rotation. La valeur minimale est 1 et la valeur maximale est 50. La valeur par défaut est 10.

Après avoir configuré MonitoringConfiguration, vous devriez être en mesure de consulter les journaux des pods Spark Operator sur un compartiment Amazon S3 ou Amazon CloudWatch sur les deux. Pour un compartiment Amazon S3, vous devez attendre 2 minutes pour que le premier fichier journal soit vidé.

Pour trouver les connexions Amazon CloudWatch, vous pouvez accéder à ce qui suit : CloudWatch> Groupes de journaux > > ***Log group namePod name***/operator/stderr

Vous pouvez également accéder à : CloudWatch> Groupes de journaux > > ***Log group namePod name***/operator/stdout

Journaux des applications Spark

Vous pouvez définir cette configuration de la manière suivante.

```
apiVersion: "sparkoperator.k8s.io/v1beta2"
kind: SparkApplication
metadata:
  name: spark-pi
  namespace: namespace
spec:
  type: Scala
  mode: cluster
  imagePullPolicy: Always
  mainClass: org.apache.spark.examples.SparkPi
  mainApplicationFile: "local:///usr/lib/spark/examples/jars/spark-examples.jar"
  sparkVersion: "3.3.1"
  emrReleaseLabel: emr_release_label
```

```
  executionRoleArn: job_execution_role_arn
  restartPolicy:
    type: Never
  volumes:
    - name: "test-volume"
      hostPath:
        path: "/tmp"
        type: Directory
  driver:
    cores: 1
    coreLimit: "1200m"
    memory: "512m"
    labels:
      version: 3.3.1
    volumeMounts:
      - name: "test-volume"
        mountPath: "/tmp"
  executor:
    cores: 1
    instances: 1
    memory: "512m"
    labels:
      version: 3.3.1
    volumeMounts:
      - name: "test-volume"
        mountPath: "/tmp"
  monitoringConfiguration:
    image: "log_agent_image"
    s3MonitoringConfiguration:
      logUri: "S3_bucket_uri"
    cloudWatchMonitoringConfiguration:
      logGroupName: "log_group_name"
      logStreamNamePrefix: "log_stream_prefix"
    sideCarResources:
      limits:
        cpuLimit: "500m"
        memoryLimit: "250Mi"
    containerLogRotationConfiguration:
      rotationSize: "2GB"
      maxFilesToKeep: "10"
```

Les options de configuration disponibles sous MonitoringConfiguration sont les suivantes.

- Image (facultatif) — Enregistrez l'URL de l'image de l'agent. Je viendrai le chercher emrReleaseLabel s'il n'est pas fourni.
- s3 MonitoringConfiguration — Définissez cette option pour archiver sur Amazon S3.
  - LoGURI (obligatoire) — Le chemin du compartiment Amazon S3 dans lequel vous souhaitez stocker vos journaux. Le premier exemple montre qu'aucune rotation des journaux n'est activée :

    ```
    s3://${logUri}/${APPLICATION NAME}-${APPLICATION UID}/${POD NAME}/stdout.gz
    s3://${logUri}/${APPLICATION NAME}-${APPLICATION UID}/${POD NAME}/stderr.gz
    ```

    La rotation des journaux est activée par défaut. Vous pouvez utiliser à la fois un fichier pivoté (avec index incrémenté) et un fichier actuel (sans horodatage).

    ```
    s3://${logUri}/${APPLICATION NAME}-${APPLICATION UID}/${POD NAME}/
    stdout_YYYYMMDD_index.gz
    s3://${logUri}/${APPLICATION NAME}-${APPLICATION UID}/${POD NAME}/
    stderr_YYYYMMDD_index.gz
    ```
- cloudWatchMonitoringConfiguration : clé de configuration vers laquelle configurer le transfert Amazon CloudWatch.
  - logGroupName(obligatoire) — Le nom du groupe de journaux Cloudwatch auquel vous souhaitez envoyer des journaux. Le groupe est automatiquement créé s'il n'existe pas.
  - logStreamNamePréfixe (facultatif) : nom du flux de journaux auquel vous souhaitez envoyer des journaux. La valeur par défaut est une chaîne vide. Le format CloudWatch est le suivant :

    ```
    ${logStreamNamePrefix}/${APPLICATION NAME}-${APPLICATION UID}/${POD NAME}/stdout
    ${logStreamNamePrefix}/${APPLICATION NAME}-${APPLICATION UID}/${POD NAME}/stderr
    ```
- sideCarResources(facultatif) — La clé de configuration pour définir les limites de ressources sur le conteneur latéral Fluentd lancé.
  - Limite de mémoire (facultatif) — Limite de mémoire. Ajustez selon vos besoins. La valeur par défaut est 250 Mi.
  - CPULimit — Limite du processeur. Ajustez selon vos besoins. La valeur par défaut est de 500 m.
- containerLogRotationConfiguration (facultatif) — Contrôle le comportement de rotation du journal du conteneur. Il est activé par défaut.
  - RotationSize (obligatoire) — Spécifie la taille du fichier pour la rotation du journal. La plage de valeurs possibles est comprise entre 2 Ko et 2 Go. La partie unitaire numérique du paramètre rotationSize est transmise sous forme d'entier. Les valeurs décimales n'étant pas prises en

charge, vous pouvez indiquer une taille de rotation de 1,5 Go, par exemple, avec la valeur 1500 Mo. La valeur par défaut est 2 Go.

- maxFilesToConserver (obligatoire) — Spécifie le nombre maximum de fichiers à conserver dans le conteneur après la rotation. La valeur minimale est de 1. La valeur maximale est de 50. La valeur par défaut est 10.

Après avoir configuré MonitoringConfiguration, vous devriez être en mesure de consulter les journaux du pilote et de l'exécuteur de votre application Spark sur un compartiment Amazon S3, ou sur les deux. CloudWatch Pour un compartiment Amazon S3, vous devez attendre 2 minutes pour que le premier fichier journal soit vidé. Par exemple, dans Amazon S3, le chemin du compartiment apparaît comme suit :

Amazon S3 > Compartiments > ***Bucket name*** > > ***Spark application name - UUID Pod Name*** > stderr.gz

Ou:

Amazon S3 > Compartiments > ***Bucket name*** > > ***Spark application name - UUID Pod Name*** > stdout.gz

Dans CloudWatch, le chemin apparaît comme suit :

CloudWatch> Groupes de journaux > ***Log group name***>*Spark application name - UUID*/*Pod name*/stderr

Ou:

CloudWatch> Groupes de journaux > ***Log group name***>*Spark application name - UUID*/*Pod name*/stdout

Sécurité et opérateur Spark avec Amazon EMR on EKS

Il existe deux manières de configurer les autorisations d'accès au cluster lorsque vous utilisez l'opérateur Spark. La première consiste à utiliser le contrôle d'accès basé sur les rôles. Le contrôle d'accès basé sur les rôles (RBAC) restreint l'accès en fonction du rôle d'une personne au sein d'une organisation. C'est devenu le principal moyen de gérer l'accès. La deuxième méthode d'accès consiste à assumer un Gestion des identités et des accès AWS rôle, qui fournit un accès aux ressources au moyen d'autorisations spécifiques attribuées.

Rubriques

- Configuration des autorisations d'accès au cluster avec le contrôle d'accès basé sur les rôles (RBAC)
- Configuration des autorisations d'accès au cluster avec des rôles IAM pour les comptes de service (IRSA)

Configuration des autorisations d'accès au cluster avec le contrôle d'accès basé sur les rôles (RBAC)

Pour déployer l'opérateur Spark, Amazon EMR on EKS crée deux rôles et comptes de service pour l'opérateur Spark et les applications Spark.

Rubriques

- Compte de service et rôle de l'opérateur
- Compte de service et rôle Spark

Compte de service et rôle de l'opérateur

Amazon EMR on EKS crée le compte de service et le rôle de l'opérateur pour gérer `SparkApplications` pour les tâches Spark et pour d'autres ressources telles que les services.

Le nom par défaut de ce compte de service est `emr-containers-sa-spark-operator`.

Les règles suivantes s'appliquent à cette fonction du service :

```
 rules:
- apiGroups:
  - ""
  resources:
  - pods
  verbs:
  - "*"
- apiGroups:
  - ""
  resources:
  - services
  - configmaps
  - secrets
  verbs:
  - create
  - get
```

```
  - delete
  - update
- apiGroups:
  - extensions
  - networking.k8s.io
  resources:
  - ingresses
  verbs:
  - create
  - get
  - delete
- apiGroups:
  - ""
  resources:
  - nodes
  verbs:
  - get
- apiGroups:
  - ""
  resources:
  - events
  verbs:
  - create
  - update
  - patch
- apiGroups:
  - ""
  resources:
  - resourcequotas
  verbs:
  - get
  - list
  - watch
- apiGroups:
  - apiextensions.k8s.io
  resources:
  - customresourcedefinitions
  verbs:
  - create
  - get
  - update
  - delete
- apiGroups:
  - admissionregistration.k8s.io
```

```
  resources:
  - mutatingwebhookconfigurations
  - validatingwebhookconfigurations
  verbs:
  - create
  - get
  - update
  - delete
- apiGroups:
  - sparkoperator.k8s.io
  resources:
  - sparkapplications
  - sparkapplications/status
  - scheduledsparkapplications
  - scheduledsparkapplications/status
  verbs:
  - "*"
  {{- if .Values.batchScheduler.enable }}
  # required for the `volcano` batch scheduler
- apiGroups:
  - scheduling.incubator.k8s.io
  - scheduling.sigs.dev
  - scheduling.volcano.sh
  resources:
  - podgroups
  verbs:
  - "*"
  {{- end }}
  {{ if .Values.webhook.enable }}
- apiGroups:
  - batch
  resources:
  - jobs
  verbs:
  - delete
  {{- end }}
```

Compte de service et rôle Spark

Le pod du pilote Spark a besoin d'un compte de service Kubernetes dans le même espace de noms que le pod. Ce compte de service a besoin d'autorisations pour créer, obtenir, répertorier, patcher et supprimer des pods d'exécuteurs, ainsi que pour créer un service Kubernetes sans tête pour le pilote.

Le pilote échoue et se termine sans le compte de service, sauf si le compte de service par défaut dans l'espace de noms du pod dispose des autorisations requises.

Le nom par défaut de ce compte de service est `emr-containers-sa-spark`.

Les règles suivantes s'appliquent à cette fonction du service :

```
 rules:
- apiGroups:
  - ""
  resources:
  - pods
  verbs:
  - "*"
- apiGroups:
  - ""
  resources:
  - services
  verbs:
  - "*"
- apiGroups:
  - ""
  resources:
  - configmaps
  verbs:
  - "*"
- apiGroups:
  - ""
  resources:
  - persistentvolumeclaims
  verbs:
  - "*"
```

Configuration des autorisations d'accès au cluster avec des rôles IAM pour les comptes de service (IRSA)

Cette section utilise un exemple pour montrer comment configurer un compte de service Kubernetes pour qu'il assume un rôle. Gestion des identités et des accès AWS Les pods qui utilisent le compte de service peuvent ensuite accéder à tout AWS service auquel le rôle est autorisé à accéder.

L'exemple suivant exécute une application Spark pour compter les mots d'un fichier dans Amazon S3. Pour ce faire, vous pouvez configurer des rôles IAM pour les comptes de service (IRSA) afin d'authentifier et d'autoriser les comptes de service Kubernetes.

> **Note**
>
> Cet exemple utilise l'espace de noms « spark-operator » pour l'opérateur Spark et pour l'espace de noms dans lequel vous soumettez l'application Spark.

Conditions préalables

Avant d'essayer l'exemple présenté sur cette page, vous devez remplir les conditions préalables suivantes :

- Préparez-vous pour l'utilisation de l'opérateur Spark.
- Installation de l'opérateur Spark.
- Créez un compartiment Amazon S3.
- Enregistrez votre poème préféré dans un fichier texte nommé `poem.txt`, puis chargez ce fichier dans votre compartiment S3. L'application Spark que vous créez sur cette page lira le contenu du fichier texte. Pour plus d'informations sur le chargement des fichiers sur S3, consultez la rubrique Chargement d'un objet dans votre compartiment du Guide de l'utilisateur Amazon Simple Storage Service.

Configuration d'un compte de service Kubernetes pour qu'il assume un rôle IAM

Suivez les étapes ci-dessous pour configurer un compte de service Kubernetes afin qu'il assume un rôle IAM que les pods peuvent utiliser pour accéder aux AWS services auxquels le rôle est autorisé à accéder.

1. Une fois leConditions préalables, utilisez le AWS Command Line Interface pour créer un `example-policy.json` fichier qui autorise un accès en lecture seule au fichier que vous avez chargé sur Amazon S3 :

```
cat >example-policy.json <<EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
```

```
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::my-pod-bucket",
                "arn:aws:s3:::my-pod-bucket/*"
            ]
        }
    ]
}
EOF
```

2. Ensuite, créez une politique IAM `example-policy` :

```
aws iam create-policy --policy-name example-policy --policy-document file://
example-policy.json
```

3. Puis créez un rôle IAM `example-role` et associez-le à un compte de service Kubernetes pour le pilote Spark :

```
eksctl create iamserviceaccount --name driver-account-sa --namespace spark-operator
 \
--cluster my-cluster --role-name "example-role" \
--attach-policy-arn arn:aws:iam::111122223333:policy/example-policy --approve
```

4. Créez un fichier yaml avec les liaisons de rôle de cluster qui sont nécessaires pour le compte de service du pilote Spark :

```
cat >spark-rbac.yaml <<EOF
apiVersion: v1
kind: ServiceAccount
metadata:
  name: driver-account-sa
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: spark-role
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
```

```
  name: edit
subjects:
  - kind: ServiceAccount
    name: driver-account-sa
    namespace: spark-operator
EOF
```

5. Appliquez les configurations de liaison des rôle du cluster :

```
kubectl apply -f spark-rbac.yaml
```

La commande kubectl doit confirmer la création réussie du compte :

```
serviceaccount/driver-account-sa created
clusterrolebinding.rbac.authorization.k8s.io/spark-role configured
```

Exécution d'une application depuis l'opérateur Spark

Après avoir configuré le compte de service Kubernetes, vous pouvez exécuter une application Spark qui compte le nombre de mots contenus dans le fichier texte que vous avez chargé dans le cadre des conditions préalables Conditions préalables.

1. Créez un nouveau fichier`word-count.yaml`, avec une `SparkApplication` définition pour votre application de comptage de mots, sur la base de la version 6 d'Amazon EMR.

```
cat >word-count.yaml <<EOF
apiVersion: "sparkoperator.k8s.io/v1beta2"
kind: SparkApplication
metadata:
  name: word-count
  namespace: spark-operator
spec:
  type: Java
  mode: cluster
  image: "895885662937.dkr.ecr.us-west-2.amazonaws.com/spark/emr-6.10.0:latest"
  imagePullPolicy: Always
  mainClass: org.apache.spark.examples.JavaWordCount
  mainApplicationFile: local:///usr/lib/spark/examples/jars/spark-examples.jar
  arguments:
    - s3://my-pod-bucket/poem.txt
  hadoopConf:
```

```
   # EMRFS filesystem
    fs.s3.customAWSCredentialsProvider:
 com.amazonaws.auth.WebIdentityTokenCredentialsProvider
    fs.s3.impl: com.amazon.ws.emr.hadoop.fs.EmrFileSystem
    fs.AbstractFileSystem.s3.impl: org.apache.hadoop.fs.s3.EMRFSDelegate
    fs.s3.buffer.dir: /mnt/s3
    fs.s3.getObject.initialSocketTimeoutMilliseconds: "2000"

 mapreduce.fileoutputcommitter.algorithm.version.emr_internal_use_only.EmrFileSystem:
 "2"
    mapreduce.fileoutputcommitter.cleanup-
failures.ignored.emr_internal_use_only.EmrFileSystem: "true"
  sparkConf:
    # Required for EMR Runtime
    spark.driver.extraClassPath: /usr/lib/hadoop-lzo/lib/*:/usr/lib/hadoop/
hadoop-aws.jar:/usr/share/aws/aws-java-sdk/*:/usr/share/aws/emr/emrfs/conf:/usr/
share/aws/emr/emrfs/lib/*:/usr/share/aws/emr/emrfs/auxlib/*:/usr/share/aws/emr/
security/conf:/usr/share/aws/emr/security/lib/*:/usr/share/aws/hmclient/lib/aws-
glue-datacatalog-spark-client.jar:/usr/share/java/Hive-JSON-Serde/hive-openx-
serde.jar:/usr/share/aws/sagemaker-spark-sdk/lib/sagemaker-spark-sdk.jar:/home/
hadoop/extrajars/*
    spark.driver.extraLibraryPath: /usr/lib/hadoop/lib/native:/usr/lib/hadoop-lzo/
lib/native:/docker/usr/lib/hadoop/lib/native:/docker/usr/lib/hadoop-lzo/lib/native
    spark.executor.extraClassPath: /usr/lib/hadoop-lzo/lib/*:/usr/lib/hadoop/
hadoop-aws.jar:/usr/share/aws/aws-java-sdk/*:/usr/share/aws/emr/emrfs/conf:/usr/
share/aws/emr/emrfs/lib/*:/usr/share/aws/emr/emrfs/auxlib/*:/usr/share/aws/emr/
security/conf:/usr/share/aws/emr/security/lib/*:/usr/share/aws/hmclient/lib/aws-
glue-datacatalog-spark-client.jar:/usr/share/java/Hive-JSON-Serde/hive-openx-
serde.jar:/usr/share/aws/sagemaker-spark-sdk/lib/sagemaker-spark-sdk.jar:/home/
hadoop/extrajars/*
    spark.executor.extraLibraryPath: /usr/lib/hadoop/lib/native:/usr/lib/hadoop-
lzo/lib/native:/docker/usr/lib/hadoop/lib/native:/docker/usr/lib/hadoop-lzo/lib/
native
  sparkVersion: "3.3.1"
  restartPolicy:
    type: Never
  driver:
    cores: 1
    coreLimit: "1200m"
    memory: "512m"
    labels:
      version: 3.3.1
    serviceAccount: my-spark-driver-sa
  executor:
```

```
    cores: 1
    instances: 1
    memory: "512m"
    labels:
      version: 3.3.1
EOF
```

Si vous utilisez l'opérateur Spark dans une version 7, vous devez ajuster certaines valeurs de configuration :

```
cat >word-count.yaml <<EOF
apiVersion: "sparkoperator.k8s.io/v1beta2"
kind: SparkApplication
metadata:
  name: word-count
  namespace: spark-operator
spec:
  type: Java
  mode: cluster
  image: "895885662937.dkr.ecr.us-west-2.amazonaws.com/spark/emr-7.7.0:latest"
  imagePullPolicy: Always
  mainClass: org.apache.spark.examples.JavaWordCount
  mainApplicationFile: local:///usr/lib/spark/examples/jars/spark-examples.jar
  arguments:
    - s3://my-pod-bucket/poem.txt
  hadoopConf:
   # EMRFS filesystem
    fs.s3.customAWSCredentialsProvider:
 com.amazonaws.auth.WebIdentityTokenCredentialsProvider
    fs.s3.impl: com.amazon.ws.emr.hadoop.fs.EmrFileSystem
    fs.AbstractFileSystem.s3.impl: org.apache.hadoop.fs.s3.EMRFSDelegate
    fs.s3.buffer.dir: /mnt/s3
    fs.s3.getObject.initialSocketTimeoutMilliseconds: "2000"

 mapreduce.fileoutputcommitter.algorithm.version.emr_internal_use_only.EmrFileSystem:
 "2"
    mapreduce.fileoutputcommitter.cleanup-
failures.ignored.emr_internal_use_only.EmrFileSystem: "true"
  sparkConf:
    # Required for EMR Runtime
    spark.driver.extraClassPath: /usr/lib/hadoop-lzo/lib/*:/usr/lib/hadoop/hadoop-
aws.jar:/usr/share/aws/aws-java-sdk/*:/usr/share/aws/aws-java-sdk-v2/*:/usr/share/
aws/emr/emrfs/conf:/usr/share/aws/emr/emrfs/lib/*:/usr/share/aws/emr/emrfs/auxlib/
```

```
*:/usr/share/aws/emr/security/conf:/usr/share/aws/emr/security/lib/*:/usr/share/
aws/hmclient/lib/aws-glue-datacatalog-spark-client.jar:/usr/share/java/Hive-JSON-
Serde/hive-openx-serde.jar:/usr/share/aws/sagemaker-spark-sdk/lib/sagemaker-spark-
sdk.jar:/home/hadoop/extrajars/*
    spark.driver.extraLibraryPath: /usr/lib/hadoop/lib/native:/usr/lib/hadoop-lzo/
lib/native:/docker/usr/lib/hadoop/lib/native:/docker/usr/lib/hadoop-lzo/lib/native
    spark.executor.extraClassPath: /usr/lib/hadoop-lzo/lib/*:/usr/lib/hadoop/
hadoop-aws.jar:/usr/share/aws/aws-java-sdk/*:/usr/share/aws/aws-java-sdk-v2/*:/usr/
share/aws/emr/emrfs/conf:/usr/share/aws/emr/emrfs/lib/*:/usr/share/aws/emr/emrfs/
auxlib/*:/usr/share/aws/emr/security/conf:/usr/share/aws/emr/security/lib/*:/usr/
share/aws/hmclient/lib/aws-glue-datacatalog-spark-client.jar:/usr/share/java/Hive-
JSON-Serde/hive-openx-serde.jar:/usr/share/aws/sagemaker-spark-sdk/lib/sagemaker-
spark-sdk.jar:/home/hadoop/extrajars/*
    spark.executor.extraLibraryPath: /usr/lib/hadoop/lib/native:/usr/lib/hadoop-
lzo/lib/native:/docker/usr/lib/hadoop/lib/native:/docker/usr/lib/hadoop-lzo/lib/
native
  sparkVersion: "3.3.1"
  restartPolicy:
    type: Never
  driver:
    cores: 1
    coreLimit: "1200m"
    memory: "512m"
    labels:
      version: 3.3.1
    serviceAccount: my-spark-driver-sa
  executor:
    cores: 1
    instances: 1
    memory: "512m"
    labels:
      version: 3.3.1
EOF
```

2. Soumettez l'application Spark.

```
kubectl apply -f word-count.yaml
```

La commande kubectl doit renvoyer la confirmation que vous avez créé avec succès un objet `SparkApplication` appelé `word-count`.

```
sparkapplication.sparkoperator.k8s.io/word-count configured
```

3. Pour vérifier les événements de l'objet `SparkApplication`, exécutez la commande suivante :

```
kubectl describe sparkapplication word-count -n spark-operator
```

La commande kubectl doit renvoyer la description de `SparkApplication` avec les événements :

```
Events:
  Type     Reason                                Age                    From
    Message
  ----     ------                                ----                   ----
    -------
  Normal   SparkApplicationSpecUpdateProcessed  3m2s (x2 over 17h)     spark-
operator  Successfully processed spec update for SparkApplication word-count
  Warning  SparkApplicationPendingRerun         3m2s (x2 over 17h)     spark-
operator  SparkApplication word-count is pending rerun
  Normal   SparkApplicationSubmitted            2m58s (x2 over 17h)    spark-
operator  SparkApplication word-count was submitted successfully
  Normal   SparkDriverRunning                   2m56s (x2 over 17h)    spark-
operator  Driver word-count-driver is running
  Normal   SparkExecutorPending                 2m50s                  spark-
operator  Executor [javawordcount-fdd1698807392c66-exec-1] is pending
  Normal   SparkExecutorRunning                 2m48s                  spark-
operator  Executor [javawordcount-fdd1698807392c66-exec-1] is running
  Normal   SparkDriverCompleted                 2m31s (x2 over 17h)    spark-
operator  Driver word-count-driver completed
  Normal   SparkApplicationCompleted            2m31s (x2 over 17h)    spark-
operator  SparkApplication word-count completed
  Normal   SparkExecutorCompleted               2m31s (x2 over 2m31s)  spark-
operator  Executor [javawordcount-fdd1698807392c66-exec-1] completed
```

L'application compte maintenant les mots de votre fichier S3. Pour connaître le nombre de mots, consultez les fichiers journaux de votre pilote :

```
kubectl logs pod/word-count-driver -n spark-operator
```

La commande kubectl doit renvoyer le contenu du fichier journal avec les résultats de votre application de comptage de mots.

```
INFO DAGScheduler: Job 0 finished: collect at JavaWordCount.java:53, took 5.146519 s
```

```
                Software: 1
```

Pour plus d'informations sur la façon de soumettre des applications à Spark via l'opérateur Spark, consultez la documentation relative SparkApplication à l'utilisation d'un opérateur Kubernetes pour Apache Spark (opérateur spark-on-k 8s) sur. GitHub

Exécution de tâches Spark en utilisant spark-submit

Les versions 6.10.0 et supérieures d'Amazon EMR prennent en charge `spark-submit` en tant qu'outil de ligne de commande que vous pouvez utiliser pour soumettre et exécuter des applications Spark sur un cluster Amazon EMR on EKS.

> **Note**
>
> Amazon EMR calcule les tarifs sur Amazon EKS en fonction du vCPU et de la consommation de mémoire. Ce calcul s'applique aux modules pilote et exécuteur. Ce calcul commence à partir du moment où vous téléchargez l'image de votre application Amazon EMR jusqu'à ce que le module Amazon EKS se termine et est arrondi à la seconde près.

Rubriques

- Configuration de spark-submit pour Amazon EMR on EKS
- Les premiers pas avec spark-submit pour Amazon EMR on EKS
- Vérifiez les exigences de sécurité du compte Spark Driver Service pour Spark-Submit

Configuration de spark-submit pour Amazon EMR on EKS

Effectuez les tâches suivantes pour vous préparer à exécuter une application en utilisant spark-submit sur Amazon EMR on EKS. Si vous êtes déjà inscrit à Amazon Web Services (AWS) et que vous avez utilisé Amazon EKS, vous êtes presque prêt à utiliser Amazon EMR on EKS. Si vous avez déjà rempli l'une des conditions préalables, vous pouvez l'ignorer et passer à la suivante.

- Installation ou mise à jour vers la dernière version du AWS CLI — Si vous avez déjà installé le AWS CLI, vérifiez que vous disposez de la dernière version.
- Configurer kubectl et eksctl — eksctl est un outil de ligne de commande que vous utilisez pour communiquer avec Amazon EKS.

- Commencez avec Amazon EKS — eksctl — Suivez les étapes pour créer un nouveau cluster Kubernetes avec des nœuds dans Amazon EKS.
- Sélectionner l'URI d'une image de base Amazon EMR (version 6.10.0 ou supérieure) – La commande `spark-submit` est prise en charge par les versions 6.10.0 et supérieures d'Amazon EMR.
- Confirmez que le compte de service du pilote dispose des autorisations nécessaires pour créer et surveiller les pods d'exécuteurs. Pour de plus amples informations, veuillez consulter Vérifiez les exigences de sécurité du compte Spark Driver Service pour Spark-Submit.
- Configurez votre profil d'informations d'identification AWS local.
- Dans la console Amazon EKS, choisissez votre cluster EKS, puis recherchez le point de terminaison du cluster EKS, situé sous Vue d'ensemble, Détails, puis point de terminaison du serveur API.

Les premiers pas avec spark-submit pour Amazon EMR on EKS

Amazon EMR en version 6.10.0 et supérieure prend en charge spark-submit pour l'exécution d'applications Spark sur un cluster Amazon EKS. La section qui suit explique comment envoyer une commande pour une application Spark.

Exécution d'une application Spark

Pour exécuter l'application Spark, procédez comme suit :

1. Pour pouvoir exécuter une application Spark à l'aide de la commande `spark-submit`, suivez les étapes indiquées dans Configuration de spark-submit pour Amazon EMR on EKS.
2. Exécutez un conteneur avec une image de base Amazon EMR on EKS. Consultez Comment sélectionner un URI d'image de base pour plus d'informations.

   ```
   kubectl run -it containerName --image=EMRonEKSImage --command -n namespace /bin/
   bash
   ```

3. Définissez les valeurs des variables d'environnement suivantes :

   ```
   export SPARK_HOME=spark-home
   export MASTER_URL=k8s://Amazon EKS-cluster-endpoint
   ```

4. Maintenant, soumettez l'application Spark avec la commande suivante :

```
$SPARK_HOME/bin/spark-submit \
 --class org.apache.spark.examples.SparkPi \
 --master $MASTER_URL \
 --conf spark.kubernetes.container.image=895885662937.dkr.ecr.us-
west-2.amazonaws.com/spark/emr-6.10.0:latest \
 --conf spark.kubernetes.authenticate.driver.serviceAccountName=spark \
 --deploy-mode cluster \
 --conf spark.kubernetes.namespace=spark-operator \
 local:///usr/lib/spark/examples/jars/spark-examples.jar 20
```

Pour plus d'informations sur la soumission des applications à Spark, consultez la rubrique Soumission d'applications dans la documentation Apache Spark.

> **Important**
>
> `spark-submit` prend uniquement en charge le mode cluster comme mécanisme de soumission.

Vérifiez les exigences de sécurité du compte Spark Driver Service pour Spark-Submit

Le pod du pilote Spark utilise un compte de service Kubernetes pour accéder au serveur d'API Kubernetes afin de créer et de surveiller les pods d'exécuteurs. Le compte de service du pilote doit disposer des autorisations appropriées pour répertorier, créer, modifier, corriger et supprimer les pods dans votre cluster. Vous pouvez vérifier que vous pouvez répertorier ces ressources en exécutant la commande suivante :

```
kubectl auth can-i list|create|edit|delete|patch pods
```

Vérifiez que vous disposez des autorisations nécessaires en exécutant chaque commande.

```
kubectl auth can-i list pods
kubectl auth can-i create pods
kubectl auth can-i edit pods
kubectl auth can-i delete pods
kubectl auth can-i patch pods
```

Les règles suivantes s'appliquent à cette fonction du service :

```
 rules:
- apiGroups:
  - ""
  resources:
  - pods
  verbs:
  - "*"
- apiGroups:
  - ""
  resources:
  - services
  verbs:
  - "*"
- apiGroups:
  - ""
  resources:
  - configmaps
  verbs:
  - "*"
- apiGroups:
  - ""
  resources:
  - persistentvolumeclaims
  verbs:
  - "*"
```

Configuration des rôles IAM pour les comptes de service (IRSA) pour spark-submit

Les sections suivantes expliquent comment configurer les rôles IAM pour les comptes de service (IRSA) afin d'authentifier et d'autoriser les comptes de service Kubernetes afin que vous puissiez exécuter des applications Spark stockées dans Amazon S3.

Conditions préalables

Avant d'essayer l'un des exemples de cette documentation, assurez-vous que vous avez rempli les conditions préalables suivantes :

- Configuration de Spark-Submit terminée
- Création d'un compartiment S3 et téléchargement du fichier jar de l'application Spark

Configuration d'un compte de service Kubernetes pour qu'il assume un rôle IAM

Les étapes suivantes expliquent comment configurer un compte de service Kubernetes pour qu'il assume un rôle Gestion des identités et des accès AWS (IAM). Une fois que vous avez configuré les pods pour utiliser le compte de service, ils peuvent accéder à tous Service AWS ceux auxquels le rôle est autorisé à accéder.

1. Créez un fichier de politique pour autoriser l'accès en lecture seule à l'objet Amazon S3 que vous avez chargé :

```
cat >my-policy.json <<EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::<my-spark-jar-bucket>",
                "arn:aws:s3:::<my-spark-jar-bucket>/*"
            ]
        }
    ]
}
EOF
```

2. Créez la politique IAM.

```
aws iam create-policy --policy-name my-policy --policy-document file://my-
policy.json
```

3. Créez un rôle IAM et associez-le à un compte de service Kubernetes pour le pilote Spark

```
eksctl create iamserviceaccount --name my-spark-driver-sa --namespace spark-
operator \
--cluster my-cluster --role-name "my-role" \
--attach-policy-arn arn:aws:iam::111122223333:policy/my-policy --approve
```

4. Créez un fichier YAML avec les autorisations requises pour le compte de service de pilote Spark :

```
cat >spark-rbac.yaml <<EOF
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: default
  name: emr-containers-role-spark
rules:
- apiGroups:
  - ""
  resources:
  - pods
  verbs:
  - "*"
- apiGroups:
  - ""
  resources:
  - services
  verbs:
  - "*"
- apiGroups:
  - ""
  resources:
  - configmaps
  verbs:
  - "*"
- apiGroups:
  - ""
  resources:
  - persistentvolumeclaims
  verbs:
  - "*"
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: spark-role-binding
  namespace: default
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
```

```
  name: emr-containers-role-spark
subjects:
- kind: ServiceAccount
  name: emr-containers-sa-spark
  namespace: default
EOF
```

5. Appliquez les configurations de liaison des rôles du cluster.

```
kubectl apply -f spark-rbac.yaml
```

6. La `kubectl` commande doit renvoyer une confirmation du compte créé.

```
serviceaccount/emr-containers-sa-spark created
clusterrolebinding.rbac.authorization.k8s.io/emr-containers-role-spark configured
```

Exécution de l'application Spark

Amazon EMR en version 6.10.0 et supérieure prend en charge spark-submit pour l'exécution d'applications Spark sur un cluster Amazon EKS. Pour exécuter l'application Spark, procédez comme suit :

1. Assurez-vous d'avoir suivi les étapes décrites dans Configuration de spark-submit pour Amazon EMR sur EKS.
2. Définissez les valeurs des variables d'environnement suivantes :

```
export SPARK_HOME=spark-home
export MASTER_URL=k8s://Amazon EKS-cluster-endpoint
```

3. Maintenant, soumettez l'application Spark avec la commande suivante :

```
$SPARK_HOME/bin/spark-submit \
 --class org.apache.spark.examples.SparkPi \
 --master $MASTER_URL \
 --conf spark.kubernetes.container.image=895885662937.dkr.ecr.us-
west-2.amazonaws.com/spark/emr-6.15.0:latest \
 --conf spark.kubernetes.authenticate.driver.serviceAccountName=emr-containers-sa-
spark \
 --deploy-mode cluster \
 --conf spark.kubernetes.namespace=default \
```

```
 --conf "spark.driver.extraClassPath=/usr/lib/hadoop-lzo/lib/*:/usr/lib/hadoop/
hadoop-aws.jar:/usr/share/aws/aws-java-sdk/*:/usr/share/aws/emr/emrfs/conf:/usr/
share/aws/emr/emrfs/lib/*:/usr/share/aws/emr/emrfs/auxlib/*:/usr/share/aws/emr/
security/conf:/usr/share/aws/emr/security/lib/*:/usr/share/aws/hmclient/lib/aws-
glue-datacatalog-spark-client.jar:/usr/share/java/Hive-JSON-Serde/hive-openx-
serde.jar:/usr/share/aws/sagemaker-spark-sdk/lib/sagemaker-spark-sdk.jar:/home/
hadoop/extrajars/*" \
 --conf "spark.driver.extraLibraryPath=/usr/lib/hadoop/lib/native:/usr/lib/hadoop-
lzo/lib/native:/docker/usr/lib/hadoop/lib/native:/docker/usr/lib/hadoop-lzo/lib/
native" \
 --conf "spark.executor.extraClassPath=/usr/lib/hadoop-lzo/lib/*:/usr/lib/hadoop/
hadoop-aws.jar:/usr/share/aws/aws-java-sdk/*:/usr/share/aws/emr/emrfs/conf:/usr/
share/aws/emr/emrfs/lib/*:/usr/share/aws/emr/emrfs/auxlib/*:/usr/share/aws/emr/
security/conf:/usr/share/aws/emr/security/lib/*:/usr/share/aws/hmclient/lib/aws-
glue-datacatalog-spark-client.jar:/usr/share/java/Hive-JSON-Serde/hive-openx-
serde.jar:/usr/share/aws/sagemaker-spark-sdk/lib/sagemaker-spark-sdk.jar:/home/
hadoop/extrajars/*" \
 --conf "spark.executor.extraLibraryPath=/usr/lib/hadoop/lib/native:/usr/lib/
hadoop-lzo/lib/native:/docker/usr/lib/hadoop/lib/native:/docker/usr/lib/hadoop-lzo/
lib/native" \
 --conf
 spark.hadoop.fs.s3.customAWSCredentialsProvider=com.amazonaws.auth.WebIdentityTokenCredent
 \
 --conf spark.hadoop.fs.s3.impl=com.amazon.ws.emr.hadoop.fs.EmrFileSystem \
 --conf
 spark.hadoop.fs.AbstractFileSystem.s3.impl=org.apache.hadoop.fs.s3.EMRFSDelegate \
 --conf spark.hadoop.fs.s3.buffer.dir=/mnt/s3 \
 --conf spark.hadoop.fs.s3.getObject.initialSocketTimeoutMilliseconds="2000" \
 --conf
 spark.hadoop.mapreduce.fileoutputcommitter.algorithm.version.emr_internal_use_only.EmrFile
 \
 --conf spark.hadoop.mapreduce.fileoutputcommitter.cleanup-
failures.ignored.emr_internal_use_only.EmrFileSystem="true" \
 s3://my-pod-bucket/spark-examples.jar 20
```

4. Une fois que le pilote Spark a terminé la tâche Spark, vous devriez voir une ligne de journal à la fin de la soumission indiquant que la tâche Spark est terminée.

```
23/11/24 17:02:14 INFO LoggingPodStatusWatcherImpl: Application
 org.apache.spark.examples.SparkPi with submission ID default:org-apache-spark-
examples-sparkpi-4980808c03ff3115-driver finished
23/11/24 17:02:14 INFO ShutdownHookManager: Shutdown hook called
```

Nettoyage

Lorsque vous avez terminé d'exécuter vos applications, vous pouvez effectuer le nettoyage à l'aide de la commande suivante.

```
kubectl delete -f spark-rbac.yaml
```

Utilisation d'Apache Livy avec Amazon EMR sur EKS

Avec les versions 7.1.0 et supérieures d'Amazon EMR, vous pouvez utiliser Apache Livy pour soumettre des tâches sur Amazon EMR sur EKS. À l'aide d'Apache Livy, vous pouvez configurer votre propre point de terminaison REST Apache Livy et l'utiliser pour déployer et gérer des applications Spark sur vos clusters Amazon EKS. Après avoir installé Livy dans votre cluster Amazon EKS, vous pouvez utiliser le point de terminaison Livy pour envoyer des applications Spark à votre serveur Livy. Le serveur gère le cycle de vie des applications Spark.

> **Note**
>
> Amazon EMR calcule les tarifs sur Amazon EKS en fonction du vCPU et de la consommation de mémoire. Ce calcul s'applique aux modules pilote et exécuteur. Ce calcul commence à partir du moment où vous téléchargez l'image de votre application Amazon EMR jusqu'à ce que le module Amazon EKS se termine et est arrondi à la seconde près.

Rubriques

Configuration d'Apache Livy pour Amazon EMR sur EKS

Avant de pouvoir installer Apache Livy sur votre cluster Amazon EKS, vous devez installer et configurer un ensemble d'outils prérequis. Il s'agit notamment de l' AWS CLI outil de ligne de commande fondamental pour utiliser les AWS ressources, des outils de ligne de commande pour travailler avec Amazon EKS et d'un contrôleur utilisé dans ce cas d'utilisation pour mettre votre application de cluster à disposition sur Internet et pour acheminer le trafic réseau.

- Installation ou mise à jour vers la dernière version du AWS CLI — Si vous avez déjà installé le AWS CLI, vérifiez que vous disposez de la dernière version.
- Configurer kubectl et eksctl — eksctl est un outil de ligne de commande que vous utilisez pour communiquer avec Amazon EKS.
- Installer Helm – Le gestionnaire de packages Helm pour Kubernetes vous aide à installer et à gérer des applications sur votre cluster Kubernetes.
- Commencez avec Amazon EKS — eksctl — Suivez les étapes pour créer un nouveau cluster Kubernetes avec des nœuds dans Amazon EKS.
- Sélectionnez une étiquette de version Amazon EMR : Apache Livy est compatible avec les versions 7.1.0 et supérieures d'Amazon EMR.
- Installez le contrôleur ALB : le contrôleur ALB gère AWS Elastic Load Balancing pour les clusters Kubernetes. Il crée un AWS Network Load Balancer (NLB) lorsque vous créez une entrée Kubernetes lors de la configuration d'Apache Livy.

Commencer à utiliser Apache Livy sur Amazon EMR sur EKS

Procédez comme suit pour installer Apache Livy. Elles incluent la configuration du gestionnaire de packages, la création d'un espace de noms pour exécuter les charges de travail Spark, l'installation de Livy, la configuration de l'équilibrage de charge et les étapes de vérification. Vous devez suivre ces étapes pour exécuter un traitement par lots avec Spark.

1. Si ce n'est pas déjà fait, configurez Apache Livy pour Amazon EMR sur EKS.
2. Authentifiez votre client Helm dans le registre Amazon ECR. Vous pouvez trouver la `ECR-registry-account` valeur correspondante pour vos comptes Région AWS de registre Amazon ECR par région.

```
aws ecr get-login-password \--region <AWS_REGION> | helm registry login \
--username AWS \
```

```
--password-stdin <ECR-registry-account>.dkr.ecr.<region-id>.amazonaws.com
```

3. La configuration de Livy crée un compte de service pour le serveur Livy et un autre compte pour l'application Spark. Pour configurer l'IRSA pour les comptes de service, consultez la section Configuration des autorisations d'accès avec les rôles IAM pour les comptes de service (IRSA).
4. Créez un espace de noms pour exécuter vos charges de travail Spark.

```
kubectl create ns <spark-ns>
```

5. Utilisez la commande suivante pour installer Livy.

   Ce point de terminaison Livy est uniquement disponible en interne pour le VPC dans le cluster EKS. Pour activer l'accès au-delà du VPC, définissez `–-set loadbalancer.internal=false` votre commande d'installation Helm.

   > **Note**
   >
   > Par défaut, le protocole SSL n'est pas activé dans ce point de terminaison Livy et le point de terminaison n'est visible que dans le VPC du cluster EKS. Si vous définissez `loadbalancer.internal=false` et`ssl.enabled=false`, vous exposez un point de terminaison non sécurisé à l'extérieur de votre VPC. Pour configurer un point de terminaison Livy sécurisé, voir Configuration d'un point de terminaison Apache Livy sécurisé avec TLS/SSL.

```
helm install livy-demo \
  oci://895885662937.dkr.ecr.region-id.amazonaws.com/livy \
  --version 7.12.0 \
  --namespace livy-ns \
  --set image=ECR-registry-account.dkr.ecr.region-id.amazonaws.com/livy/
emr-7.12.0:latest \
  --set sparkNamespace=<spark-ns> \
  --create-namespace
```

   Le résultat suivant doit s'afficher.

```
NAME: livy-demo
LAST DEPLOYED: Mon Mar 18 09:23:23 2024
NAMESPACE: livy-ns
```

```
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
The Livy server has been installed.
Check installation status:
1. Check Livy Server pod is running
  kubectl --namespace livy-ns get pods -l "app.kubernetes.io/instance=livy-demo"
2. Verify created NLB is in Active state and it's target groups are healthy (if
 loadbalancer.enabled is true)

Access LIVY APIs:
    # Ensure your NLB is active and healthy
    # Get the Livy endpoint using command:
    LIVY_ENDPOINT=$(kubectl get svc -n livy-ns -l app.kubernetes.io/
instance=livy-demo,emr-containers.amazonaws.com/type=loadbalancer -o
 jsonpath='{.items[0].status.loadBalancer.ingress[0].hostname}' |  awk '{printf
 "%s:8998\n", $0}')
    # Access Livy APIs using http://$LIVY_ENDPOINT or https://$LIVY_ENDPOINT (if
 SSL is enabled)
    # Note: While uninstalling Livy, makes sure the ingress and NLB are deleted
 after running the helm command to avoid dangling resources
```

Les noms de compte de service par défaut pour le serveur Livy et la session Spark sont `emr-containers-sa-livy` et`emr-containers-sa-spark-livy`. Pour utiliser des noms personnalisés, utilisez les `sparkServiceAccount.name` paramètres `serviceAccounts.name` et.

```
--set serviceAccounts.name=my-service-account-for-livy
--set sparkServiceAccount.name=my-service-account-for-spark
```

6. Vérifiez que vous avez installé le graphique Helm.

```
helm list -n livy-ns -o yaml
```

La `helm list` commande doit renvoyer des informations sur votre nouveau graphique Helm.

```
app_version: 0.7.1-incubating
chart: livy-emr-7.12.0
name: livy-demo
namespace: livy-ns
```

```
revision: "1"
status: deployed
updated: 2024-02-08 22:39:53.539243 -0800 PST
```

7. Vérifiez que le Network Load Balancer est actif.

```
LIVY_NAMESPACE=<livy-ns>
LIVY_APP_NAME=<livy-app-name>
AWS_REGION=<AWS_REGION>

# Get the NLB Endpoint URL
NLB_ENDPOINT=$(kubectl --namespace $LIVY_NAMESPACE get svc -l "app.kubernetes.io/
instance=$LIVY_APP_NAME,emr-containers.amazonaws.com/type=loadbalancer" -o
 jsonpath='{.items[0].status.loadBalancer.ingress[0].hostname}')

# Get all the load balancers in the account's region
ELB_LIST=$(aws elbv2 describe-load-balancers --region $AWS_REGION)

# Get the status of the NLB that matching the endpoint from the Kubernetes service
NLB_STATUS=$(echo $ELB_LIST | grep -A 8 "\"DNSName\": \"$NLB_ENDPOINT\"" | awk '/
Code/{print $2}/}/' | tr -d '"},\n')
echo $NLB_STATUS
```

8. Vérifiez maintenant que le groupe cible du Network Load Balancer est sain.

```
LIVY_NAMESPACE=<livy-ns>
LIVY_APP_NAME=<livy-app-name>
AWS_REGION=<AWS_REGION>

# Get the NLB endpoint
NLB_ENDPOINT=$(kubectl --namespace $LIVY_NAMESPACE get svc -l "app.kubernetes.io/
instance=$LIVY_APP_NAME,emr-containers.amazonaws.com/type=loadbalancer" -o
 jsonpath='{.items[0].status.loadBalancer.ingress[0].hostname}')

# Get all the load balancers in the account's region
ELB_LIST=$(aws elbv2 describe-load-balancers --region $AWS_REGION)

# Get the NLB ARN from the NLB endpoint
NLB_ARN=$(echo $ELB_LIST | grep -B 1 "\"DNSName\": \"$NLB_ENDPOINT\"" | awk
 '/"LoadBalancerArn":/,/"/'| awk '/:/{print $2}' | tr -d \",)

# Get the target group from the NLB. Livy setup only deploys 1 target group
```

```
TARGET_GROUP_ARN=$(aws elbv2 describe-target-groups --load-balancer-arn $NLB_ARN
 --region $AWS_REGION | awk '/"TargetGroupArn":/,/"/'| awk '/:/{print $2}' | tr -d
 \",)

# Get health of target group
aws elbv2 describe-target-health --target-group-arn $TARGET_GROUP_ARN
```

Voici un exemple de sortie qui montre l'état du groupe cible :

```
{
    "TargetHealthDescriptions": [
        {
            "Target": {
                "Id": "<target IP>",
                "Port": 8998,
                "AvailabilityZone": "us-west-2d"
            },
            "HealthCheckPort": "8998",
            "TargetHealth": {
                "State": "healthy"
            }
        }
    ]
}
```

Une fois que le statut de votre NLB sera atteint `active` et que votre groupe cible l'est`healthy`, vous pouvez continuer. Cette opération peut durer quelques minutes.

9. Récupérez le point de terminaison Livy depuis l'installation de Helm. La sécurité de votre point de terminaison Livy dépend de l'activation ou non du protocole SSL.

```
LIVY_NAMESPACE=<livy-ns>
 LIVY_APP_NAME=livy-app-name
 LIVY_ENDPOINT=$(kubectl get svc -n livy-ns -l app.kubernetes.io/
instance=livy-app-name,emr-containers.amazonaws.com/type=loadbalancer -o
 jsonpath='{.items[0].status.loadBalancer.ingress[0].hostname}' |  awk '{printf
 "%s:8998\n", $0}')
 echo "$LIVY_ENDPOINT"
```

10. Récupérez le compte de service Spark depuis l'installation de Helm

```
SPARK_NAMESPACE=spark-ns
LIVY_APP_NAME=<livy-app-name>
SPARK_SERVICE_ACCOUNT=$(kubectl --namespace $SPARK_NAMESPACE
 get sa -l "app.kubernetes.io/instance=$LIVY_APP_NAME" -o
 jsonpath='{.items[0].metadata.name}')
echo "$SPARK_SERVICE_ACCOUNT"
```

Le résultat doit ressembler à celui-ci :

```
emr-containers-sa-spark-livy
```

11. Si vous avez configuré `internalALB=true` pour activer l'accès depuis l'extérieur de votre VPC, créez une EC2 instance Amazon et assurez-vous que le Network Load Balancer autorise le trafic réseau provenant de l'instance. EC2 Vous devez le faire pour que l'instance ait accès à votre point de terminaison Livy. Pour plus d'informations sur l'exposition sécurisée de votre point de terminaison en dehors de votre VPC, consultez Configuration avec un point de terminaison Apache Livy sécurisé avec TLS/SSL.
12. L'installation de Livy crée le compte de service `emr-containers-sa-spark` pour exécuter les applications Spark. Si votre application Spark utilise des AWS ressources telles que S3 ou appelle des opérations d' AWS API ou de CLI, vous devez associer un rôle IAM doté des autorisations nécessaires à votre compte de service Spark. Pour plus d'informations, voir Configuration des autorisations d'accès avec les rôles IAM pour les comptes de service (IRSA).

Apache Livy prend en charge des configurations supplémentaires que vous pouvez utiliser lors de l'installation de Livy. Pour plus d'informations, consultez la section Propriétés d'installation d'Apache Livy sur Amazon EMR sur les versions EKS.

Exécution d'une application Spark avec Apache Livy pour Amazon EMR sur EKS

Avant de pouvoir exécuter une application Spark avec Apache Livy, assurez-vous d'avoir suivi les étapes décrites dans Configuration d'Apache Livy pour Amazon EMR sur EKS et Démarrage avec Apache Livy pour Amazon EMR sur EKS.

Vous pouvez utiliser Apache Livy pour exécuter deux types d'applications :

- Sessions par lots : type de charge de travail Livy permettant de soumettre des tâches par lots Spark.
- Sessions interactives : type de charge de travail Livy qui fournit une interface programmatique et visuelle pour exécuter des requêtes Spark.

> **Note**
>
> Les pods pilote et exécuteur issus de différentes sessions peuvent communiquer entre eux. Les espaces de noms ne garantissent aucune sécurité entre les pods. Kubernetes n'autorise pas d'autorisations sélectives sur un sous-ensemble de pods au sein d'un espace de noms donné.

Exécution de sessions par lots

Pour soumettre un traitement par lots, utilisez la commande suivante.

```
curl -s -k -H 'Content-Type: application/json' -X POST \
      -d '{
            "name": "my-session",
            "file": "entryPoint_location (S3 or local)",
            "args": ["argument1", "argument2", ...],
            "conf": {
                "spark.kubernetes.namespace": "<spark-namespace>",
                "spark.kubernetes.container.image": "public.ecr.aws/emr-on-eks/spark/
emr-7.12.0:latest",
                "spark.kubernetes.authenticate.driver.serviceAccountName": "<spark-
service-account>"
            }
          }' <livy-endpoint>/batches
```

Pour surveiller votre traitement par lots, utilisez la commande suivante.

```
curl -s -k -H 'Content-Type: application/json' -X GET <livy-endpoint>/batches/my-
session
```

Organisation de sessions interactives

Pour exécuter des sessions interactives avec Apache Livy, suivez les étapes ci-dessous.

1. Assurez-vous d'avoir accès à un bloc-notes Jupyter auto-hébergé ou géré, tel qu'un bloc-notes Jupyter SageMaker AI. Sparkmagic doit être installé sur votre bloc-notes Jupyter.
2. Créez un bucket pour la configuration de Spark`spark.kubernetes.file.upload.path`. Assurez-vous que le compte de service Spark dispose d'un accès en lecture et en écriture au bucket. Pour plus de détails sur la configuration de votre compte de service Spark, consultez Configuration des autorisations d'accès avec les rôles IAM pour les comptes de service (IRSA)
3. Chargez sparkmagic dans le bloc-notes Jupyter à l'aide de la commande. `%load_ext sparkmagic.magics`
4. Exécutez la commande `%manage_spark` pour configurer votre point de terminaison Livy avec le bloc-notes Jupyter. Choisissez l'onglet Ajouter des points de terminaison, choisissez le type d'authentification configuré, ajoutez le point de terminaison Livy au bloc-notes, puis choisissez Ajouter un point de terminaison.
5. Exécutez à `%manage_spark` nouveau pour créer le contexte Spark, puis accédez à la session Create. Choisissez le point de terminaison Livy, spécifiez un nom de session unique, choisissez une langue, puis ajoutez les propriétés suivantes.

```
{
  "conf": {
    "spark.kubernetes.namespace": "livy-namespace",
    "spark.kubernetes.container.image": "public.ecr.aws/emr-on-eks/spark/
emr-7.12.0:latest",
    "spark.kubernetes.authenticate.driver.serviceAccountName": "<spark-service-
account>",
    "spark.kubernetes.file.upload.path": "<URI_TO_S3_LOCATION_>"
  }
}
```

6. Soumettez l'application et attendez qu'elle crée le contexte Spark.
7. Pour contrôler l'état de la session interactive, exécutez la commande suivante.

```
curl -s -k -H 'Content-Type: application/json' -X GET livy-endpoint/sessions/my-
interactive-session
```

Surveillance des applications Spark

Pour suivre la progression de vos applications Spark avec l'interface utilisateur Livy, utilisez le lien`http://<livy-endpoint>/ui`.

Désinstaller Apache Livy avec Amazon EMR sur EKS

Pour désinstaller Apache Livy, procédez comme suit.

1. Supprimez la configuration de Livy en utilisant les noms de votre espace de noms et le nom de votre application. Dans cet exemple, le nom de l'application est `livy-demo` et l'espace de noms est`livy-ns`.

   ```
   helm uninstall livy-demo -n livy-ns
   ```

2. Lors de la désinstallation, Amazon EMR on EKS supprime le service Kubernetes dans Livy, les équilibreurs de charge et AWS les groupes cibles que vous avez créés lors de l'installation. La suppression de ressources peut prendre quelques minutes. Assurez-vous que les ressources sont supprimées avant de réinstaller Livy sur l'espace de noms.
3. Supprimez l'espace de noms Spark.

   ```
   kubectl delete namespace spark-ns
   ```

Sécurité pour Apache Livy avec Amazon EMR sur EKS

Consultez les rubriques suivantes pour en savoir plus sur la configuration de la sécurité pour Apache Livy avec Amazon EMR sur EKS. Ces options incluent l'utilisation de la sécurité de la couche de transport, le contrôle d'accès basé sur les rôles, c'est-à-dire l'accès basé sur le rôle d'une personne au sein d'une organisation, et l'utilisation des rôles IAM, qui fournissent un accès aux ressources, en fonction des autorisations accordées.

Rubriques

- Configuration d'un point de terminaison Apache Livy sécurisé avec TLS/SSL
- Configuration des autorisations des applications Apache Livy et Spark avec le contrôle d'accès basé sur les rôles (RBAC)
- Configuration des autorisations d'accès avec des rôles IAM pour les comptes de service (IRSA)

Configuration d'un point de terminaison Apache Livy sécurisé avec TLS/SSL

Consultez les sections suivantes pour en savoir plus sur la configuration d'Apache Livy pour Amazon EMR sur EKS end-to-end avec le chiffrement TLS et SSL.

Configuration du chiffrement TLS et SSL

Pour configurer le chiffrement SSL sur votre point de terminaison Apache Livy, procédez comme suit.

- Installez le pilote CSI Secrets Store et le fournisseur de AWS secrets et de configuration (ASCP) : le pilote CSI et l'ASCP de Secrets Store stockent en toute sécurité les certificats JKS et les mots de passe de Livy dont le pod de serveur Livy a besoin pour activer le protocole SSL. Vous pouvez également installer uniquement le pilote CSI Secrets Store et utiliser n'importe quel autre fournisseur de secrets compatible.
- Créez un certificat ACM : ce certificat est nécessaire pour sécuriser la connexion entre le client et le point de terminaison ALB.
- Configurez un certificat JKS, un mot de passe clé et un mot de passe de banque de clés pour AWS Secrets Manager : nécessaire pour sécuriser la connexion entre le point de terminaison ALB et le serveur Livy.
- Ajoutez des autorisations au compte de service Livy pour récupérer les secrets AWS Secrets Manager : le serveur Livy a besoin de ces autorisations pour récupérer les secrets depuis ASCP et ajouter les configurations Livy pour sécuriser le serveur Livy. Pour ajouter des autorisations IAM à un compte de service, voir Configuration des autorisations d'accès avec des rôles IAM pour les comptes de service (IRSA).

Configuration d'un certificat JKS avec une clé et un mot de passe de keystore pour AWS Secrets Manager

Procédez comme suit pour configurer un certificat JKS avec une clé et un mot de passe pour le keystore.

1. Générez un fichier keystore pour le serveur Livy.

```
keytool -genkey -alias <host> -keyalg RSA -keysize 2048 -dname
 CN=<host>,OU=hw,O=hw,L=<your_location>,ST=<state>,C=<country> -
keypass <keyPassword> -keystore <keystore_file> -storepass <storePassword> --
validity 3650
```

2. Créez un certificat.

```
keytool -export -alias <host> -keystore mykeystore.jks -rfc -
file mycertificate.cert -storepass <storePassword>
```

3. Créez un fichier Truststore.

```
keytool -import -noprompt -alias <host>-file <cert_file> -
keystore <truststore_file> -storepass <truststorePassword>
```

4. Enregistrez le certificat JKS dans. AWS Secrets Manager`livy-jks-secret`Remplacez-le par votre code secret et `fileb://mykeystore.jks` par le chemin d'accès à votre certificat JKS de keystore.

```
aws secretsmanager create-secret \
--name livy-jks-secret \
--description "My Livy keystore JKS secret" \
--secret-binary fileb://mykeystore.jks
```

5. Enregistrez le keystore et le mot de passe clé dans Secrets Manager. Assurez-vous d'utiliser vos propres paramètres.

```
aws secretsmanager create-secret \
--name livy-jks-secret \
--description "My Livy key and keystore password secret" \
--secret-string "{\"keyPassword\":\"<test-key-password>\",\"keyStorePassword\":
\"<test-key-store-password>\"}"
```

6. Créez un espace de noms de serveur Livy à l'aide de la commande suivante.

```
kubectl create ns <livy-ns>
```

7. Créez l'`ServiceProviderClass`objet pour le serveur Livy qui possède le certificat JKS et les mots de passe.

```
cat >livy-secret-provider-class.yaml << EOF
apiVersion: secrets-store.csi.x-k8s.io/v1
kind: SecretProviderClass
metadata:
  name: aws-secrets
spec:
  provider: aws
  parameters:
    objects: |
        - objectName: "livy-jks-secret"
          objectType: "secretsmanager"
        - objectName: "livy-passwords"
          objectType: "secretsmanager"
```

```
EOF
kubectl apply -f livy-secret-provider-class.yaml -n <livy-ns>
```

Commencer à utiliser Apache Livy compatible SSL

Après avoir activé le SSL sur votre serveur Livy, vous devez configurer le `serviceAccount` pour avoir accès aux `keyPasswords` secrets `keyStore` et. AWS Secrets Manager

1. Créez l'espace de noms du serveur Livy.

```
kubectl create namespace <livy-ns>
```

2. Configurez le compte de service Livy pour avoir accès aux secrets dans Secrets Manager. Pour plus d'informations sur la configuration de l'IRSA, voir Configuration de l'IRSA lors de l'installation d'Apache Livy.

```
aws ecr get-login-password \--region region-id | helm registry login \
--username AWS \
--password-stdin ECR-registry-account.dkr.ecr.region-id.amazonaws.com
```

3. Installez Livy. Pour le paramètre Helm chart --version, utilisez votre étiquette de version Amazon EMR, telle que. `7.1.0` Vous devez également remplacer l'ID de compte de registre Amazon ECR et l'ID de région par les vôtres. IDs Vous pouvez trouver la `ECR-registry-account` valeur correspondante pour vos comptes Région AWS de registre Amazon ECR par région.

```
helm install <livy-app-name> \
  oci://895885662937.dkr.ecr.region-id.amazonaws.com/livy \
  --version 7.12.0 \
  --namespace livy-namespace-name \
  --set image=<ECR-registry-account.dkr.ecr>.<region>.amazonaws.com/livy/
emr-7.12.0:latest \
  --set sparkNamespace=spark-namespace \
  --set ssl.enabled=true
  --set ssl.CertificateArn=livy-acm-certificate-arn
  --set ssl.secretProviderClassName=aws-secrets
  --set ssl.keyStoreObjectName=livy-jks-secret
  --set ssl.keyPasswordsObjectName=livy-passwords
  --create-namespace
```

4. Continuez à partir de l'étape 5 de l'installation d'Apache Livy sur Amazon EMR sur EKS.

Configuration des autorisations des applications Apache Livy et Spark avec le contrôle d'accès basé sur les rôles (RBAC)

Pour déployer Livy, Amazon EMR on EKS crée un compte et un rôle de service de serveur, ainsi qu'un compte et un rôle de service Spark. Ces rôles doivent disposer des autorisations RBAC nécessaires pour terminer la configuration et exécuter les applications Spark.

Autorisations RBAC pour le compte de service et le rôle du serveur

Amazon EMR on EKS crée le compte de service et le rôle du serveur Livy pour gérer les sessions Livy pour les tâches Spark et acheminer le trafic vers et depuis l'entrée et d'autres ressources.

Le nom par défaut de ce compte de service est `emr-containers-sa-livy`. Il doit disposer des autorisations suivantes.

```
rules:
- apiGroups:
  - ""
  resources:
  - "namespaces"
  verbs:
  - "get"
- apiGroups:
  - ""
  resources:
  - "serviceaccounts"
    "services"
    "configmaps"
    "events"
    "pods"
    "pods/log"
  verbs:
  - "get"
    "list"
    "watch"
    "describe"
    "create"
    "edit"
    "delete"
    "deletecollection"
    "annotate"
    "patch"
```

```
    "label"
 - apiGroups:
   - ""
   resources:
   - "secrets"
   verbs:
   - "create"
     "patch"
     "delete"
     "watch"
 - apiGroups:
   - ""
   resources:
   - "persistentvolumeclaims"
   verbs:
   - "get"
     "list"
     "watch"
     "describe"
     "create"
     "edit"
     "delete"
     "annotate"
     "patch"
     "label"
```

Autorisations RBAC pour le compte et le rôle du service Spark

Le pod du pilote Spark a besoin d'un compte de service Kubernetes dans le même espace de noms que le pod. Ce compte de service a besoin d'autorisations pour gérer les modules d'exécution et toutes les ressources requises par le module de pilotes. À moins que le compte de service par défaut de l'espace de noms ne dispose des autorisations requises, le pilote échoue et se ferme. Les autorisations RBAC suivantes sont requises.

```
rules:
- apiGroups:
  - ""
    "batch"
    "extensions"
    "apps"
  resources:
  - "configmaps"
    "serviceaccounts"
```

```
    "events"
    "pods"
    "pods/exec"
    "pods/log"
    "pods/portforward"
    "secrets"
    "services"
    "persistentvolumeclaims"
    "statefulsets"
  verbs:
  - "create"
    "delete"
    "get"
    "list"
    "patch"
    "update"
    "watch"
    "describe"
    "edit"
    "deletecollection"
    "patch"
    "label"
```

Configuration des autorisations d'accès avec des rôles IAM pour les comptes de service (IRSA)

Par défaut, le serveur Livy et le pilote et les exécuteurs de l'application Spark n'ont pas accès aux AWS ressources. Le compte de service du serveur et le compte de service Spark contrôlent l'accès aux AWS ressources pour le serveur Livy et les pods de l'application Spark. Pour accorder l'accès, vous devez associer les comptes de service à un rôle IAM disposant des AWS autorisations nécessaires.

Vous pouvez configurer le mappage IRSA avant d'installer Apache Livy, pendant l'installation ou après l'avoir terminée.

Configuration de l'IRSA lors de l'installation d'Apache Livy (pour le compte de service du serveur)

> **Note**
>
> Ce mappage n'est pris en charge que pour le compte de service du serveur.

1. Assurez-vous que vous avez terminé de configurer Apache Livy pour Amazon EMR sur EKS et que vous êtes en train d'installer Apache Livy avec Amazon EMR sur EKS.
2. Créez un espace de noms Kubernetes pour le serveur Livy. Dans cet exemple, le nom de l'espace de noms est `livy-ns`.
3. Créez une politique IAM qui inclut les Services AWS autorisations auxquelles vous souhaitez que vos pods accèdent. L'exemple suivant crée une politique IAM permettant d'obtenir des ressources Amazon S3 pour le point d'entrée Spark.

```
cat >my-policy.json <<EOF{
"Version": "2012-10-17",
    "Statement": [
        {
"Effect": "Allow",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::my-spark-entrypoint-bucket"
        }
    ]
}
EOF

aws iam create-policy --policy-name my-policy --policy-document file://my-policy.json
```

4. Utilisez la commande suivante pour attribuer une variable à votre Compte AWS identifiant.

```
account_id=$(aws sts get-caller-identity --query "Account" --output text)
```

5. Définissez le fournisseur d'identité OpenID Connect (OIDC) de votre cluster sur une variable d'environnement.

```
oidc_provider=$(aws eks describe-cluster --name my-cluster --region $AWS_REGION --query "cluster.identity.oidc.issuer" --output text | sed -e "s/^https:\/\///")
```

6. Définissez des variables pour l'espace de noms et le nom du compte de service. Veillez à utiliser vos propres valeurs.

```
export namespace=default
export service_account=my-service-account
```

7. Créez un fichier de politique de confiance à l'aide de la commande suivante. Si vous souhaitez accorder l'accès au rôle à tous les comptes de service d'un espace de noms,

copiez la commande suivante, remplacez `StringEquals` par `StringLike` et remplacez `$service_account` par*.

```
cat >trust-relationship.json <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::$account_id:oidc-provider/$oidc_provider"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "$oidc_provider:aud": "sts.amazonaws.com",
          "$oidc_provider:sub": "system:serviceaccount:$namespace:$service_account"
        }
      }
    }
  ]
}
EOF
```

8. Créez le rôle.

```
aws iam create-role --role-name my-role --assume-role-policy-document file://trust-
relationship.json --description "my-role-description"
```

9. Utilisez la commande d'installation Helm suivante pour configurer IRSA `serviceAccount.executionRoleArn` pour mapper l'IRSA. Voici un exemple de commande d'installation de Helm. Vous pouvez trouver la `ECR-registry-account` valeur correspondante pour vos comptes Région AWS de registre Amazon ECR par région.

```
helm install livy-demo \
  oci://895885662937.dkr.ecr.us-west-2.amazonaws.com/livy \
  --version 7.12.0 \
  --namespace livy-ns \
  --set image=ECR-registry-account.dkr.ecr.region-id.amazonaws.com/livy/
emr-7.12.0:latest \
  --set sparkNamespace=spark-ns \
  --set serviceAccount.executionRoleArn=arn:aws:iam::123456789012:role/my-role
```

Associer IRSA à un compte de service Spark

Avant de mapper IRSA à un compte de service Spark, assurez-vous d'avoir effectué les tâches suivantes :

- Assurez-vous que vous avez terminé de configurer Apache Livy pour Amazon EMR sur EKS et que vous êtes en train d'installer Apache Livy avec Amazon EMR sur EKS.
- Vous devez disposer d'un fournisseur IAM OpenID Connect (OIDC) existant pour votre cluster. Pour savoir si vous en avez déjà un ou comment en créer un, voir Créer un fournisseur IAM OIDC pour votre cluster.
- Assurez-vous d'avoir installé la version 0.171.0 ou ultérieure de la `eksctl` CLI installée ou. AWS CloudShell Pour installer ou mettre à jour`eksctl`, consultez la section Installation de la `eksctl` documentation.

Pour associer IRSA à votre compte de service Spark, procédez comme suit :

1. Utilisez la commande suivante pour obtenir le compte de service Spark.

```
SPARK_NAMESPACE=<spark-ns>
LIVY_APP_NAME=<livy-app-name>
kubectl --namespace $SPARK_NAMESPACE describe sa -l "app.kubernetes.io/instance=
$LIVY_APP_NAME" | awk '/^Name:/ {print $2}'
```

2. Définissez vos variables pour l'espace de noms et le nom du compte de service.

```
export namespace=default
export service_account=my-service-account
```

3. Utilisez la commande suivante pour créer un fichier de politique de confiance pour le rôle IAM. L'exemple suivant autorise tous les comptes de service de l'espace de noms à utiliser le rôle. Pour ce faire, remplacez `StringEquals` par `StringLike` et remplacez `$service_account` par *.

```
cat >trust-relationship.json <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
```

```
        "Federated": "arn:aws:iam::$account_id:oidc-provider/$oidc_provider"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "$oidc_provider:aud": "sts.amazonaws.com",
          "$oidc_provider:sub": "system:serviceaccount:$namespace:$service_account"
        }
      }
    }
  ]
}
EOF
```

4. Créez le rôle.

```
aws iam create-role --role-name my-role --assume-role-policy-document file://trust-relationship.json --description "my-role-description"
```

5. Mappez le serveur ou le compte de service Spark à l'aide de la `eksctl` commande suivante. Assurez-vous d'utiliser vos propres valeurs.

```
eksctl create iamserviceaccount --name spark-sa \
--namespace spark-namespace --cluster livy-eks-cluster \
--attach-role-arn arn:aws:iam::0123456789012:role/my-role \
--approve --override-existing-serviceaccounts
```

Propriétés d'installation d'Apache Livy sur Amazon EMR sur les versions EKS

L'installation d'Apache Livy vous permet de sélectionner une version du graphique Livy Helm. Le diagramme Helm propose une variété de propriétés pour personnaliser votre expérience d'installation et de configuration. Ces propriétés sont prises en charge pour Amazon EMR sur EKS versions 7.1.0 et supérieures.

Rubriques

- Propriétés d'installation d'Amazon EMR 7.1.0

Propriétés d'installation d'Amazon EMR 7.1.0

Le tableau suivant décrit toutes les propriétés Livy prises en charge. Lors de l'installation d'Apache Livy, vous pouvez choisir la version du graphique Livy Helm. Pour définir une propriété lors de l'installation, utilisez la commande`--set <property>=<value>`.

Propriété	Description	Par défaut
image	L'URI de version Amazon EMR du serveur Livy. Il s'agit d'une configuration obligatoire.	""
Espace de noms Spark	Espace de noms pour exécuter des sessions Livy Spark. Par exemple, spécifiez « livy ». Il s'agit d'une configuration obligatoire.	""
Nom Override	Entrez un nom au lieu de`livy`. Le nom est défini comme étiquette pour toutes les ressources Livy	« Livy »
Nom complet Override	Indiquez un nom à utiliser au lieu du nom complet des ressources.	""
activé par SSL	Active le end-to-end protocole SSL depuis le point de terminaison Livy vers le serveur Livy.	FALSE
Certificat SSL ARN	Si le protocole SSL est activé, il s'agit de l'ARN du certificat ACM pour le NLB créé par le service.	""
ssl. secretProviderClassNom	Si le protocole SSL est activé, il s'agit du nom de classe du	""

Propriété	Description	Par défaut
	fournisseur secret permettant de sécuriser le NLB pour la connexion au serveur Livy avec SSL.	
ssl. keyStoreObjectNom	Si le protocole SSL est activé, le nom de l'objet du certificat keystore dans la classe du fournisseur secret.	""
ssl. keyPasswordsObjectNom	Si le protocole SSL est activé, nom de l'objet du secret contenant le keystore et le mot de passe clé.	""
rbac.create	Si vrai, crée des ressources RBAC.	FALSE
Compte de service. Créer	Si c'est vrai, crée un compte de service Livy.	TRUE
Nom du compte de service	Le nom du compte de service à utiliser pour Livy. Si vous ne définissez pas cette propriété et ne créez pas de compte de service, Amazon EMR sur EKS génère automatiquement un nom à l'aide de la propriété `fullname` override.	"emr-containers-sa-livy"
Compte de service. execution RoleArn	L'ARN du rôle d'exécution du compte de service Livy.	""
sparkServiceAccount.créer	SI c'est vrai, crée le compte de service Spark dans `.Release.Namespace`	TRUE

Propriété	Description	Par défaut
sparkServiceAccount.nom	Le nom du compte de service à utiliser pour Spark. Si vous ne définissez pas cette propriété et ne créez pas de compte de service Spark, Amazon EMR sur EKS génère automatiquement un nom avec le suffixe de la `fullnameOverride` propriété. `-spark-livy`	« emr-containers-sa-spark - Livy »
nom du service	Nom du service Livy	`"emr-containers-livy"`
service.annotations	Annotations du service Livy	`{}`
loadbalancer.enabled	S'il faut créer un équilibreur de charge pour le service Livy utilisé pour exposer le point de terminaison Livy en dehors du cluster Amazon EKS.	`FALSE`
équilibreur de charge interne	S'il faut configurer le point de terminaison Livy en tant qu'interne au VPC ou externe. La définition de cette propriété pour `FALSE` exposer le point de terminaison à des sources extérieures au VPC. Nous vous recommandons de sécuriser votre terminal avec le protocole TLS/SSL. Pour plus d'informations, consultez la section Configuration du chiffrement TLS et SSL.	`FALSE`

Propriété	Description	Par défaut
imagePullSecrets	La liste des `imagePull Secret` noms à utiliser pour extraire une image de Livy depuis des référentiels privés.	[]
resources	Les demandes de ressource s et les limites pour les conteneurs Livy.	{}
nodeSelector	Les nœuds pour lesquels planifier les modules Livy.	{}
tolérances	Une liste contenant les tolérances à définir pour les pods Livy.	[]
affinité	Les règles d'affinité des Livy Pods.	{}
persistence.enabled	Si vrai, active la persistan ce pour les répertoires de sessions.	FALSE
Persistence.subPath	Le sous-chemin PVC à monter dans les répertoires de sessions.	""
Persistance. Réclamation existante	Le PVC à utiliser au lieu d'en créer un nouveau.	{}

Propriété	Description	Par défaut
Persistance. Classe de stockage	Classe de stockage à utiliser. Pour définir ce paramètre, utilisez le format`storageClassName: <storageClass>` . La définition de ce paramètre pour "-" désactiver le provisionnement dynamique. Si vous définissez ce paramètre sur null ou si vous ne spécifiez rien, Amazon EMR sur EKS ne définit pas de fournisseur storageClassName et utilise le fournisseur par défaut.	""
Persistance. Mode d'accès	Le mode d'accès au PVC.	`ReadWriteOnce`
persistence.size	La taille du PVC.	20 Gi
persistence.annotations	Annotations supplémentaires pour le PVC.	{}
env. *	Envs supplémentaires à régler sur le conteneur Livy. Pour plus d'informations, voir Saisir vos propres configurations Livy et Spark lors de l'installation de Livy.	{}
Env à partir de. *	Environnements supplémentaires à définir sur Livy à partir d'une carte de configuration ou d'un secret de Kubernetes.	[]

Propriété	Description	Par défaut
LivyConf. *	Entrées livy.conf supplémentaires à définir à partir d'une carte ou d'un secret de configuration Kubernetes monté.	{}
sparkDefaultsConf.*	`spark-defaults.conf` Entrées supplémentaires à définir à partir d'une carte ou d'un secret de configuration Kubernetes monté.	{}

Résoudre les erreurs de format courantes liées aux variables d'environnement

Lorsque vous saisissez des configurations Livy et Spark, certains formats de variables d'environnement ne sont pas pris en charge et peuvent provoquer des erreurs. La procédure vous guide à travers une série d'étapes pour vous assurer que vous utilisez les bons formats.

Entrez vos propres configurations Livy et Spark lors de l'installation de Livy

Vous pouvez configurer n'importe quelle variable d'environnement Apache Livy ou Apache Spark avec la propriété `env.*` Helm. Suivez les étapes ci-dessous pour convertir l'exemple de configuration dans un format `example.config.with-dash.withUppercase` de variable d'environnement pris en charge.

1. Remplacez les lettres majuscules par un 1 et une minuscule de la lettre. Par exemple, `example.config.with-dash.withUppercase` devient `example.config.with-dash.with1uppercase`.
2. Remplacez les tirets (-) par 0. Par exemple, `example.config.with-dash.with1uppercase` devient `example.config.with0dash.with1uppercase`
3. Remplacez les points (.) par des traits de soulignement (_). Par exemple, `example.config.with0dash.with1uppercase` devient `example_config_with0dash_with1uppercase`.

4. Remplacez toutes les lettres minuscules par des lettres majuscules.
5. Ajoutez le préfixe `LIVY_` au nom de la variable.
6. Utilisez la variable lors de l'installation de Livy via le graphique de barre en utilisant le format --set env. *YOUR_VARIABLE_NAME*.valeur= *yourvalue*

Par exemple, pour définir les configurations Livy et Spark `livy.server.recovery.state-store = filesystem` et utiliser `spark.kubernetes.executor.podNamePrefix = my-prefix` les propriétés Helm suivantes :

```
–set env.LIVY_LIVY_SERVER_RECOVERY_STATE0STORE.value=filesystem
–set env.LIVY_SPARK_KUBERNETES_EXECUTOR_POD0NAME0PREFIX.value=myprefix
```

Gestion des exécutions de tâches sur Amazon EMR on EKS

Les sections suivantes couvrent des sujets qui vous aident à gérer vos exécutions de tâches Amazon EMR on EKS. Il s'agit notamment de configurer les paramètres d'exécution des tâches lorsque vous utilisez le AWS CLI, de configurer le mode de stockage des données de vos journaux, d'exécuter des scripts Spark SQL pour exécuter des requêtes, de comprendre les états d'exécution des tâches et de savoir comment surveiller les tâches. Vous pouvez parcourir ces rubriques, généralement dans l'ordre, si vous souhaitez configurer et terminer une exécution de travail pour traiter des données.

Rubriques

- Gérer les exécutions de tâches à l'aide du AWS CLI
- Exécution de scripts Spark SQL via l' StartJobRun API
- États d'exécution de la tâche
- Affichage des tâches dans la console Amazon EMR
- Erreurs courantes lors de l'exécution de tâches

Gérer les exécutions de tâches à l'aide du AWS CLI

Cette rubrique explique comment gérer les exécutions de tâches avec le AWS Command Line Interface (AWS CLI). Il décrit en détail les propriétés, telles que les paramètres de sécurité, le pilote et les divers paramètres de remplacement. Il inclut également des sous-rubriques qui couvrent les différentes manières de configurer la journalisation.

Rubriques

Options de configuration d'une exécution de tâche

Utilisez les options suivantes pour configurer les paramètres d'exécution des tâches :

- `--execution-role-arn` : vous devez indiquer un rôle IAM utilisé pour exécuter des tâches. Pour de plus amples informations, veuillez consulter Utilisation des rôles d'exécution de tâches avec Amazon EMR on EKS.
- `--release-label` : vous pouvez déployer Amazon EMR on EKS à l'aide d'Amazon EMR en version 5.32.0 et 6.2.0 ou ultérieure. Amazon EMR on EKS n'est pas pris en charge dans les versions précédentes d'Amazon EMR. Pour de plus amples informations, veuillez consulter Versions Amazon EMR on EKS.
- `--job-driver` : le pilote de tâche est utilisé pour fournir des informations sur la tâche principale. Il s'agit d'un champ de type union dans lequel vous ne pouvez transmettre qu'une seule des valeurs correspondant au type de tâche que vous souhaitez exécuter. Les types de tâches prises en charge incluent :
  - Tâches de soumission Spark : utilisées pour exécuter une commande via la soumission Spark. Vous pouvez utiliser ce type de tâche pour exécuter Scala, SparkR PySpark, SparkSQL et toute autre tâche prise en charge par le biais de Spark Submit. Ce type de tâche a les paramètres suivants :
    - Point d'entrée : il s'agit de la référence HCFS (système de fichiers compatible Hadoop) au jar/py fichier principal que vous souhaitez exécuter.
    - EntryPointArguments - Il s'agit d'un tableau d'arguments que vous souhaitez transmettre à votre jar/py fichier principal. Vous devez gérer la lecture de ces paramètres à l'aide de votre code entrypoint. Chaque argument du tableau doit être séparé par une virgule. EntryPointArguments ne peut pas contenir de crochets ou de parenthèses, tels que (), {} ou [].

- SparkSubmitParameters - Il s'agit des paramètres Spark supplémentaires que vous souhaitez envoyer à la tâche. Utilisez ce paramètre pour remplacer les propriétés Spark par défaut, telles que la mémoire du pilote ou le nombre d'exécuteurs tels que —conf ou —class. Pour plus d'informations, consultez la rubrique Lancement d'applications à l'aide de spark-submit.

- Tâches SQL Spark : utilisées pour exécuter un fichier de requête SQL via Spark SQL. Vous pouvez utiliser ce type de tâche pour exécuter des tâches Spark SQL. Ce type de tâche a les paramètres suivants :

  - Entrypoint : c'est la référence HCFS (système de fichiers compatible Hadoop) au fichier de requête SQL que vous souhaitez exécuter.

    Pour obtenir la liste des paramètres Spark supplémentaires que vous pouvez utiliser pour une tâche Spark SQL, consultez Exécution de scripts Spark SQL via l' StartJobRun API.

- `--configuration-overrides` : vous pouvez remplacer les configurations par défaut des applications en fournissant un objet de configuration. Vous pouvez utiliser une syntaxe abrégée pour fournir la configuration, ou vous pouvez faire référence à l'objet de configuration dans un fichier JSON. Les objets de configuration sont composés d'une classification, de propriétés et de configurations imbriquées en option. Les propriétés sont les paramètres que vous souhaitez remplacer dans ce fichier. Vous pouvez spécifier plusieurs classifications pour plusieurs applications d'un seul objet JSON. Les classifications de configuration qui sont disponibles varient d'une version d'Amazon EMR à l'autre. Pour obtenir une liste des classifications de configurations disponibles pour chaque version d'Amazon EMR, consultez Versions Amazon EMR on EKS.

  Si vous transmettez la même configuration dans un remplacement d'application et dans les paramètres de soumission Spark, les paramètres de soumission Spark auront la priorité. La liste complète des priorités de configuration suit, de la priorité la plus élevée à la plus faible.

  - Configuration fournie lors de la création de `SparkSession`.
  - Configuration fournie dans le cadre de `sparkSubmitParameters` en utilisant `–conf`.
  - Configuration fournie dans le cadre des remplacements d'applications.
  - Configurations optimisées choisies par Amazon EMR pour la version.
  - Configurations open-source par défaut pour l'application.

  Pour surveiller les exécutions de tâches à l'aide d'Amazon CloudWatch ou d'Amazon S3, vous devez fournir les détails de configuration pour CloudWatch. Pour plus d’informations, consultez Configuration d'une exécution de tâche pour utiliser les journaux Amazon S3 et Configurer une tâche exécutée pour utiliser Amazon CloudWatch Logs. Si le compartiment ou le groupe de

CloudWatch journaux S3 n'existe pas, Amazon EMR le crée avant de télécharger les journaux dans le compartiment.

- Pour une liste supplémentaire d'options de configuration Kubernetes, consultez la rubrique Propriétés Spark sur Kubernetes.

  Les configurations Spark suivantes ne sont pas prises en charge.

  - `spark.kubernetes.authenticate.driver.serviceAccountName`
  - `spark.kubernetes.authenticate.executor.serviceAccountName`
  - `spark.kubernetes.namespace`
  - `spark.kubernetes.driver.pod.name`
  - `spark.kubernetes.container.image.pullPolicy`
  - `spark.kubernetes.container.image`

  > **Note**
  >
  > Vous pouvez utiliser `spark.kubernetes.container.image` pour personnaliser les images Docker. Pour de plus amples informations, veuillez consulter Personnalisation d'images Docker pour Amazon EMR on EKS.

Configuration d'une exécution de tâche pour utiliser les journaux Amazon S3

Pour être en mesure de suivre l'avancement des tâches et de résoudre les problèmes d'échec, vous devez configurer vos tâches de manière à envoyer des informations de journal à Amazon S3, Amazon CloudWatch Logs ou aux deux. Cette rubrique vous aide à commencer à publier des journaux d'application sur Amazon S3 pour vos tâches lancées à l'aide d'Amazon EMR on EKS.

Politique IAM des journaux S3

Pour que vos tâches puissent envoyer des données de journal à Amazon S3, les autorisations suivantes doivent être incluses dans la politique d'autorisations du rôle d'exécution des tâches. *amzn-s3-demo-logging-bucket*Remplacez-le par le nom de votre bucket de journalisation.

JSON

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

```
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket",
        "arn:aws:s3:::amzn-s3-demo-bucket/*"
      ],
      "Sid": "AllowS3Putobject"
    }
  ]
}
```

> **Note**
>
> Amazon EMR on EKS peut également créer un compartiment Amazon S3. Si aucun compartiment Amazon S3 n'est disponible, incluez l'autorisation `"s3:CreateBucket"` dans la politique IAM.

Une fois que vous avez donné à votre rôle d'exécution les autorisations appropriées pour envoyer des journaux à Amazon S3, les données de vos journaux sont envoyées aux emplacements Amazon S3 suivants lorsque la `s3MonitoringConfiguration` est transmise dans la section `monitoringConfiguration` d'une demande `start-job-run`, comme indiqué dans Gérer les exécutions de tâches à l'aide du AWS CLI.

- Journaux des soumissionnaires -//*virtual-cluster-id*/jobs/ /containers/*logUri*/(*job-id*stderr.gz/stdout.gz) *pod-name*
- Journaux de pilotes -//*virtual-cluster-id*/jobs/ *logUri job-id* /containers/ /spark- -driver/ (stderr.gz/stdout.gz*spark-application-id*) *job-id*
- Journaux de l'exécuteur -//*virtual-cluster-id*/jobs/ *logUri job-id* /containers///(stderr.gz/ stdout.gz*spark-application-id*) *executor-pod-name*

Configurer une tâche exécutée pour utiliser Amazon CloudWatch Logs

Pour suivre l'avancement des tâches et résoudre les problèmes d'échec, vous devez configurer vos tâches de manière à envoyer des informations de journal à Amazon S3, Amazon CloudWatch Logs ou aux deux. Cette rubrique vous aide à commencer à utiliser CloudWatch les journaux sur vos tâches lancées avec Amazon EMR sur EKS. Pour plus d'informations sur CloudWatch les journaux, consultez la section Surveillance des fichiers journaux dans le guide de CloudWatch l'utilisateur Amazon.

CloudWatch Politique IAM des journaux

Pour que vos tâches envoient des données de journal à CloudWatch Logs, les autorisations suivantes doivent être incluses dans la politique d'autorisation relative au rôle d'exécution des tâches. Remplacez *my_log_group_name* et *my_log_stream_prefix* par les noms de votre groupe de CloudWatch journaux et les noms de vos flux de journaux, respectivement. Amazon EMR on EKS crée le groupe de journaux et le flux de journaux s'ils n'existent pas, à condition que l'ARN du rôle d'exécution dispose des autorisations appropriées.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogStream",
        "logs:DescribeLogGroups",
        "logs:DescribeLogStreams"
      ],
      "Resource": [
        "arn:aws:logs:*:*:*"
      ],
      "Sid": "AllowLOGSCreatelogstream"
    },
    {
      "Effect": "Allow",
      "Action": [
        "logs:PutLogEvents"
      ],
      "Resource": [
```

```
        "arn:aws:logs:*:*:log-group:my_log_group_name:log-
stream:my_log_stream_prefix/*"
      ],
      "Sid": "AllowLOGSPutlogevents"
    }
  ]
}
```

> **Note**
>
> Amazon EMR on EKS peut également créer un flux de journaux. S'il n'existe pas de flux de journaux, la politique IAM doit inclure l'autorisation `"logs:CreateLogGroup"`.

Une fois que vous avez accordé les autorisations appropriées à votre rôle d'exécution, votre application envoie ses données de journal à CloudWatch Logs lorsqu'elles `cloudWatchMonitoringConfiguration` sont transmises dans la `monitoringConfiguration` section d'une `start-job-run` demande, comme indiqué dansGérer les exécutions de tâches à l'aide du AWS CLI.

Dans l'`StartJobRun`API, *`log_group_name`* il s'agit du nom du groupe de journaux pour CloudWatch et *`log_stream_prefix`* du préfixe du nom du flux de journaux pour CloudWatch. Vous pouvez afficher et y faire des recherches dans la AWS Management Console.

- Journaux de l'expéditeur -*`logGroup`*//*`virtual-cluster-id`*/jobs/ *`logStreamPrefix`* / containers//(*`job-id`*stderr/stdout) *`pod-name`*
- Journaux de pilotes -*`logGroup`*//*`virtual-cluster-id`*/jobs/ *`logStreamPrefix`* *`job-id`* / containers/ /spark- -driver/ (stderrstdout*`spark-application-id`*) *`job-id`*
- Journaux de l'exécuteur -*`logGroup`*//*`virtual-cluster-id`*/jobs/ *`logStreamPrefix`* *`job-id`* / containers///(*`spark-application-id`*stderr/stdout) *`executor-pod-name`*

Liste des exécutions de tâches

Vous pouvez exécuter `list-job-run` pour afficher l'état des exécutions de tâches, comme le montre l'exemple ci-dessous.

```
aws emr-containers list-job-runs --virtual-cluster-id <cluster-id>
```

Description d'une exécution de tâche

Vous pouvez exécuter `describe-job-run` pour obtenir plus de détails sur la tâche, tels que l'état de la tâche, les détails de l'état et le nom de la tâche, comme le montre l'exemple ci-dessous.

```
aws emr-containers describe-job-run --virtual-cluster-id cluster-id --id job-run-id
```

Annulation d'une exécution de tâche

Vous pouvez exécuter `cancel-job-run` pour annuler des tâches en cours d'exécution, comme le montre l'exemple ci-dessous.

```
aws emr-containers cancel-job-run --virtual-cluster-id cluster-id --id job-run-id
```

Exécution de scripts Spark SQL via l' StartJobRun API

Les versions 6.7.0 et ultérieures d'Amazon EMR on EKS comprennent un pilote de tâche Spark SQL qui vous permet d'exécuter des scripts Spark SQL via l'API `StartJobRun`. Vous pouvez fournir des fichiers de points d'entrée SQL pour exécuter directement des requêtes Spark SQL sur Amazon EMR on EKS à l'aide de l'API `StartJobRun`, sans aucune modification des scripts Spark SQL existants. Le tableau suivant répertorie les paramètres Spark pris en charge pour les tâches SQL Spark via l' StartJobRun API.

Vous pouvez choisir parmi les paramètres Spark suivants à envoyer à une tâche SQL Spark. Utilisez ces paramètres pour remplacer les propriétés Spark par défaut.

Option	Description
--name NOM	Nom de l'application
--jars fichiers JAR	Liste des fichiers JAR, séparés par des virgules, à inclure dans le chemin de classe du pilote et de l'exécuteur.
--packages	Liste des coordonnées Maven des fichiers JAR, séparées par des virgules, à inclure dans les chemins de classe du pilote et de l'exécuteur.

Option	Description
--exclude-packages	Liste des groupId:artifactId, séparés par des virgules, à exclure lors de la résolution des dépendances fournies dans –packages afin d'éviter les conflits de dépendances.
--repositories	Liste des référentiels distants supplémentaires, séparés par des virgules, à rechercher pour les coordonnées maven données fournies – packages.
--files FICHIERS	Liste des fichiers, séparés par des virgules, à placer dans le répertoire de travail de chaque exécuteur.
--conf PROPRIÉTÉ = VALEUR	Propriété de configuration Spark.
--properties-file FICHIER	Chemin d'accès au fichier à partir duquel charger des propriétés supplémentaires.
--driver-memory MEM	Mémoire pour le pilote. Par défaut, 1024 Mo.
--driver-java-options	Options Java supplémentaires à transmettre au pilote.
--driver-library-path	Entrées de chemin de bibliothèque supplémen taires à transmettre au pilote.
--driver-class-path	Entrées de chemin de classe supplémentaires à transmettre au pilote.
--executor-memory MEM	Mémoire par exécuteur. Valeur par défaut : 1 Go.
--driver-cores NUM	Nombre de cœurs utilisés par le pilote.
-- total-executor-cores NOMBRE	Nombre total de cœurs pour tous les exécuteur s.

Option	Description
--executor-cores NUM	Nombre de cœurs utilisés par chaque exécuteur.
--num-executors NUM	Nombre d'exécuteurs à lancer.
-hivevar <key=value>	Substitution de variables à appliquer aux commandes Hive, par exemple, `-hivevar A=B`
-hiveconf <property=value>	Valeur à utiliser pour la propriété donnée.

Pour une tâche SQL Spark, créez un fichier start-job-run-request .json et spécifiez les paramètres requis pour l'exécution de votre tâche, comme dans l'exemple suivant :

```
{
  "name": "myjob",
  "virtualClusterId": "123456",
  "executionRoleArn": "iam_role_name_for_job_execution",
  "releaseLabel": "emr-6.7.0-latest",
  "jobDriver": {
    "sparkSqlJobDriver": {
      "entryPoint": "entryPoint_location",
       "sparkSqlParameters": "--conf spark.executor.instances=2 --conf
 spark.executor.memory=2G --conf spark.executor.cores=2 --conf spark.driver.cores=1"
    }
  },
  "configurationOverrides": {
    "applicationConfiguration": [
      {
        "classification": "spark-defaults",
        "properties": {
          "spark.driver.memory":"2G"
         }
      }
    ],
    "monitoringConfiguration": {
      "persistentAppUI": "ENABLED",
      "cloudWatchMonitoringConfiguration": {
        "logGroupName": "my_log_group",
```

```
        "logStreamNamePrefix": "log_stream_prefix"
      },
      "s3MonitoringConfiguration": {
        "logUri": "s3://my_s3_log_location"
      }
    }
  }
}
```

États d'exécution de la tâche

Lorsque vous envoyez une tâche à une file d'attente de tâches Amazon EMR on EKS, l'exécution de la tâche passe à l'état `PENDING`. Elle passe ensuite par les états suivants jusqu'à ce qu'elle réussisse (se termine avec le code `0`) ou qu'elle échoue (se termine avec un code non nul).

Les exécutions de tâches peuvent avoir les états suivants :

- `PENDING` : état initial de la tâche lorsque l'exécution de tâche est soumise à Amazon EMR on EKS. La tâche attend d'être soumise au cluster virtuel, et Amazon EMR on EKS travaille à la soumission de cette tâche.
- `SUBMITTED` : une exécution de tâche qui a été soumise avec succès au cluster virtuel. Le planificateur de cluster essaie ensuite d'exécuter cette tâche sur le cluster.
- `RUNNING` : une tâche exécutée dans le cluster virtuel. Dans les applications Spark, cela signifie que le processus du pilote Spark est en état `running`.
- `FAILED` : une exécution de tâche qui n'a pas pu être soumise au cluster virtuel ou qui s'est terminée sans succès. Regardez StateDetails et trouvez FailureReason des informations supplémentaires sur cet échec de travail.
- `COMPLETED` : une exécution de travail qui s'est terminée avec succès.
- `CANCEL_PENDING` : l'annulation d'une exécution de tâche a été demandée. Amazon EMR on EKS essaie d'annuler la tâche sur le cluster virtuel.
- `CANCELLED` : une exécution de tâche qui a été annulée avec succès.

Affichage des tâches dans la console Amazon EMR

Les données d'exécution des tâches peuvent être consultées, ce qui vous permet de surveiller chaque tâche au fur et à mesure qu'elle passe par les états. Pour afficher les tâches dans la console Amazon EMR, procédez comme suit.

1. Dans le menu de gauche de la console Amazon EMR, sous Amazon EMR on EKS, choisissez Clusters virtuels.
2. Dans la liste des clusters virtuels, sélectionnez le cluster virtuel dont vous souhaitez consulter les tâches.
3. Dans le tableau Exécutions de tâches, sélectionnez Afficher les journaux pour afficher les détails d'une exécution de tâche.

> **Note**
>
> La prise en charge de l'expérience en un clic est activée par défaut. Elle peut être désactivée en réglant `persistentAppUI` sur `DISABLED` dans `monitoringConfiguration` lors de la soumission des tâches. Pour de plus amples informations, veuillez consulter Afficher les interfaces utilisateur d'application persistante.

Erreurs courantes lors de l'exécution de tâches

Les erreurs suivantes peuvent se produire lorsque vous exécutez l'API `StartJobRun`. Le tableau répertorie chaque erreur et fournit des mesures d'atténuation afin que vous puissiez résoudre les problèmes rapidement.

Message d'erreur	Condition d'erreur	Étapes suivantes recommandées
erreur : argument --*argument* est obligatoire	Les paramètres obligatoires sont manquants.	Ajoutez les arguments manquants à la demande de l'API.
Une erreur s'est produite (AccessDeniedException) lors de l'appel de l' StartJobRunopération : L'utilisateur : n'*ARN*est pas autorisé à effectuer : emr-containers : StartJobRun	Le rôle d'exécution est manquant.	Consultez la rubrique Utilisation on de Utilisation des rôles d'exécution de tâches avec Amazon EMR on EKS.

Message d’erreur	Condition d'erreur	Étapes suivantes recommandées
Une erreur s'est produite (AccessDeniedException) lors de l'appel de l' StartJobRunopération : L'utilisateur : n'*ARN*est pas autorisé à effectuer : emr-containers : StartJobRun	L'appelant n'est pas autorisé à accéder au rôle d'exécution [format valide/non valide] via les clés de condition.	Consultez Utilisation des rôles d'exécution de tâches avec Amazon EMR on EKS.
Une erreur s'est produite (AccessDeniedException) lors de l'appel de l' StartJobRunopération : L'utilisateur : n'*ARN*est pas autorisé à effectuer : emr-containers : StartJobRun	L'ARN du soumissionnaire de la tâche et celui du rôle d'exécution proviennent de comptes différents.	Assurez-vous que l'ARN du soumissionnaire de la tâche et celui du rôle d'exécution proviennent du même compte AWS .
1 erreur de validation détectée : la valeur *Role* à « executionRoleArn » ne répondait pas au modèle d'expression régulière ARN : ^arn :(aws [a-Za-Z0-9-] *) :iam : :(\ d {12}) ? : (rôle ((\ u002F) | (\ u002F [\ u0021- \ u007F] + \ u002F)) [\ w +=, .@-] +)	L'appelant dispose d'autorisations pour le rôle d'exécution via des clés de condition, mais le rôle ne satisfait pas aux contraintes du format ARN.	Indiquez le rôle d'exécution en respectant le format ARN. Consultez Utilisation des rôles d'exécution de tâches avec Amazon EMR on EKS.
Une erreur s'est produite (ResourceNotFoundException) lors de l'appel de l' StartJobRunopération : Le cluster virtuel *Virtual Cluster ID* n'existe pas.	L'identifiant du cluster virtuel est introuvable.	Indiquez un identifiant de cluster virtuel enregistré dans Amazon EMR on EKS.

Message d’erreur	Condition d'erreur	Étapes suivantes recommandées
Une erreur s'est produite (ValidationException) lors de l'appel de l' StartJobRunopération : l'état du cluster virtuel n'*state*est pas valide pour créer une ressource JobRun.	Le cluster virtuel n'est pas prêt à exécuter la tâche.	Consultez États du cluster virtuel.
Une erreur s'est produite (ResourceNotFoundException) lors de l'appel de l' StartJobRunopération : Release *RELEASE* doesn't exist.	La version spécifiée lors de la soumission de la tâche est incorrecte.	Consultez Versions Amazon EMR on EKS.
Une erreur s'est produite (AccessDeniedException) lors de l'appel de l' StartJobRunopération : L'utilisateur : n'*ARN*est pas autorisé à effectuer : emr-containers : StartJobRun on resource : *ARN* avec un refus explicite. Une erreur s'est produite (AccessDeniedException) lors de l'appel de l' StartJobRunopération : L'utilisateur : n'*ARN*est pas autorisé à effectuer : emr-containers : StartJobRun on resource : *ARN*	L'utilisateur n'est pas autorisé à appeler StartJobRun.	Consultez Utilisation des rôles d'exécution de tâches avec Amazon EMR on EKS.

Message d’erreur	Condition d'erreur	Étapes suivantes recommandées
Une erreur s'est produite (ValidationException) lors de l'appel de l' StartJobRunopération : ConfigurationOverrides.MonitoringConfiguration.s3 MonitoringConfiguration .LogURI n'a pas réussi à satisfaire la contrainte : %s	La syntaxe de l'URI du chemin S3 n'est pas valide.	logUri doit être au format s3 ://...

Les erreurs suivantes peuvent se produire lorsque vous exécutez l'API `DescribeJobRun` avant les exécutions de tâches.

Message d’erreur	Condition d'erreur	Étapes suivantes recommandées
StateDetails : échec JobRun de la soumission. La classification *classification* n'est pas prise en charge. failureReason : VALIDATION_ERROR état : ÉCHEC.	Les paramètres saisis ne StartJobRun sont pas valides.	Consultez Versions Amazon EMR on EKS.
StateDetails : *EKS Cluster ID* Le cluster n'existe pas. failureReason : CLUSTER_UNAVAILABLE	Le cluster EKS n'est pas disponible.	Vérifiez si le cluster EKS existe et dispose des autorisations appropriées. Pour de plus amples informations, veuillez consulter Configuration d'Amazon EMR on EKS.

Message d’erreur	Condition d'erreur	Étapes suivantes recommandées
état : ÉCHEC		
StateDetails : *EKS Cluster ID* Le cluster ne dispose pas d'autorisations suffisantes. failureReason : CLUSTER_UNAVAILABLE état : ÉCHEC	Amazon EMR n'est pas autorisé à accéder au cluster EKS.	Vérifiez que les autorisations sont configurées pour Amazon EMR sur l'espace de noms enregistré. Pour de plus amples informations, veuillez consulter Configuration d'Amazon EMR on EKS.
StateDetails : *EKS Cluster ID* Le cluster n'est actuellement pas accessible. failureReason : CLUSTER_UNAVAILABLE état : ÉCHEC	Le cluster EKS n'est pas accessible.	Vérifiez que le cluster EKS existe et qu'il dispose des autorisations appropriées. Pour de plus amples informations, veuillez consulter Configuration d'Amazon EMR on EKS.
StateDetails : JobRun la soumission a échoué en raison d'une erreur interne. failureReason : INTERNAL_ERROR état : ÉCHEC	Une erreur interne s'est produite avec le cluster EKS.	N/A
StateDetails : *EKS Cluster ID* Le cluster ne dispose pas de ressources suffisantes. failureReason : USER_ERROR état : ÉCHEC	Les ressources du cluster EKS sont insuffisantes pour exécuter la tâche.	Ajoutez de la capacité au groupe de nœuds EKS ou configurez EKS Autoscaler. Pour plus d'informations, consultez la rubrique Cluster Autoscaler.

Les erreurs suivantes peuvent se produire lorsque vous exécutez l'API `DescribeJobRun` après les exécutions de tâches.

Message d'erreur	Condition d'erreur	Étapes suivantes recommandées
StateDetails : Problème lors de la surveillance de votre. JobRun *EKS Cluster ID*Le cluster n'existe pas. failureReason : CLUSTER_UNAVAILABLE état : ÉCHEC	Le cluster EKS n'existe pas.	Vérifiez que le cluster EKS existe et qu'il dispose des autorisations appropriées. Pour de plus amples informations, veuillez consulter Configuration d'Amazon EMR on EKS.
StateDetails : Problème lors de la surveillance de votre. JobRun *EKS Cluster ID*Le cluster ne dispose pas d'autorisations suffisantes. failureReason : CLUSTER_UNAVAILABLE état : ÉCHEC	Amazon EMR n'est pas autorisé à accéder au cluster EKS.	Vérifiez que les autorisations sont configurées pour Amazon EMR sur l'espace de noms enregistré. Pour de plus amples informations, veuillez consulter Configuration d'Amazon EMR on EKS.
StateDetails : Problème lors de la surveillance de votre. JobRun *EKS Cluster ID*Le cluster n'est actuellement pas joignable.	Le cluster EKS n'est pas accessible.	Vérifiez que le cluster EKS existe et qu'il dispose des autorisations appropriées. Pour de plus amples informations, veuillez consulter Configuration d'Amazon EMR on EKS.

Message d’erreur	Condition d'erreur	Étapes suivantes recommandées
failureReason : CLUSTER_UNAVAILABLE état : ÉCHEC		
StateDetails : Problème lors de la surveillance de votre JobRun compte en raison d'une erreur interne failureReason : INTERNAL_ERROR état : ÉCHEC	Une erreur interne s'est produite et empêche la JobRun surveillance.	N/A

L'erreur suivante peut se produire lorsqu'une tâche ne peut pas démarrer et qu'elle reste en attente en état SOUMIS pendant 15 minutes. Cela peut être dû à un manque de ressources du cluster.

Message d’erreur	Condition d'erreur	Étapes suivantes recommandées
délai d'expiration du cluster	La tâche est à l'état SOUMIS depuis 15 minutes ou plus.	Vous pouvez remplacer la valeur par défaut de 15 minutes pour ce paramètre par la configuration ci-dessous.

Utilisez la configuration ci-dessous pour modifier le paramètre de délai d'expiration du cluster à 30 minutes. Notez que vous indiquez la nouvelle valeur `job-start-timeout` en secondes :

```
{
"configurationOverrides": {
  "applicationConfiguration": [{
      "classification": "emr-containers-defaults",
      "properties": {
          "job-start-timeout":"1800"
```

```
      }
  }]
}
```

Utilisation des modèles de tâche

Un modèle de tâche stocke des valeurs qui peuvent être partagées lors de l'invocation de l'API `StartJobRun` pour démarrer une exécution de tâche. Il prend en charge deux cas d'utilisation :

- Pour éviter les valeurs répétitives et récurrentes des demandes de l'API `StartJobRun`.
- Pour imposer une règle selon laquelle certaines valeurs doivent être fournies via des demandes de l'API `StartJobRun`.

Les modèles de tâches vous permettent de définir un modèle réutilisable pour les exécutions de tâches afin d'appliquer une personnalisation supplémentaire, par exemple :

- Configuration de la capacité de calcul de l'exécuteur et du pilote
- Définition des propriétés de sécurité et de gouvernance telles que les rôles IAM
- Personnalisation d'une image Docker à utiliser dans plusieurs applications et pipelines de données

Les rubriques suivantes fournissent des informations détaillées sur l'utilisation des modèles, notamment sur leur utilisation pour démarrer l'exécution d'une tâche et sur la modification des paramètres des modèles.

Rubriques

Création et utilisation d'un modèle de tâche pour démarrer une exécution de tâche

Cette section décrit la création d'un modèle de tâche et son utilisation pour démarrer une tâche exécutée avec le AWS Command Line Interface (AWS CLI).

Création d'un modèle de tâche

1. Créez un fichier `create-job-template-request.json` et indiquez les paramètres requis pour votre modèle de tâche, comme le montre l'exemple de fichier JSON ci-dessous. Pour plus d'informations sur tous les paramètres disponibles, consultez l'CreateJobTemplateAPI.

   La plupart des valeurs requises pour l'API `StartJobRun` le sont également pour `jobTemplateData`. Si vous souhaitez utiliser des espaces réservés pour n'importe quel paramètre et fournir des valeurs lors de l'appel à StartJobRun l'aide d'un modèle de tâche, consultez la section suivante sur les paramètres du modèle de tâche.

```
{
   "name": "mytemplate",
   "jobTemplateData": {
        "executionRoleArn": "iam_role_arn_for_job_execution",
        "releaseLabel": "emr-6.7.0-latest",
        "jobDriver": {
            "sparkSubmitJobDriver": {
                "entryPoint": "entryPoint_location",
                "entryPointArguments": [ "argument1","argument2",...],
                "sparkSubmitParameters": "--class <main_class> --conf
 spark.executor.instances=2 --conf spark.executor.memory=2G --conf
 spark.executor.cores=2 --conf spark.driver.cores=1"
            }
       },
       "configurationOverrides": {
            "applicationConfiguration": [
                {
                    "classification": "spark-defaults",
                    "properties": {
                         "spark.driver.memory":"2G"
                    }
                }
            ],
            "monitoringConfiguration": {
                "persistentAppUI": "ENABLED",
                "cloudWatchMonitoringConfiguration": {
                    "logGroupName": "my_log_group",
                    "logStreamNamePrefix": "log_stream_prefix"
                },
                "s3MonitoringConfiguration": {
                    "logUri": "s3://my_s3_log_location/"
                }
            }
```

```
        }
    }
}
```

2. Utilisez la commande `create-job-template` avec un chemin d'accès au fichier `create-job-template-request.json` stocké localement.

```
aws emr-containers create-job-template \
--cli-input-json file://./create-job-template-request.json
```

Démarrage d'une exécution de tâche à l'aide d'un modèle de tâche

Saisissez l'identifiant du cluster virtuel, l'identifiant du modèle de tâche et le nom de la tâche dans la commande `StartJobRun`, comme indiqué dans l'exemple ci-dessous.

```
aws emr-containers start-job-run \
--virtual-cluster-id 123456 \
--name myjob \
--job-template-id 1234abcd
```

Définition des paramètres du modèle de tâche

Les paramètres du modèle de tâche vous permettent d'indiquer des variables dans le modèle de tâche. Les valeurs de ces variables de paramètres devront être indiquées lors du démarrage d'une exécution de tâche à l'aide de ce modèle de tâche. Les paramètres du modèle de tâche sont indiqués au format `${parameterName}`. Vous pouvez choisir d'indiquer n'importe quelle valeur dans un champ `jobTemplateData` comme paramètre de modèle de tâche. Pour chacune des variables de paramètre du modèle de tâche, indiquez son type de données (`STRING` ou `NUMBER`) et éventuellement une valeur par défaut. L'exemple ci-dessous montre comment vous pouvez indiquer les paramètres du modèle de tâche pour l'emplacement du point d'entrée, la classe principale et les valeurs de l'emplacement du journal S3.

Indication de l'emplacement du point d'entrée, de la classe principale et de l'emplacement du journal Amazon S3 en tant que paramètres du modèle de tâche

1. Créez un fichier `create-job-template-request.json` et indiquez les paramètres requis pour votre modèle de tâche, comme le montre l'exemple de fichier JSON ci-dessous. Pour plus d'informations sur les paramètres, consultez l'CreateJobTemplateAPI.

```
{
   "name": "mytemplate",
   "jobTemplateData": {
        "executionRoleArn": "iam_role_arn_for_job_execution",
        "releaseLabel": "emr-6.7.0-latest",
        "jobDriver": {
            "sparkSubmitJobDriver": {
                "entryPoint": "${EntryPointLocation}",
                "entryPointArguments": [ "argument1","argument2",...],
                "sparkSubmitParameters": "--class ${MainClass} --conf
 spark.executor.instances=2 --conf spark.executor.memory=2G --conf
 spark.executor.cores=2 --conf spark.driver.cores=1"
            }
       },
       "configurationOverrides": {
            "applicationConfiguration": [
                {
                    "classification": "spark-defaults",
                    "properties": {
                         "spark.driver.memory":"2G"
                    }
                }
            ],
            "monitoringConfiguration": {
                "persistentAppUI": "ENABLED",
                "cloudWatchMonitoringConfiguration": {
                    "logGroupName": "my_log_group",
                    "logStreamNamePrefix": "log_stream_prefix"
                },
                "s3MonitoringConfiguration": {
                    "logUri": "${LogS3BucketUri}"
                }
            }
       },
       "parameterConfiguration": {
            "EntryPointLocation": {
                "type": "STRING"
            },
            "MainClass": {
                "type": "STRING",
                "defaultValue":"Main"
            },
            "LogS3BucketUri": {
```

```
                "type": "STRING",
                "defaultValue":"s3://my_s3_log_location/"
            }
        }
    }
}
```

2. Utilisez la commande `create-job-template` avec un chemin d'accès au fichier `create-job-template-request.json` stocké localement ou dans Amazon S3.

```
aws emr-containers create-job-template \
--cli-input-json file://./create-job-template-request.json
```

Démarrage d'une exécution de tâche à l'aide d'un modèle de tâche et des paramètres du modèle de tâche

Pour démarrer une tâche avec un modèle de tâche contenant les paramètres du modèle de tâche, indiquez l'identifiant du modèle de tâche ainsi que les valeurs des paramètres du modèle de tâche dans la demande de l'API `StartJobRun`, comme indiqué ci-dessous.

```
aws emr-containers start-job-run \
--virtual-cluster-id 123456 \
--name myjob \
--job-template-id 1234abcd \
--job-template-parameters '{"EntryPointLocation": "entry_point_location","MainClass":
 "ExampleMainClass","LogS3BucketUri": "s3://example_s3_bucket/"}'
```

Contrôle de l'accès aux modèles de tâches

La politique `StartJobRun` vous permet de vous assurer qu'un utilisateur ou un rôle ne peut exécuter des tâches qu'à l'aide de modèles de tâches que vous indiquez et ne peut pas exécuter d'opérations `StartJobRun` sans utiliser les modèles de tâches indiqués. Pour ce faire, assurez-vous d'abord d'accorder à l'utilisateur ou au rôle une autorisation de lecture pour les modèles de tâches spécifiés, comme indiqué ci-dessous.

JSON

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

```
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "emr-containers:DescribeJobRun"
      ],
      "Resource": [
        "arn:aws:emr-containers:*:*:jobtemplate/job_template_1_id",
        "arn:aws:emr-containers:*:*:jobtemplate/job_template_2_id"
      ],
      "Sid": "AllowEMRCONTAINERSDescribejobtemplate"
    }
  ]
}
```

Pour garantir qu'un utilisateur ou un rôle ne peut invoquer une opération `StartJobRun` que s'il utilise des modèles de tâches spécifiques, vous pouvez accorder l'autorisation de politique `StartJobRun` suivante à un utilisateur ou à un rôle donné.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "emr-containers:StartJobRun"
      ],
      "Resource": [
        "arn:aws:emr-containers:*:*:/virtualclusters/virtual_cluster_id"
      ],
      "Condition": {
        "ArnLike": {
          "emr-containers:JobTemplateArn": [
            "arn:aws:emr-containers:*:*:jobtemplate/job_template_1_id",
            "arn:aws:emr-containers:*:*:jobtemplate/job_template_2_id"
          ]
        }
      },
      "Sid": "AllowEMRCONTAINERSStartjobrun"
```

```
    }
  ]
}
```

Si le modèle de tâche spécifie un paramètre de modèle de tâche dans le champ ARN du rôle d'exécution, l'utilisateur pourra indiquer une valeur pour ce paramètre et pourra ainsi invoquer `StartJobRun` à l'aide d'un rôle d'exécution arbitraire. Pour limiter les rôles d'exécution que l'utilisateur peut indiquer, consultez la rubrique Contrôle de l'accès au rôle d'exécution dans Utilisation des rôles d'exécution de tâches avec Amazon EMR on EKS.

Si aucune condition n'est indiquée dans la politique d'action `StartJobRun` ci-dessus pour un utilisateur ou un rôle donné, l'utilisateur ou le rôle sera autorisé à invoquer une action `StartJobRun` sur le cluster virtuel indiqué en utilisant un modèle de tâche arbitraire auquel il a accès en lecture ou en utilisant un rôle d'exécution arbitraire.

Utilisation de modèles de pods

À partir des versions 5.33.0 ou 6.3.0 d'Amazon EMR, Amazon EMR on EKS prend en charge la fonctionnalité de modèle de pod de Spark. Le pod est un groupe d'un ou plusieurs conteneurs, avec des ressources de stockage et de réseau partagées, et une spécification sur la manière d'exécuter les conteneurs. Les modèles de pods sont des spécifications qui déterminent comment exécuter chaque pod. Vous pouvez utiliser des fichiers de modèles de pods pour définir les configurations des pods de pilote ou d'exécuteur que les configurations Spark ne prennent pas en charge. Pour plus d'informations sur la fonctionnalité de modèle de pod de Spark, consultez la rubrique Modèle de pod.

> **Note**
>
> La fonctionnalité de modèle de pod ne fonctionne qu'avec les pods de pilote et d'exécuteur. Vous ne pouvez pas configurer les modules d'envoi de tâches à l'aide du modèle de module.

Scénarios courants

Vous pouvez définir comment exécuter des tâches Spark sur des clusters EKS partagés en utilisant des modèles de pods sur Amazon EMR on EKS et économiser des coûts tout en améliorant l'utilisation des ressources et les performances.

- Pour réduire les coûts, vous pouvez planifier l'exécution des tâches du pilote Spark sur des instances Amazon EC2 On-Demand tout en planifiant l'exécution des tâches de l'exécuteur Spark sur des instances Amazon EC2 Spot.
- Pour augmenter l'utilisation des ressources, vous pouvez prendre en charge plusieurs équipes exécutant leurs charges de travail sur le même cluster EKS. Chaque équipe aura un groupe de EC2 nœuds Amazon désigné sur lequel exécuter ses charges de travail. Vous pouvez utiliser des modèles de pods pour appliquer une tolérance correspondante à leur charge de travail.
- Pour améliorer la surveillance, vous pouvez utiliser un conteneur de journalisation distinct pour transmettre les journaux à votre application de surveillance existante.

Par exemple, le fichier de modèle de pod suivant illustre un scénario d'utilisation courant.

```
apiVersion: v1
kind: Pod
spec:
  volumes:
    - name: source-data-volume
      emptyDir: {}
    - name: metrics-files-volume
      emptyDir: {}
  nodeSelector:
    eks.amazonaws.com/nodegroup: emr-containers-nodegroup
  containers:
  - name: spark-kubernetes-driver # This will be interpreted as driver Spark main
 container
    env:
      - name: RANDOM
        value: "random"
    volumeMounts:
      - name: shared-volume
        mountPath: /var/data
      - name: metrics-files-volume
        mountPath: /var/metrics/data
  - name: custom-side-car-container # Sidecar container
    image: <side_car_container_image>
    env:
      - name: RANDOM_SIDECAR
        value: random
    volumeMounts:
      - name: metrics-files-volume
        mountPath: /var/metrics/data
```

```
    command:
      - /bin/sh
      - '-c'
      -  <command-to-upload-metrics-files>
  initContainers:
  - name: spark-init-container-driver # Init container
    image: <spark-pre-step-image>
    volumeMounts:
      - name: source-data-volume # Use EMR predefined volumes
        mountPath: /var/data
    command:
      - /bin/sh
      - '-c'
      -  <command-to-download-dependency-jars>
```

Le modèle de pod exécute les tâches suivantes :

- Ajoutez un nouveau conteneur d'initialisation qui est exécuté avant le démarrage du conteneur principal Spark. Le conteneur d'initialisation partage le EmptyDirvolume appelé `source-data-volume` avec le conteneur principal Spark. Vous pouvez demander à votre conteneur d'initialisation d'exécuter des étapes d'initialisation, telles que le téléchargement de dépendances ou la génération de données d'entrée. Le conteneur principal Spark consomme ensuite les données.
- Ajoutez un autre conteneur sidecar exécuté en même temps que le conteneur principal Spark. Les deux conteneurs partagent un autre volume `EmptyDir` appelé `metrics-files-volume`. Votre tâche Spark peut générer des métriques, telles que les métriques Prometheus. La tâche Spark peut ensuite placer les métriques dans un fichier et demander au conteneur sidecar de charger les fichiers dans votre propre système de BI en vue d'une analyse ultérieure.
- Ajoutez une nouvelle variable d'environnement au conteneur principal Spark. Vous pouvez demander à votre tâche de consommer la variable d'environnement.
- Définissez un sélecteur de nœuds afin que le pod soit uniquement planifié sur le groupe de nœuds `emr-containers-nodegroup`. Cela permet d'isoler les ressources informatiques entre les tâches et les équipes.

Activation des modèles de pods avec Amazon EMR on EKS

Pour activer la fonctionnalité de modèle de pod avec Amazon EMR on EKS, configurez les propriétés `spark.kubernetes.driver.podTemplateFile` et `spark.kubernetes.executor.podTemplateFile` de Spark pour qu'elles renvoient aux fichiers de modèle de pod sur Amazon S3. Spark télécharge ensuite le fichier de modèle de pod et l'utilise pour construire des pods de pilote et d'exécuteur.

> **Note**
>
> Spark utilise le rôle d'exécution de tâches pour charger le modèle de pod. Le rôle d'exécution de tâches doit donc être autorisé à accéder à Amazon S3 afin de charger les modèles de pod. Pour de plus amples informations, veuillez consulter Création d'un rôle d'exécution des tâches.

Vous pouvez utiliser les `SparkSubmitParameters` pour indiquer le chemin Amazon S3 vers le modèle de pod, comme le montre le fichier JSON d'exécution de tâches suivant.

```
{
  "name": "myjob",
  "virtualClusterId": "123456",
  "executionRoleArn": "iam_role_name_for_job_execution",
  "releaseLabel": "release_label",
  "jobDriver": {
    "sparkSubmitJobDriver": {
      "entryPoint": "entryPoint_location",
      "entryPointArguments": ["argument1", "argument2", ...],
       "sparkSubmitParameters": "--class <main_class> \
        --conf
 spark.kubernetes.driver.podTemplateFile=s3://path_to_driver_pod_template \
        --conf
 spark.kubernetes.executor.podTemplateFile=s3://path_to_executor_pod_template \
        --conf spark.executor.instances=2 \
        --conf spark.executor.memory=2G \
        --conf spark.executor.cores=2 \
        --conf spark.driver.cores=1"
    }
  }
}
```

Vous pouvez également utiliser les `configurationOverrides` pour indiquer le chemin Amazon S3 vers le modèle de pod, comme le montre le fichier JSON d'exécution de tâches suivant.

```
{
  "name": "myjob",
  "virtualClusterId": "123456",
  "executionRoleArn": "iam_role_name_for_job_execution",
  "releaseLabel": "release_label",
  "jobDriver": {
    "sparkSubmitJobDriver": {
      "entryPoint": "entryPoint_location",
      "entryPointArguments": ["argument1", "argument2", ...],
       "sparkSubmitParameters": "--class <main_class> \
         --conf spark.executor.instances=2 \
         --conf spark.executor.memory=2G \
         --conf spark.executor.cores=2 \
         --conf spark.driver.cores=1"
    }
  },
  "configurationOverrides": {
    "applicationConfiguration": [
      {
        "classification": "spark-defaults",
        "properties": {
          "spark.driver.memory":"2G",
          "spark.kubernetes.driver.podTemplateFile":"s3://path_to_driver_pod_template",

 "spark.kubernetes.executor.podTemplateFile":"s3://path_to_executor_pod_template"
         }
      }
    ]
  }
}
```

> **Note**
>
> 1. Vous devez suivre les consignes de sécurité lorsque vous utilisez la fonctionnalité de modèle de pod avec Amazon EMR on EKS, telles que l'isolation du code d'application non fiable. Pour de plus amples informations, veuillez consulter Bonnes pratiques de sécurité pour Amazon EMR on EKS.

2. Vous ne pouvez pas modifier les noms des conteneurs principaux Spark en utilisant `spark.kubernetes.driver.podTemplateContainerName` et `spark.kubernetes.executor.podTemplateContainerName`, car ces noms sont codés en dur comme `spark-kubernetes-driver` et `spark-kubernetes-executors`. Si vous souhaitez personnaliser le conteneur principal Spark, vous devez indiquer le conteneur dans un modèle de pod avec ces noms codés en dur.

Champs du modèle de pod

Tenez compte des restrictions de champ suivantes lors de la configuration d'un modèle de pod avec Amazon EMR on EKS.

- Amazon EMR on EKS n'autorise que les champs suivants dans un modèle de pod pour permettre une planification correcte des tâches.

  Voici les champs autorisés au niveau du pod :

  - `apiVersion`
  - `kind`
  - `metadata`
  - `spec.activeDeadlineSeconds`
  - `spec.affinity`
  - `spec.containers`
  - `spec.enableServiceLinks`
  - `spec.ephemeralContainers`
  - `spec.hostAliases`
  - `spec.hostname`
  - `spec.imagePullSecrets`
  - `spec.initContainers`
  - `spec.nodeName`
  - `spec.nodeSelector`
  - `spec.overhead`
  - `spec.preemptionPolicy`
  - `spec.priority`

- spec.priorityClassName
- spec.readinessGates
- spec.runtimeClassName
- spec.schedulerName
- spec.subdomain
- spec.terminationGracePeriodSeconds
- spec.tolerations
- spec.topologySpreadConstraints
- spec.volumes

Voici les champs autorisés au niveau du conteneur principal Spark :

- env
- envFrom
- name
- lifecycle
- livenessProbe
- readinessProbe
- resources
- startupProbe
- stdin
- stdinOnce
- terminationMessagePath
- terminationMessagePolicy
- tty
- volumeDevices
- volumeMounts
- workingDir

Lorsque vous utilisez des champs non autorisés dans le modèle de pod, Spark génère une exception et la tâche échoue. L'exemple suivant montre un message d'erreur dans le journal du contrôleur Spark en raison de champs non autorisés.

```
Executor pod template validation failed.
Field container.command in Spark main container not allowed but specified.
```

- Amazon EMR on EKS prédéfinit les paramètres suivants dans un modèle de pod. Les champs que vous indiquez dans un modèle de pod ne doivent pas se chevaucher avec ces champs.

  Voici les noms de volumes prédéfinis :

  - `emr-container-communicate`
  - `config-volume`
  - `emr-container-application-log-dir`
  - `emr-container-event-log-dir`
  - `temp-data-dir`
  - `mnt-dir`
  - `home-dir`
  - `emr-container-s3`

  Voici les montages de volume prédéfinis qui s'appliquent uniquement au conteneur principal Spark :

  - Nom : `emr-container-communicate` ; MountPath : `/var/log/fluentd`
  - Nom : `emr-container-application-log-dir` ; MountPath : `/var/log/spark/user`
  - Nom : `emr-container-event-log-dir` ; MountPath : `/var/log/spark/apps`
  - Nom : `mnt-dir` ; MountPath : `/mnt`
  - Nom : `temp-data-dir` ; MountPath : `/tmp`
  - Nom : `home-dir` ; MountPath : `/home/hadoop`

  Voici les variables d'environnement prédéfinies qui s'appliquent uniquement au conteneur principal Spark :

  - `SPARK_CONTAINER_ID`
  - `K8S_SPARK_LOG_URL_STDERR`
  - `K8S_SPARK_LOG_URL_STDOUT`
  - `SIDECAR_SIGNAL_FILE`

> **Note**
>
> Vous pouvez toujours utiliser ces volumes prédéfinis et les monter dans vos conteneurs sidecar supplémentaires. Par exemple, vous pouvez utiliser `emr-container-application-log-dir` et le monter sur votre propre conteneur sidecar défini dans le modèle de pod.

Si les champs que vous indiquez entrent en conflit avec l'un des champs prédéfinis du modèle de pod, Spark génère une exception et la tâche échoue. L'exemple suivant montre un message d'erreur dans le journal de l'application Spark en raison de conflits avec les champs prédéfinis.

```
Defined volume mount path on main container must not overlap with reserved mount
 paths: [<reserved-paths>]
```

Considérations relatives aux conteneurs sidecar

Amazon EMR contrôle le cycle de vie des pods provisionnés par Amazon EMR on EKS. Les conteneurs sidecar doivent suivre le même cycle de vie que le conteneur principal Spark. Si vous injectez des conteneurs sidecar supplémentaires dans vos pods, nous vous recommandons d'intégrer la gestion du cycle de vie des pods définie par Amazon EMR afin que le conteneur sidecar puisse s'arrêter de lui-même lorsque le conteneur principal Spark s'arrête.

Pour réduire les coûts, nous vous recommandons de mettre en œuvre un processus qui empêche les pods de pilotes avec des conteneurs sidecar de continuer à fonctionner après la fin de votre tâche. Le pilote Spark supprime les pods de l'exécuteur lorsque celui-ci a terminé sa tâche. Toutefois, lorsqu'un programme de pilote est terminé, les conteneurs sidecar supplémentaires continuent de fonctionner. Le pod est facturé jusqu'à ce qu'Amazon EMR on EKS nettoie le pod du pilote, généralement en moins d'une minute après la fin de l'exécution du conteneur principal Spark du pilote. Pour réduire les coûts, vous pouvez intégrer vos conteneurs sidecar supplémentaires au mécanisme de gestion du cycle de vie qu'Amazon EMR on EKS définit pour les pods de pilote et d'exécuteur, comme décrit dans la section suivante.

Le conteneur principal Spark dans les pods de pilote et d'exécuteur envoie `heartbeat` à un `/var/log/fluentd/main-container-terminated` de fichier toutes les deux secondes. En ajoutant le montage de volume `emr-container-communicate` prédéfini Amazon EMR à votre

conteneur sidecar, vous pouvez définir un sous-processus de votre conteneur sidecar pour suivre périodiquement l'heure de la dernière modification de ce fichier. Le sous-processus s'arrête alors de lui-même s'il découvre que le conteneur principal Spark arrête la `heartbeat` pendant une durée plus longue.

L'exemple suivant illustre un sous-processus qui suit le fichier de pulsation et s'arrête de lui-même. *`your_volume_mount`*Remplacez-le par le chemin où vous montez le volume prédéfini. Le script est intégré à l'image utilisée par le conteneur sidecar. Dans un fichier de modèle de pod, vous pouvez indiquer un conteneur de sidecar à l'aide des commandes `sub_process_script.sh` et `main_command` suivantes.

```
MOUNT_PATH="your_volume_mount"
FILE_TO_WATCH="$MOUNT_PATH/main-container-terminated"
INITIAL_HEARTBEAT_TIMEOUT_THRESHOLD=60
HEARTBEAT_TIMEOUT_THRESHOLD=15
SLEEP_DURATION=10

function terminate_main_process() {
  # Stop main process
}

# Waiting for the first heartbeat sent by Spark main container
echo "Waiting for file $FILE_TO_WATCH to appear..."
start_wait=$(date +%s)
while ! [[ -f "$FILE_TO_WATCH" ]]; do
    elapsed_wait=$(expr $(date +%s) - $start_wait)
    if [ "$elapsed_wait" -gt "$INITIAL_HEARTBEAT_TIMEOUT_THRESHOLD" ]; then
        echo "File $FILE_TO_WATCH not found after $INITIAL_HEARTBEAT_TIMEOUT_THRESHOLD
 seconds; aborting"
        terminate_main_process
        exit 1
    fi
    sleep $SLEEP_DURATION;
done;
echo "Found file $FILE_TO_WATCH; watching for heartbeats..."

while [[ -f "$FILE_TO_WATCH" ]]; do
    LAST_HEARTBEAT=$(stat -c %Y $FILE_TO_WATCH)
    ELAPSED_TIME_SINCE_AFTER_HEARTBEAT=$(expr $(date +%s) - $LAST_HEARTBEAT)
    if [ "$ELAPSED_TIME_SINCE_AFTER_HEARTBEAT" -gt "$HEARTBEAT_TIMEOUT_THRESHOLD" ];
 then
```

```
        echo "Last heartbeat to file $FILE_TO_WATCH was more than
 $HEARTBEAT_TIMEOUT_THRESHOLD seconds ago at $LAST_HEARTBEAT; terminating"
        terminate_main_process
        exit 0
    fi
    sleep $SLEEP_DURATION;
done;
echo "Outside of loop, main-container-terminated file no longer exists"

# The file will be deleted once the fluentd container is terminated

echo "The file $FILE_TO_WATCH doesn't exist any more;"
terminate_main_process
exit 0
```

Utilisation des politiques de relance des tâches

Dans Amazon EMR on EKS en version 6.9.0 et ultérieure, vous pouvez définir une politique de relance pour vos exécutions de tâches. Les politiques de relance entraînent le redémarrage automatique d'un pod de pilote de tâche en cas d'échec ou de suppression. Cela permet aux tâches de streaming Spark de longue durée d'être plus résistantes aux échecs.

Définition d'une politique de relance pour une tâche

Pour configurer une politique de nouvelle tentative, vous devez fournir un `RetryPolicyConfiguration` champ à l'aide de l'StartJobRunAPI. Voici un exemple de `retryPolicyConfiguration` :

```
aws emr-containers start-job-run \
--virtual-cluster-id cluster_id \
--name sample-job-name \
--execution-role-arn execution-role-arn \
--release-label emr-6.9.0-latest \
--job-driver '{
  "sparkSubmitJobDriver": {
    "entryPoint": "local:///usr/lib/spark/examples/src/main/python/pi.py",
    "entryPointArguments": [ "2" ],
    "sparkSubmitParameters": "--conf spark.executor.instances=2 --conf
 spark.executor.memory=2G --conf spark.executor.cores=2 --conf spark.driver.cores=1"
  }
}' \
```

```
--retry-policy-configuration '{
    "maxAttempts": 5
  }' \
--configuration-overrides '{
  "monitoringConfiguration": {
    "cloudWatchMonitoringConfiguration": {
      "logGroupName": "my_log_group_name",
      "logStreamNamePrefix": "my_log_stream_prefix"
    },
    "s3MonitoringConfiguration": {
       "logUri": "s3://amzn-s3-demo-logging-bucket"
    }
  }
}'
```

> **Note**
>
> `retryPolicyConfiguration`n'est disponible qu'à partir de la version AWS CLI 1.27.68. Pour effectuer la mise AWS CLI à jour vers la dernière version, voir Installation ou mise à jour de la dernière version du AWS CLI

Indiquez dans ce champ `maxAttempts` le nombre maximum de fois que vous souhaitez que le pod de pilote de tâches soit redémarré en cas d'échec ou de suppression. L'intervalle d'exécution entre deux tentatives de relance du pilote de tâche est un intervalle de relance exponentiel de (10 secondes, 20 secondes, 40 secondes, etc.) qui est plafonné à 6 minutes, comme décrit dans la documentation Kubernetes.

> **Note**
>
> Chaque exécution supplémentaire d'un pilote de tâche sera facturée comme une autre tâche et sera soumise à la tarification d'Amazon EMR on EKS.

Valeurs de configuration de la politique de relance

- Politique de relance par défaut d'une tâche : `StartJobRun` comprend une politique de relance définie par défaut sur 1 tentative maximum. Vous pouvez configurer la politique de relance comme vous le souhaitez.

> **Note**
>
> Si le paramètre `maxAttempts` de `retryPolicyConfiguration` est défini sur 1, cela signifie qu'aucune nouvelle tentative n'aura lieu pour relancer le pod du pilote en cas d'échec.

- Désactivation de la politique de nouvelles tentatives pour une tâche : pour désactiver une politique de nouvelles tentatives, définissez la valeur maximale de tentatives sur 1. retryPolicyConfiguration

```
"retryPolicyConfiguration": {
    "maxAttempts": 1
}
```

- Définition du paramètre maxAttempts d'une tâche dans la plage valide : l'appel `StartJobRun` échouera si la valeur `maxAttempts` est en dehors de la plage valide. La plage `maxAttempts` valide est comprise entre 1 et 2 147 483 647 (entier 32 bits), qui correspond à la plage prise en charge pour le paramètre de configuration `backOffLimit` de Kubernetes. Pour plus d'informations, consultez la rubrique Politique en cas d'échec du mécanisme de retrait exponentiel pour le redémarrage des pods dans la documentation de Kubernetes. Si la valeur `maxAttempts` n'est pas valide, le message d'erreur suivant est renvoyé :

```
{
 "message": "Retry policy configuration's parameter value of maxAttempts is invalid"
}
```

Récupération de l'état de la politique de relance d'une tâche

Vous pouvez consulter l'état des nouvelles tentatives pour une tâche avec le `ListJobRuns`et `DescribeJobRun` APIs. Lorsque vous demandez une tâche avec une configuration de politique de relance activée, les réponses `ListJobRun` et `DescribeJobRun` contiennent l'état de la politique de relance dans le champ `RetryPolicyExecution`. En outre, la réponse `DescribeJobRun` contiendra la `RetryPolicyConfiguration` saisie dans la demande `StartJobRun` pour la tâche.

Exemples de réponses

ListJobRuns response

```
{
```

```
  "jobRuns": [
    ...
    ...
    "retryPolicyExecution" : {
      "currentAttemptCount": 2
    }
    ...
    ...
  ]
}
```

DescribeJobRun response

```
{
  ...
  ...
  "retryPolicyConfiguration": {
    "maxAttempts": 5
   },
   "retryPolicyExecution" : {
    "currentAttemptCount": 2
  },
  ...
  ...
}
```

Ces champs ne seront pas visibles lorsque la politique de relance est désactivée dans la tâche, comme décrit ci-dessous dans Valeurs de configuration de la politique de relance.

Surveillance d'une tâche à l'aide d'une politique de relance

Lorsque vous activez une politique de nouvelle tentative, un CloudWatch événement est généré pour chaque pilote de tâche créé. Pour vous abonner à ces événements, configurez une règle d' CloudWatch événement à l'aide de la commande suivante :

```
aws events put-rule \
--name cwe-test \
--event-pattern '{"detail-type": ["EMR Job Run New Driver Attempt"]}'
```

L'événement renverra des informations sur le `newDriverPodName`, l'horodatage `newDriverCreatedAt`, le `previousDriverFailureMessage` et les

`currentAttemptCount` des pilotes de tâche. Ces événements ne seront pas créés si la politique de relance est désactivée.

Pour plus d'informations sur la façon de surveiller votre travail à l'aide d' CloudWatch événements, consultezSurveillez les offres d'emploi avec Amazon CloudWatch Events.

Recherche de journaux pour les pilotes et les exécuteurs

Les noms des pods de pilotes suivent le format `spark-<job id>-driver-<random-suffix>`. Le même `random-suffix` est ajouté aux noms des pods d'exécuteur générés par le pilote. Lorsque vous utilisez ce `random-suffix`, vous pouvez trouver les journaux d'un pilote et de ses exécuteurs associés. Le `random-suffix` n'est présent que si la politique de relance est activée pour la tâche ; dans le cas contraire, le `random-suffix` est absent.

Pour plus d'informations sur la manière de configurer les tâches avec la configuration de surveillance pour la journalisation, consultez Exécution d'une application Spark.

Utilisation de la rotation des journaux des événements Spark

Avec Amazon EMR en version 6.3.0 et ultérieure, vous pouvez activer la fonctionnalité de rotation des journaux des événements Spark pour Amazon EMR on EKS. Au lieu de générer un seul fichier journal des événements, cette fonctionnalité effectue la rotation des fichiers en fonction de l'intervalle de temps configuré et supprime les fichiers journaux des événements les plus anciens.

La rotation des journaux des événements Spark peut vous aider à éviter les problèmes potentiels liés à un fichier journal des événements Spark volumineux généré par des tâches de longue durée ou des tâches en streaming. Par exemple, vous démarrez une tâche Spark de longue durée avec un journal des événements activé avec le paramètre `persistentAppUI`. Le pilote Spark génère un fichier journal des événements. Si la tâche s'exécute pendant des heures ou des jours et que l'espace disque sur le nœud Kubernetes est limité, le fichier journal des événements peut consommer tout l'espace disque disponible. L'activation de la fonctionnalité de rotation des journaux des événements Spark résout le problème en divisant le fichier journal en plusieurs fichiers et en supprimant les fichiers les plus anciens.

> **Note**
>
> Cette fonctionnalité n'est disponible qu'avec Amazon EMR on EKS. Amazon EMR exécuté sur Amazon EC2 ne prend pas en charge la rotation du journal des événements Spark.

Pour activer la fonctionnalité de rotation des journaux des événements Spark, configurez les paramètres Spark suivants :

- `spark.eventLog.rotation.enabled` : active la rotation des journaux. Ce paramètre est désactivé par défaut dans le fichier de configuration de Spark. Réglez-le sur « true » pour activer cette fonctionnalité.
- `spark.eventLog.rotation.interval` : indique l'intervalle de temps pour la rotation des journaux. La valeur minimale est 60 secondes. La valeur par défaut est de 300 secondes.
- `spark.eventLog.rotation.minFileSize` : indique une taille de fichier minimale pour la rotation du fichier journal. La valeur minimale et par défaut est de 1 Mo.
- `spark.eventLog.rotation.maxFilesToRetain` : indique le nombre de fichiers journaux en rotation à conserver pendant le nettoyage. La plage valide est comprise entre 1 et 10. La valeur par défaut est 2.

Vous pouvez indiquer ces paramètres dans la section `sparkSubmitParameters` de l'API `StartJobRun`, comme le montre l'exemple ci-dessous.

```
"sparkSubmitParameters": "--class org.apache.spark.examples.SparkPi --conf
 spark.eventLog.rotation.enabled=true --conf spark.eventLog.rotation.interval=300 --
conf spark.eventLog.rotation.minFileSize=1m --conf
 spark.eventLog.rotation.maxFilesToRetain=2"
```

Utilisation de la rotation des journaux des conteneurs Spark

Avec Amazon EMR en version 6.11.0 et ultérieure, vous pouvez activer la fonctionnalité de rotation des journaux des conteneurs Spark pour Amazon EMR on EKS. Au lieu de générer un seul fichier journal `stdout` ou `stderr`, cette fonctionnalité effectue une rotation des fichiers en fonction de la taille de rotation configurée et supprime les fichiers journaux les plus anciens du conteneur.

La rotation des journaux des conteneurs Spark peut vous aider à éviter les problèmes potentiels liés aux fichiers journaux Spark volumineux générés pour les tâches de longue durée ou en streaming. Par exemple, vous pouvez démarrer une tâche Spark de longue durée et le pilote Spark génère un fichier journal du conteneur. Si la tâche s'exécute pendant des heures ou des jours et que l'espace disque sur le nœud Kubernetes est limité, le fichier journal du conteneur peut consommer tout l'espace disque disponible. Lorsque vous activez la rotation des journaux des conteneurs Spark, vous divisez le fichier journal en plusieurs fichiers et vous supprimez les fichiers les plus anciens.

Pour activer la fonctionnalité de rotation des journaux des conteneurs Spark, configurez les paramètres Spark suivants :

`containerLogRotationConfiguration`

Incluez ce paramètre dans `monitoringConfiguration` pour activer la rotation des journaux. Ce paramètre est désactivé par défaut. Vous devez utiliser `containerLogRotationConfiguration` en plus de `s3MonitoringConfiguration`.

`rotationSize`

Le paramètre `rotationSize` indique la taille du fichier pour la rotation des journaux. La plage de valeurs possibles est comprise entre `2KB` et `2GB`. La partie unitaire numérique du paramètre `rotationSize` est transmise sous forme d'entier. Les valeurs décimales n'étant pas prises en charge, vous pouvez indiquer une taille de rotation de 1,5 Go, par exemple, avec la valeur `1500MB`.

`maxFilesToKeep`

Le paramètre `maxFilesToKeep` indique le nombre maximum de fichiers à retenir dans le conteneur après la rotation. La valeur minimale est 1 et la valeur maximale est 50.

Vous pouvez indiquer ces paramètres dans la section `monitoringConfiguration` de l'API `StartJobRun`, comme le montre l'exemple ci-dessous. Dans cet exemple, avec `rotationSize = "10 MB"` et `maxFilesToKeep = 3`, Amazon EMR on EKS effectue une rotation de vos journaux à 10 Mo, génère un nouveau fichier journal, puis purge le fichier journal le plus ancien une fois que le nombre de fichiers journaux atteint 3.

```
{
  "name": "my-long-running-job",
  "virtualClusterId": "123456",
  "executionRoleArn": "iam_role_name_for_job_execution",
  "releaseLabel": "emr-6.11.0-latest",
  "jobDriver": {
    "sparkSubmitJobDriver": {
      "entryPoint": "entryPoint_location",
      "entryPointArguments": ["argument1", "argument2", ...],
       "sparkSubmitParameters": "--class main_class --conf spark.executor.instances=2
 --conf spark.executor.memory=2G --conf spark.executor.cores=2 --conf
 spark.driver.cores=1"
    }
```

```
  },
  "configurationOverrides": {
    "applicationConfiguration": [
      {
        "classification": "spark-defaults",
        "properties": {
          "spark.driver.memory":"2G"
         }
      }
    ],
    "monitoringConfiguration": {
      "persistentAppUI": "ENABLED",
      "cloudWatchMonitoringConfiguration": {
        "logGroupName": "my_log_group",
        "logStreamNamePrefix": "log_stream_prefix"
      },
      "s3MonitoringConfiguration": {
        "logUri": "s3://my_s3_log_location"
      },
      "containerLogRotationConfiguration": {
        "rotationSize":"10MB",
        "maxFilesToKeep":"3"
      }
    }
  }
}
```

Pour démarrer une exécution de tâche avec la rotation des journaux des conteneurs Spark, incluez dans la commande `StartJobRun` un chemin d'accès au fichier JSON que vous avez configuré avec ces paramètres.

```
aws emr-containers start-job-run \
--cli-input-json file://path-to-json-request-file
```

Utilisation de la mise à l'échelle automatique verticale avec les tâches Spark sur Amazon EMR

La mise à l'échelle automatique verticale d'Amazon EMR on EKS permet d'adapter automatiquement les ressources de mémoire et de CPU aux besoins de la charge de travail que vous fournissez aux applications Spark sur Amazon EMR. Cela simplifie la gestion des ressources.

Pour suivre l'utilisation des ressources de vos applications Spark sur Amazon EMR, aussi bien en temps réel qu'historiquement, la mise à l'échelle verticale automatique utilise l'outil Vertical Pod Autoscaler (VPA) de Kubernetes. La capacité de mise à l'échelle automatique verticale utilise les données collectées par VPA pour ajuster automatiquement les ressources de mémoire et de CPU attribuées à vos applications Spark. Ce processus simplifié améliore la fiabilité et optimise les coûts.

Rubriques

Configuration de la mise à l'échelle automatique verticale pour Amazon EMR on EKS

Cette rubrique vous aide à préparer votre cluster Amazon EKS à soumettre des tâches Spark Amazon EMR avec mise à l'échelle automatique verticale. Le processus de configuration nécessite que vous confirmiez ou effectuiez les tâches décrites dans les sections suivantes :

Rubriques

Conditions préalables

Effectuez les tâches ci-dessous avant d'installer l'opérateur Kubernetes de mise à l'échelle automatique verticale sur votre cluster. Si vous avez déjà rempli l'une des conditions préalables, vous pouvez l'ignorer et passer à la suivante.

- Installation ou mise à jour vers la dernière version du AWS CLI — Si vous avez déjà installé le AWS CLI, vérifiez que vous disposez de la dernière version.

- Installer kubectl – kubectl est un outil de ligne de commande que vous utilisez pour communiquer avec le serveur d'API Kubernetes. Vous avez besoin de kubectl pour installer et surveiller les artefacts liés à la mise à l'échelle automatique verticale sur votre cluster Amazon EKS.
- Installer le kit SDK de l'opérateur – Amazon EMR on EKS utilise le kit SDK de l'opérateur en tant que gestionnaire de packages pour la durée de vie de l'opérateur de mise à l'échelle automatique verticale que vous installez sur votre cluster.
- Installer Docker – Vous devez accéder à la CLI Docker pour vous authentifier et récupérer les images Docker relatives à la mise à l'échelle automatique verticale à installer sur votre cluster Amazon EKS.
- Installation du serveur Kubernetes Metrics : vous devez d'abord installer le serveur de métriques afin que l'autoscaler vertical du pod puisse récupérer les métriques depuis le serveur d'API Kubernetes.
- Commencez avec Amazon EKS — eksctl (version 1.24 ou supérieure) — La mise à l'échelle automatique verticale est prise en charge par les versions 1.24 et supérieures d'Amazon EKS. Une fois le cluster créé, enregistrez-le pour l'utiliser avec Amazon EMR.
- Sélectionner l'URI d'une image de base Amazon EMR (version 6.10.0 ou supérieure) – La mise à l'échelle automatique verticale est prise en charge par Amazon EMR à partir de la version 6.10.0.

Installation d'Operator Lifecycle Manager (OLM) sur votre cluster Amazon EKS

Utilisez l'interface CLI du kit SDK de l'opérateur pour installer Operator Lifecycle Manager (OLM) sur le cluster Amazon EMR on EKS où vous souhaitez configurer la mise à l'échelle automatique verticale, comme indiqué dans l'exemple ci-dessous. Une fois que vous l'avez configuré, vous pouvez utiliser OLM pour installer et gérer le cycle de vie de l'opérateur de mise à l'échelle automatique verticale d'Amazon EMR.

```
operator-sdk olm install
```

Pour valider l'installation, exécutez la commande `olm status` :

```
operator-sdk olm status
```

Vérifiez que la commande renvoie un résultat positif, similaire à l'exemple ci-dessous :

```
INFO[0007] Successfully got OLM status for version X.XX
```

Si votre installation échoue, consultez Résolution des problèmes de mise à l'échelle automatique verticale d'Amazon EMR on EKS.

Installation de l'opérateur de mise à l'échelle automatique verticale d'Amazon EMR on EKS

Suivez les étapes ci-dessous pour installer l'opérateur de mise à l'échelle automatique verticale sur votre cluster Amazon EKS :

1. Configurez les variables d'environnement ci-dessous que vous utiliserez pour terminer l'installation :

   - **`$REGION`** renvoie à la Région AWS correspondant à votre cluster. Par exemple, `us-west-2`.
   - **`$ACCOUNT_ID`** renvoie à l'identifiant du compte Amazon ECR de votre région. Pour de plus amples informations, veuillez consulter Comptes de registre Amazon ECR par région.
   - **`$RELEASE`** renvoie à la version Amazon EMR que vous souhaitez utiliser pour votre cluster. Avec la mise à l'échelle automatique verticale, vous devez utiliser Amazon EMR en version 6.10.0 ou supérieure.
2. Ensuite, obtenez jetons d'authentification pour le registre Amazon ECR destiné à l'opérateur.

   ```
   aws ecr get-login-password \
    --region region-id | docker login \
    --username AWS \
    --password-stdin $ACCOUNT_ID.dkr.ecr.region-id.amazonaws.com
   ```
3. Installez l'opérateur de mise à l'échelle automatique verticale d'Amazon EMR on EKS à l'aide de la commande suivante :

   ```
   ECR_URL=$ACCOUNT_ID.dkr.ecr.$REGION.amazonaws.com && \
   REPO_DEST=dynamic-sizing-k8s-operator-olm-bundle && \
   BUNDLE_IMG=emr-$RELEASE-dynamic-sizing-k8s-operator && \
   operator-sdk run bundle \
   $ECR_URL/$REPO_DEST/$BUNDLE_IMG\:latest
   ```

   Cela créera une version de l'opérateur de mise à l'échelle automatique verticale dans l'espace de noms par défaut de votre cluster Amazon EKS. Utilisez cette commande pour effectuer l'installation dans un autre espace de noms :

   ```
   operator-sdk run bundle \
   ```

```
$ACCOUNT_ID.dkr.ecr.$REGION.amazonaws.com/dynamic-sizing-k8s-operator-olm-bundle/
emr-$RELEASE-dynamic-sizing-k8s-operator:latest \
-n operator-namespace
```

> **Note**
>
> Si l'espace de noms que vous avez spécifié n'existe pas, OLM n'installera pas l'opérateur. Pour de plus amples informations, veuillez consulter L'espace de noms Kubernetes est introuvable.

4. Vérifiez que vous avez bien installé l'opérateur à l'aide de l'outil de ligne de commande kubectl de Kubernetes.

```
kubectl get csv -n operator-namespace
```

La commande `kubectl` doit renvoyer votre opérateur de mise à l'échelle automatique verticale nouvellement déployé avec un état de phase indiquant Réussi. Si vous rencontrez des difficultés lors de l'installation ou de la configuration, consultez Résolution des problèmes de mise à l'échelle automatique verticale d'Amazon EMR on EKS.

Les premiers pas avec la mise à l'échelle automatique verticale pour Amazon EMR on EKS

Utilisez l'autoscaling vertical pour Amazon EMR sur EKS lorsque vous souhaitez régler automatiquement la mémoire et les ressources du processeur afin de les adapter à la charge de travail de votre application Amazon EMR Spark. Pour plus d'informations, consultez la section Utilisation de la mise à l'échelle automatique verticale avec les tâches Amazon EMR Spark.

Soumission d'une tâche Spark avec mise à l'échelle automatique verticale

Lorsque vous soumettez une tâche via l'StartJobRunAPI, ajoutez les deux configurations suivantes au pilote de votre tâche Spark afin d'activer la mise à l'échelle automatique verticale :

```
"spark.kubernetes.driver.annotation.emr-containers.amazonaws.com/
dynamic.sizing":"true",
"spark.kubernetes.driver.annotation.emr-containers.amazonaws.com/
dynamic.sizing.signature":"YOUR_JOB_SIGNATURE"
```

Dans le code ci-dessus, la première ligne active la fonctionnalité de mise à l'échelle automatique verticale. La ligne suivante est une configuration de signature obligatoire qui vous permet de choisir une signature pour votre tâche.

Pour plus d'informations sur ces configurations et les valeurs de paramètres acceptables, consultez Configuration de la mise à l'échelle automatique verticale pour Amazon EMR on EKS. Par défaut, votre tâche est soumise en mode Désactivé, réservé à la surveillance uniquement, de la mise à l'échelle automatique verticale. Cet état de surveillance vous permet de calculer et de consulter les recommandations en matière de ressources sans procéder à la mise à l'échelle automatique. Pour de plus amples informations, veuillez consulter Modes de mise à l'échelle automatique verticale.

L'exemple suivant montre comment exécuter un exemple de commande `start-job-run` avec la mise à l'échelle automatique verticale :

```
aws emr-containers start-job-run \
--virtual-cluster-id $VIRTUAL_CLUSTER_ID \
--name $JOB_NAME \
--execution-role-arn $EMR_ROLE_ARN \
--release-label emr-6.10.0-latest \
--job-driver '{
  "sparkSubmitJobDriver": {
     "entryPoint": "local:///usr/lib/spark/examples/src/main/python/pi.py"
   }
 }' \
--configuration-overrides '{
    "applicationConfiguration": [{
        "classification": "spark-defaults",
        "properties": {
          "spark.kubernetes.driver.annotation.emr-containers.amazonaws.com/
dynamic.sizing": "true",
          "spark.kubernetes.driver.annotation.emr-containers.amazonaws.com/
dynamic.sizing.signature": "test-signature"
        }
    }]
  }'
```

Vérification de la fonctionnalité de mise à l'échelle automatique verticale

Pour vérifier que la mise à l'échelle automatique verticale fonctionne correctement pour la tâche soumise, utilisez kubectl pour obtenir la ressource personnalisée `verticalpodautoscaler` et consulter vos recommandations de mise à l'échelle. Par exemple, la commande suivante demande

des recommandations sur l'exemple de tâche à partir de la section Soumission d'une tâche Spark avec mise à l'échelle automatique verticale :

```
kubectl get verticalpodautoscalers --all-namespaces \
-l=emr-containers.amazonaws.com/dynamic.sizing.signature=test-signature
```

Le résultat de cette requête devrait ressembler à ce qui suit :

```
NAME                                                              MODE   CPU         MEM
 PROVIDED   AGE
ds-jceyefkxnhrvdzw6djum3naf2abm6o63a6dvjkkedqtkhlrf25eq-vpa   Off    3304504865  True
         87m
```

Si votre résultat ne ressemble pas à cela ou contient un code d'erreur, consultez Résolution des problèmes de mise à l'échelle automatique verticale d'Amazon EMR on EKS pour des étapes permettant de résoudre le problème.

Configuration de la mise à l'échelle automatique verticale pour Amazon EMR on EKS

Vous pouvez configurer l'autoscaling vertical lorsque vous soumettez des tâches Amazon EMR Spark via StartJobRunl'API. Définissez les paramètres de configuration liés à la mise à l'échelle automatique sur le pod du pilote Spark, comme indiqué dans l'exemple Soumission d'une tâche Spark avec mise à l'échelle automatique verticale.

L'opérateur de mise à l'échelle automatique verticale d'Amazon EMR on EKS écoute les pods de pilotes équipés de la fonctionnalité de mise à l'échelle automatique. Il assure ensuite l'intégration avec Vertical Pod Autoscaler (VPA) de Kubernetes en se basant sur les paramètres de ces pods de pilotes. Cela facilite le suivi des ressources et la mise à l'échelle automatique des pods d'exécuteurs Spark.

Les sections suivantes décrivent les paramètres que vous pouvez utiliser lorsque vous configurez la mise à l'échelle automatique verticale pour votre cluster Amazon EKS.

> **Note**
>
> Configurez le paramètre de basculement de fonctionnalité sous forme d'étiquette et définissez les autres paramètres sous forme d'annotations sur le pod du pilote Spark.

Les paramètres de mise à l'échelle automatique appartiennent au domaine `emr-containers.amazonaws.com/` et portent le préfixe `dynamic.sizing`.

Paramètres requis

Vous devez inclure les deux paramètres ci-dessous dans le pilote de tâche Spark lorsque vous soumettez votre tâche :

Clé	Description	Valeurs acceptées	Valeur par défaut	Type	Paramètre Spark[1]
`dynamic.sizing`	Basculement de fonctionnalité	`true`, `false`	non défini	étiquette	`spark.kubernetes.driver.label.emr-containers.amazonaws.com/dynamic.sizing`
`dynamic.sizing.signature`	Signature de la tâche	string	non défini	annotation	`spark.kubernetes.driver.annotation.emr-containers.amazonaws.com/dynamic.sizing.signature`

[1] Utilisez ce paramètre en tant que `SparkSubmitParameter` ou `ConfigurationOverride` dans l'API `StartJobRun`.

- **`dynamic.sizing`** – Vous pouvez activer ou désactiver la mise à l'échelle automatique verticale à l'aide de l'étiquette `dynamic.sizing`. Pour activer la mise à l'échelle automatique verticale, attribuez à `dynamic.sizing` la valeur `true` sur le pod du pilote Spark. Si vous omettez cette étiquette ou si vous lui attribuez une valeur autre que `true`, la mise à l'échelle automatique verticale est désactivée.
- **`dynamic.sizing.signature`** – Définissez la signature de la tâche à l'aide de l'annotation `dynamic.sizing.signature` sur le pod du pilote. La mise à l'échelle automatique verticale compile les données d'utilisation des ressources sur diverses exécutions de tâches Spark d'Amazon EMR afin de fournir des recommandations concernant les ressources. Vous fournissez l'identifiant unique pour relier les tâches entre elles.

> **Note**
>
> Si votre tâche se reproduit à un intervalle fixe, par exemple tous les jours ou toutes les semaines, la signature de votre tâche doit rester la même pour chaque nouvelle instance de la tâche. Cela garantit que la mise à l'échelle automatique verticale peut calculer et agréger les recommandations au cours des différentes étapes de la tâche.

[1] Utilisez ce paramètre en tant que `SparkSubmitParameter` ou `ConfigurationOverride` dans l'API `StartJobRun`.

Paramètres facultatifs

La mise à l'échelle automatique verticale prend également en charge les paramètres facultatifs ci-dessous. Définissez-les sous forme d'annotations sur le pod du pilote.

Clé	Description	Valeurs acceptées	Valeur par défaut	Type	Paramètre Spark[1]
`dynamic.sizing.mode`	Mode de mise à l'échelle automatique verticale	`Off`, `Initial`, `Auto`	`Off`	annotation	`spark.kubernetes.driver.annotation.emr-containers.amazonaws.com`

Clé	Description	Valeurs acceptées	Valeur par défaut	Type	Paramètre Spark[1]
					`/dynamic.sizing.mode`
`dynamic.sizing.scale.memory`	Permet la mise à l'échelle de la mémoire	*`true`*, *`false`*	`true`	annotation	`spark.kubernetes.driver.annotation.emr-containers.amazonaws.com/dynamic.sizing.scale.memory`
`dynamic.sizing.scale.cpu`	Activer ou désactiver la mise à l'échelle de la CPU	*`true`*, *`false`*	`false`	annotation	`spark.kubernetes.driver.annotation.emr-containers.amazonaws.com/dynamic.sizing.scale.cpu`

Clé	Description	Valeurs acceptées	Valeur par défaut	Type	Paramètre Spark[1]
dynamic.sizing.scale.memory.min	Limite minimale de la mise à l'échelle de la mémoire	chaîne, quantité de ressources K8s, p. ex. : 1G	non défini	annotation	spark.kubernetes.driver.annotation.emr-containers.amazonaws.com/dynamic.sizing.scale.memory.min
dynamic.sizing.scale.memory.max	Limite maximale de mise à l'échelle de la mémoire	chaîne, quantité de ressources K8s, p. ex. : 4G	non défini	annotation	spark.kubernetes.driver.annotation.emr-containers.amazonaws.com/dynamic.sizing.scale.memory.max

Clé	Description	Valeurs acceptées	Valeur par défaut	Type	Paramètre Spark[1]
dynamic.sizing.scale.cpu.min	Limite minimale de la mise à l'échelle de la CPU	chaîne, quantité de ressources K8s, p. ex. : 1	non défini	annotation	spark.kubernetes.driver.annotation.emr-containers.amazonaws.com/dynamic.sizing.scale.cpu.min
dynamic.sizing.scale.cpu.max	Limite maximale de la mise à l'échelle de la CPU	chaîne, quantité de ressources K8s, p. ex. : 2	non défini	annotation	spark.kubernetes.driver.annotation.emr-containers.amazonaws.com/dynamic.sizing.scale.cpu.max

Modes de mise à l'échelle automatique verticale

Le paramètre `mode` correspond aux différents modes de mise à l'échelle automatique pris en charge par VPA. Utilisez l'annotation `dynamic.sizing.mode` sur le pod du pilote pour définir le mode. Les valeurs suivantes sont prises en charge pour ce paramètre :

- Désactivé – Mode de simulation où vous pouvez consulter les recommandations, mais la mise à l'échelle automatique n'est pas effectuée. Il s'agit du mode par défaut pour la mise à l'échelle automatique verticale. Dans ce mode, la ressource Vertical Pod Autoscaler associée calcule les

recommandations, et vous pouvez consulter ces recommandations à l'aide d'outils tels que kubectl, Prometheus et Grafana.

- Initial – Dans ce mode, VPA redimensionne automatiquement les ressources au démarrage de la tâche si des recommandations sont disponibles sur la base de l'historique des exécutions de la tâche, comme dans le cas d'une tâche récurrente.
- Automatique – Dans ce mode, VPA expulse les pods d'exécuteurs Spark et les met automatiquement à l'échelle avec les paramètres de ressources recommandés lorsque le pod du pilote Spark les redémarre. VPA expulse parfois les pods d'exécuteurs Spark en cours d'exécution, ce qui peut entraîner une latence supplémentaire lorsqu'il réessaie l'exécuteur interrompu.

Mise à l'échelle des ressources

Lorsque vous configurez la mise à l'échelle automatique verticale, vous pouvez choisir de mettre à l'échelle les ressources de CPU et de mémoire. Définissez les annotations `dynamic.sizing.scale.cpu` et `dynamic.sizing.scale.memory` sur `true` ou `false`. Par défaut, la mise à l'échelle de la CPU est définie sur `false`, et la mise à l'échelle de la mémoire est définie sur `true`.

Ressources minimales et maximales (limites)

En option, vous pouvez également définir des limites pour les ressources de CPU et de mémoire. Choisissez une valeur minimale et maximale pour ces ressources avec les annotations `dynamic.sizing.[memory/cpu].[min/max]` lorsque vous activez la mise à l'échelle automatique. Par défaut, les ressources ne sont pas limitées. Définissez les annotations sous forme de chaînes représentant une quantité de ressources Kubernetes. Par exemple, définissez `dynamic.sizing.memory.max` sur `4G` pour représenter 4 Go.

Surveillance de la mise à l'échelle automatique verticale pour Amazon EMR on EKS

Vous pouvez utiliser l'outil de ligne de commande kubectl Kubernetes pour répertorier les recommandations actives et verticales liées à l'autoscaling sur votre cluster. Vous pouvez également consulter les signatures de vos tâches suivies et purger les ressources inutiles associées aux signatures.

Liste des recommandations de mise à l'échelle automatique verticale pour votre cluster

Utilisez kubectl pour obtenir la ressource `verticalpodautoscaler` et en afficher l'état actuel et les recommandations. L'exemple de requête ci-dessous renvoie toutes les ressources actives de votre cluster Amazon EKS.

```
kubectl get verticalpodautoscalers \
-o custom-columns="NAME:.metadata.name,"\
"SIGNATURE:.metadata.labels.emr-containers\.amazonaws\.com/dynamic\.sizing
\.signature,"\
"MODE:.spec.updatePolicy.updateMode,"\
"MEM:.status.recommendation.containerRecommendations[0].target.memory" \
--all-namespaces
```

Le résultat de cette requête ressemble à ce qui suit :

```
NAME                  SIGNATURE                MODE      MEM
ds-example-id-1-vpa   job-signature-1          Off       none
ds-example-id-2-vpa   job-signature-2          Initial   12936384283
```

Interrogation et suppression des recommandations de mise à l'échelle automatique verticale pour votre cluster

Lorsque vous supprimez une ressource d'exécution de tâches à mise à l'échelle automatique verticale Amazon EMR, l'objet VPA associé qui suit et stocke les recommandations est automatiquement effacé.

L'exemple ci-dessous utilise kubectl pour purger les recommandations pour une tâche identifiée par une signature :

```
kubectl delete jobrun -n emr -l=emr-containers\.amazonaws\.com/dynamic\.sizing
\.signature=integ-test
jobrun.dynamicsizing.emr.services.k8s.aws "ds-job-signature" deleted
```

Si vous ne connaissez pas la signature spécifique de la tâche ou si vous souhaitez purger toutes les ressources du cluster, vous pouvez utiliser `--all` ou `--all-namespaces` dans votre commande au lieu de l'identifiant unique de la tâche, comme le montre l'exemple ci-dessous :

```
kubectl delete jobruns --all --all-namespaces
```

```
jobrun.dynamicsizing.emr.services.k8s.aws "ds-example-id" deleted
```

Désinstallation de l'opérateur de mise à l'échelle automatique verticale d'Amazon EMR on EKS

Si vous souhaitez supprimer l'opérateur de mise à l'échelle automatique verticale de votre cluster Amazon EKS, utilisez la commande `cleanup` avec l'interface CLI du kit SDK de l'opérateur, comme indiqué dans l'exemple ci-dessous. Cette opération supprime également les dépendances en amont qui ont été installées avec l'opérateur, comme Vertical Pod Autoscaler.

```
operator-sdk cleanup emr-dynamic-sizing
```

Si des tâches sont en cours d'exécution sur le cluster lorsque vous supprimez l'opérateur, elles continuent de s'exécuter sans mise à l'échelle automatique verticale. Si vous soumettez des tâches sur le cluster après avoir supprimé l'opérateur, Amazon EMR on EKS ignorera tous les paramètres liés à la mise à l'échelle automatique verticale que vous auriez définis lors de la configuration.

Exécution de charges de travail interactives sur Amazon EMR on EKS

Le point de terminaison interactif est une passerelle qui relie Amazon EMR Studio à Amazon EMR on EKS afin que vous puissiez exécuter des charges de travail interactives. Vous pouvez utiliser des points de terminaison interactifs avec EMR Studio pour exécuter des analyses interactives avec des jeux de données dans des magasins de données tels qu'Amazon S3 et Amazon DynamoDB.

Cas d'utilisation

- Créez un script ETL avec l'expérience de l'IDE EMR Studio. L'IDE ingère des données sur site et les stocke dans Amazon S3 après les avoir transformées en vue d'une analyse ultérieure.
- Utilisez des blocs-notes pour explorer des jeux de données et entraînez un modèle de machine learning pour détecter des anomalies dans les jeux de données.
- Créez des scripts qui génèrent des rapports quotidiens pour les applications analytiques telles que les tableaux de bord commerciaux.

Rubriques

Vue d'ensemble des points de terminaison interactifs

Le point de terminaison interactif permet à des clients interactifs tels qu'Amazon EMR Studio de se connecter à Amazon EMR sur des clusters EKS pour exécuter des charges de travail interactives. Le point de terminaison interactif est soutenu par une passerelle Jupyter Enterprise Gateway qui fournit la capacité de gestion à distance du cycle de vie du noyau dont les clients interactifs ont besoin. Les

noyaux sont des processus spécifiques au langage qui interagissent avec le client Amazon EMR Studio basé sur Jupyter pour exécuter des charges de travail interactives.

Les points de terminaison interactifs prennent en charge les noyaux suivants :

- Python 3
- PySpark sur Kubernetes
- Apache Spark avec Scala

> **Note**
>
> La tarification d'Amazon EMR on EKS s'applique aux points de terminaison et aux noyaux interactifs. Pour plus d'informations, consultez la page de tarification d'Amazon EMR on EKS.

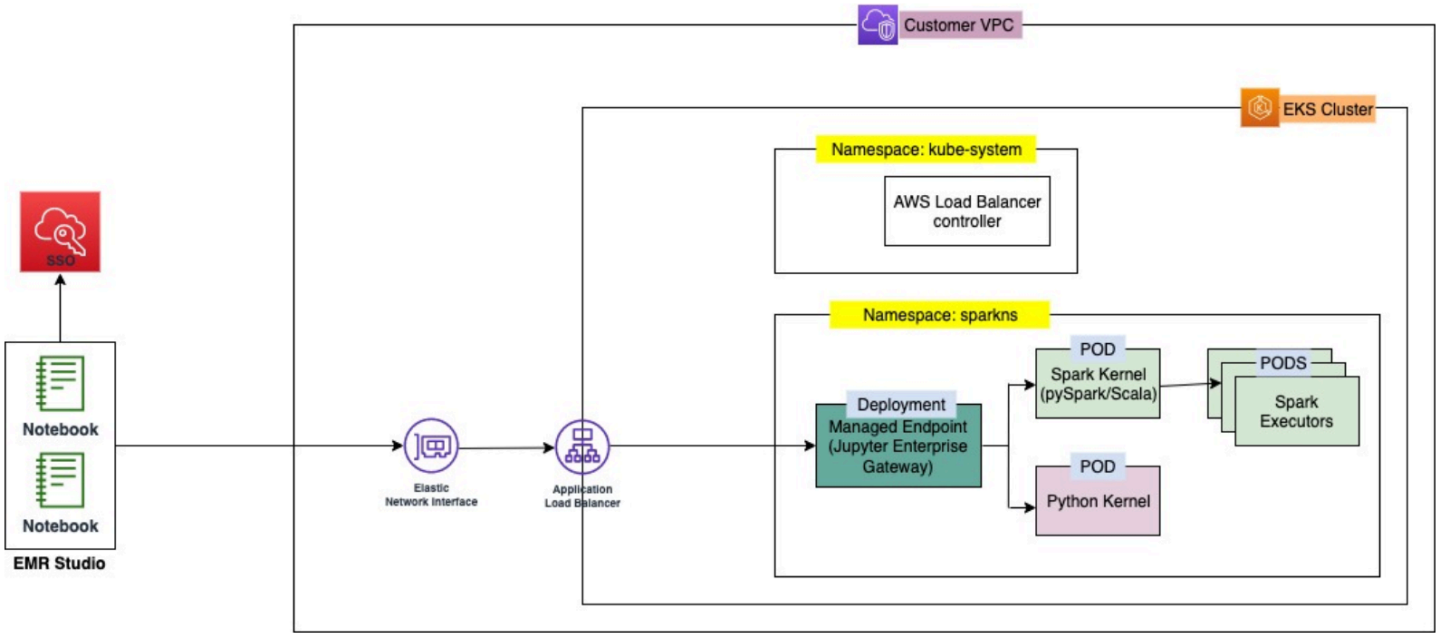
Les entités suivantes sont nécessaires pour qu'EMR Studio se connecte à Amazon EMR on EKS.

- Cluster virtuel Amazon EMR on EKS : le cluster virtuel est un espace de noms Kubernetes que vous enregistrez sur Amazon EMR. Amazon EMR utilise des clusters virtuels pour exécuter des tâches et héberger des points de terminaison. Vous pouvez sauvegarder plusieurs clusters virtuels avec le même cluster physique. Toutefois, chaque cluster virtuel correspond à un espace de noms sur un cluster Amazon EKS. Les clusters virtuels ne créent aucune ressource active qui contribue à votre facture ou qui nécessite une gestion du cycle de vie en dehors du service.
- Point de terminaison interactif Amazon EMR on EKS : le point de terminaison interactif est un point de terminaison HTTPS auquel les utilisateurs d'EMR Studio peuvent connecter un espace de travail. Vous ne pouvez accéder aux points de terminaison HTTPS que depuis votre EMR Studio, et vous les créez dans un sous-réseau privé d'Amazon Virtual Private Cloud (Amazon VPC) pour votre cluster Amazon EKS.

  Les noyaux Python et Spark Scala utilisent les autorisations définies dans votre rôle d'exécution de tâches Amazon EMR on EKS pour en invoquer d'autres. PySpark Services AWS Tous les noyaux et utilisateurs qui se connectent au point de terminaison interactif utilisent le rôle que vous avez spécifié lors de la création du point de terminaison. Nous vous recommandons de créer des points de terminaison distincts pour les différents utilisateurs, et de veiller à ce que les utilisateurs aient des rôles Gestion des identités et des accès AWS (IAM) différents.
- AWS Contrôleur Application Load Balancer : le contrôleur AWS Application Load Balancer gère Elastic Load Balancing pour un cluster Amazon EKS Kubernetes. Le contrôleur provisionne

un équilibreur de charge Application Load Balancer (ALB) lorsque vous créez une ressource Kubernetes Ingress. L'équilibreur de charge ALB expose un service Kubernetes, tel qu'un point de terminaison interactif, en dehors du cluster Amazon EKS, mais au sein du même Amazon VPC. Lorsque vous créez un point de terminaison interactif, une ressource Ingress est également déployée pour exposer le point de terminaison interactif au moyen de l'équilibreur de charge ALB afin que les clients interactifs puissent s'y connecter. Il vous suffit d'installer un contrôleur AWS Application Load Balancer pour chaque cluster Amazon EKS.

Le diagramme suivant décrit l'architecture des points de terminaison interactifs dans Amazon EMR on EKS. Le cluster Amazon EKS comprend le calcul nécessaire pour exécuter les charges de travail analytiques et le point de terminaison interactif. Le contrôleur d'équilibreur de charge Application Load Balancer s'exécute dans l'espace de noms `kube-system` ; les charges de travail et les points de terminaison interactifs s'exécutent dans l'espace de noms que vous indiquez lors de la création du cluster virtuel. Lorsque vous créez un point de terminaison interactif, le plan de contrôle Amazon EMR on EKS crée le déploiement du point de terminaison interactif dans le cluster Amazon EKS. En outre, une instance de l'entrée de l'équilibreur de charge de l'application est créée par le contrôleur de l'équilibreur de AWS charge. L'équilibreur de charge Application Load Balancer fournit une interface externe permettant aux clients comme EMR Studio de se connecter au cluster Amazon EMR et exécuter des charges de travail interactives.



Conditions préalables à la création d'un point de terminaison interactif sur Amazon EMR on EKS

Cette section décrit les conditions préalables à la configuration d'un point de terminaison interactif qu'EMR Studio peut utiliser pour se connecter à un cluster Amazon EMR on EKS et exécuter des charges de travail interactives.

AWS CLI

Suivez les étapes décrites dans Installer ou mettre à jour vers la dernière version du AWS CLI pour installer la dernière version du AWS Command Line Interface (AWS CLI).

Installation d'eksctl

Suivez les étapes décrites dans Installer kubectl pour installer la dernière version d'eksctl. Si vous utilisez Kubernetes en version 1.22 ou ultérieure pour votre cluster Amazon EKS, utilisez une version eksctl supérieure à 0.117.0.

Cluster Amazon EKS

Créez un cluster Amazon EKS. Enregistrez le cluster en tant que cluster virtuel dans Amazon EMR on EKS. Les exigences et considérations pour ce cluster sont les suivantes :

- Le cluster doit se trouver dans le même cloud privé virtuel (VPC) Amazon que votre EMR Studio.
- Le cluster doit disposer d'au moins un sous-réseau privé pour activer les points de terminaison interactifs, pour lier les référentiels Git et pour lancer l'équilibreur de charge Application Load Balancer en mode privé.
- Il doit y avoir au moins un sous-réseau privé en commun entre votre EMR Studio et le cluster Amazon EKS que vous utilisez pour enregistrer votre cluster virtuel. Ainsi, votre point de terminaison interactif apparaît comme une option dans vos espaces de travail Studio et active la connectivité de Studio à l'équilibreur de charge Application Load Balancer.

  Vous pouvez choisir entre deux méthodes pour connecter votre Studio et votre cluster Amazon EKS :

  - Créez un cluster Amazon EKS et associez-le aux sous-réseaux appartenant à votre EMR Studio.

  - Vous pouvez également créer un EMR Studio et spécifier les sous-réseaux privés de votre cluster Amazon EKS.
- ARM optimisé pour Amazon EKS Amazon Linux AMIs n'est pas pris en charge pour Amazon EMR sur les points de terminaison interactifs EKS.
- Seuls les groupes de nœuds gérés par Amazon EKS et les nœuds provisionnés par Karpenter sont pris en charge.

Autorisation d'Amazon EMR on EKS à accéder aux clusters

Suivez les étapes décrites dans la rubrique Autorisation de l'accès aux clusters pour Amazon EMR on EKS pour accorder à Amazon EMR on EKS l'accès à un espace de noms spécifique de votre cluster.

Activation d'IRSA sur le cluster Amazon EKS

Pour activer les rôles IAM pour les comptes de service (IRSA) sur le cluster Amazon EKS, suivez les étapes décrites dans la rubrique Activation des rôles IAM pour les comptes de service (IRSA).

Création d'un rôle d'exécution des tâches IAM

Vous devez créer un rôle IAM pour exécuter des charges de travail sur Amazon EMR sur des points de terminaison interactifs EKS. Dans cette documentation, nous appelons ce rôle IAM le rôle d'exécution des tâches. Ce rôle IAM est attribué à la fois au conteneur de point de terminaison interactif et aux conteneurs d'exécution réels créés lorsque vous soumettez des tâches avec EMR Studio. Vous aurez besoin du nom Amazon Resource Name (ARN) associé à votre rôle d'exécution des tâches pour Amazon EMR on EKS. Deux étapes sont nécessaires pour cela :

- Créer un rôle IAM pour l'exécution des tâches.
- Mettre à jour la politique d'approbation du rôle d'exécution des tâches.

Autorisation des utilisateurs à accéder à Amazon EMR on EKS

L'entité IAM (utilisateur ou rôle) qui effectue la demande de création d'un point de terminaison interactif doit également disposer des autorisations Amazon EC2 et `emr-containers` suivantes. Suivez les étapes décrites dans Autorisation des utilisateurs à accéder à Amazon EMR on EKS pour accorder ces autorisations qui permettent à Amazon EMR on EKS de créer, de gérer et de

supprimer les groupes de sécurité qui limitent le trafic entrant à l'équilibreur de charge de votre point de terminaison interactif.

Les autorisations `emr-containers` suivantes permettent à l'utilisateur d'effectuer des opérations de base sur le point de terminaison interactif :

```
"ec2:CreateSecurityGroup",
"ec2:DeleteSecurityGroup",
"ec2:AuthorizeSecurityGroupEgress",
"ec2:AuthorizeSecurityGroupIngress",
"ec2:RevokeSecurityGroupEgress",
"ec2:RevokeSecurityGroupIngress"

"emr-containers:CreateManagedEndpoint",
"emr-containers:ListManagedEndpoints",
"emr-containers:DescribeManagedEndpoint",
"emr-containers:DeleteManagedEndpoint"
```

Enregistrement du cluster Amazon EKS dans Amazon EMR

Configurez un cluster virtuel et mappez-le à l'espace de noms du cluster Amazon EKS dans lequel vous souhaitez exécuter vos tâches. Pour les AWS Fargate clusters réservés uniquement, utilisez le même espace de noms pour le cluster virtuel Amazon EMR on EKS et pour le profil Fargate.

Pour plus d'informations sur la configuration d'un cluster virtuel Amazon EMR on EKS, consultez Enregistrement du cluster Amazon EKS dans Amazon EMR.

Déployer le AWS Load Balancer Controller sur le cluster Amazon EKS

Un AWS Application Load Balancer est requis pour votre cluster Amazon EKS. Il vous suffit de configurer un contrôleur d'équilibreur de charge Application Load Balancer pour chaque cluster Amazon EKS. Pour plus d'informations sur la configuration du contrôleur AWS Application Load Balancer, consultez la section Installation du module complémentaire AWS Load Balancer Controller dans le guide de l'utilisateur Amazon EKS.

Création d'un point de terminaison interactif pour votre cluster virtuel

Cette rubrique décrit deux manières de créer un point de terminaison interactif à l'aide de l'interface de ligne de AWS commande (AWS CLI) et inclut des détails sur les paramètres de configuration disponibles.

Création d'un point de terminaison interactif à l'aide de la commande **`create-managed-endpoint`**

Indiquez les paramètres de la commande `create-managed-endpoint` comme suit. Amazon EMR on EKS prend en charge la création de points de terminaison interactifs avec les versions 6.7.0 et supérieures d'Amazon EMR.

```
aws emr-containers create-managed-endpoint \
--type JUPYTER_ENTERPRISE_GATEWAY \
--virtual-cluster-id 1234567890abcdef0xxxxxxxx \
--name example-endpoint-name \
--execution-role-arn arn:aws:iam::444455556666:role/JobExecutionRole \
--release-label emr-6.9.0-latest \
--configuration-overrides '{
    "applicationConfiguration": [{
        "classification": "spark-defaults",
        "properties": {
            "spark.driver.memory": "2G"
        }
    }],
    "monitoringConfiguration": {
        "cloudWatchMonitoringConfiguration": {
            "logGroupName": "log_group_name",
            "logStreamNamePrefix": "log_stream_prefix"
        },
        "persistentAppUI": "ENABLED",
        "s3MonitoringConfiguration": {
            "logUri": "s3://my_s3_log_location"
        }
    }
}'
```

Pour de plus amples informations, veuillez consulter Paramètres de création d'un point de terminaison interactif.

Création d'un point de terminaison interactif avec les paramètres spécifiés dans un fichier JSON

1. Créez un fichier `create-managed-endpoint-request.json` et indiquez les paramètres requis pour votre point de terminaison, comme indiqué dans le fichier JSON suivant :

```
{
    "name": "MY_TEST_ENDPOINT",
    "virtualClusterId": "MY_CLUSTER_ID",
    "type": "JUPYTER_ENTERPRISE_GATEWAY",
    "releaseLabel": "emr-6.9.0-latest",
    "executionRoleArn": "arn:aws:iam::444455556666:role/JobExecutionRole",
    "configurationOverrides":
    {
        "applicationConfiguration":
        [
            {
                "classification": "spark-defaults",
                "properties":
                {
                    "spark.driver.memory": "8G"
                }
            }
        ],
        "monitoringConfiguration":
        {
            "persistentAppUI": "ENABLED",
            "cloudWatchMonitoringConfiguration":
            {
                "logGroupName": "my_log_group",
                "logStreamNamePrefix": "log_stream_prefix"
            },
            "s3MonitoringConfiguration":
            {
                "logUri": "s3://my_s3_log_location"
            }
        }
    }
}
```

2. Utilisez la commande `create-managed-endpoint` avec un chemin d'accès au fichier `create-managed-endpoint-request.json` stocké localement ou dans Amazon S3.

```
aws emr-containers create-managed-endpoint \
--cli-input-json  file://./create-managed-endpoint-request.json --region AWS-Region
```

Résultat de la création d'un point de terminaison interactif

Vous devriez voir le résultat suivant dans le terminal. Le résultat comprend le nom et l'identifiant de votre nouveau point de terminaison interactif :

```
{
    "id": "1234567890abcdef0",
    "name": "example-endpoint-name",
    "arn": "arn:aws:emr-containers:us-west-2:111122223333:/
virtualclusters/444455556666/endpoints/444455556666",
    "virtualClusterId": "111122223333xxxxxxxx"
}
```

L'exécution de `aws emr-containers create-managed-endpoint` crée un certificat auto-signé qui permet la communication HTTPS entre EMR Studio et le serveur de point de terminaison interactif.

Si vous exécutez `create-managed-endpoint` et que vous n'avez pas rempli les conditions préalables, Amazon EMR renvoie un message d'erreur indiquant les actions à entreprendre pour continuer.

Paramètres de création d'un point de terminaison interactif

Rubriques

- Paramètres requis pour les points de terminaison interactifs
- Paramètres facultatifs pour les points de terminaison interactifs

Paramètres requis pour les points de terminaison interactifs

Vous devez spécifier les paramètres suivants lorsque vous créez un point de terminaison interactif :

`--type`

Utilisez `JUPYTER_ENTERPRISE_GATEWAY`. C'est le seul type pris en charge.

`--virtual-cluster-id`

L'identifiant du cluster virtuel que vous avez enregistré dans Amazon EMR on EKS.

`--name`

Nom descriptif du point de terminaison interactif qui permet aux utilisateurs d'EMR Studio de le sélectionner dans la liste déroulante.

`--execution-role-arn`

L'Amazon Resource Name (ARN) de votre rôle d'exécution de tâches IAM pour Amazon EMR on EKS qui a été créé dans le cadre des conditions préalables.

`--release-label`

L'étiquette de la version d'Amazon EMR à utiliser pour le point de terminaison. Par exemple, `emr-6.9.0-latest`. Amazon EMR on EKS prend en charge les points de terminaison interactifs avec les versions 6.7.0 et supérieures d'Amazon EMR.

Paramètres facultatifs pour les points de terminaison interactifs

En option, vous pouvez également spécifier les paramètres suivants lorsque vous créez un point de terminaison interactif :

`--configuration-overrides`

Pour remplacer les configurations par défaut des applications, il faut fournir un objet de configuration. Vous pouvez utiliser une syntaxe abrégée pour fournir la configuration, ou vous pouvez faire référence à l'objet de configuration dans un fichier JSON.

Les objets de configuration sont composés d'une classification, de propriétés et de configurations imbriquées en option. Les propriétés sont les paramètres que vous souhaitez remplacer dans ce fichier. Vous pouvez spécifier plusieurs classifications pour plusieurs applications d'un seul objet JSON. Les classifications de configuration disponibles varient selon la version d'Amazon EMR on EKS. Pour la liste des classifications de configuration disponibles pour chaque version d'Amazon EMR on EKS, consultez Versions Amazon EMR on EKS. Outre les classifications de configuration

répertoriées pour chaque version, les points de terminaison interactifs apportent la classification supplémentaire `jeg-config`. Pour de plus amples informations, veuillez consulter Options de configuration de Jupyter Enterprise Gateway (JEG).

Configuration des paramètres pour les points de terminaison interactifs

Cette section contient une série de rubriques qui couvrent les différentes configurations des points de terminaison interactifs et des paramètres des pods. Ils vous permettent de surveiller et de résoudre les défaillances, d'envoyer des informations de journal à Amazon S3 ou à Amazon S3 Amazon CloudWatch Logs, ou de créer des points de terminaison interactifs dans lesquels vous spécifiez des modèles de pods personnalisés.

Rubriques

- Surveillance des tâches Spark
- Spécification de modèles de pods personnalisés avec des points de terminaison interactifs
- Déploiement d'un pod JEG sur un groupe de nœuds
- Options de configuration de Jupyter Enterprise Gateway (JEG)
- Modification des paramètres PySpark de session
- Image de noyau personnalisée avec point de terminaison interactif

Surveillance des tâches Spark

Afin de pouvoir surveiller et résoudre les défaillances, configurez vos points de terminaison interactifs afin que les tâches initiées avec le point de terminaison puissent envoyer des informations de journal à Amazon S3, Amazon CloudWatch Logs ou aux deux. Les sections suivantes décrivent comment envoyer les journaux d'application Spark à Amazon S3 pour les tâches Spark que vous lancez avec Amazon EMR sur des points de terminaison interactifs EKS.

Configuration de la politique IAM pour les journaux Amazon S3

Avant que vos noyaux puissent envoyer des données de journal à Amazon S3, la politique d'autorisations pour le rôle d'exécution des tâches doit inclure les autorisations suivantes. *amzn-s3-demo-destination-bucket*Remplacez-le par le nom de votre bucket de journalisation.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket",
        "arn:aws:s3:::amzn-s3-demo-bucket/*"
      ],
      "Sid": "AllowS3Putobject"
    }
  ]
}
```

> **Note**
>
> Amazon EMR on EKS peut également créer un compartiment S3. Si aucun compartiment S3 n'est disponible, incluez l'autorisation `s3:CreateBucket` dans la politique IAM.

Une fois que vous avez accordé à votre rôle d'exécution les autorisations nécessaires pour envoyer des journaux au compartiment S3, vos données de journal sont envoyées aux emplacements Amazon S3 suivants. Cela se produit lorsque `s3MonitoringConfiguration` est transmise dans la section `monitoringConfiguration` d'une demande `create-managed-endpoint`.

- Journaux de pilote – `logUri/virtual-cluster-id/endpoints/endpoint-id/containers/spark-application-id/spark-application-id-driver/(stderr.gz/stdout.gz)`
- Journaux d'exécuteur – `logUri/virtual-cluster-id/endpoints/endpoint-id/containers/spark-application-id/executor-pod-name-exec-<Number>/(stderr.gz/stdout.gz)`

> **Note**
>
> Amazon EMR on EKS ne charge pas les journaux des points de terminaison dans votre compartiment S3.

Spécification de modèles de pods personnalisés avec des points de terminaison interactifs

Vous pouvez créer des points de terminaison interactifs dans lesquels vous indiquez des modèles de pods personnalisés pour les pilotes et les exécuteurs. Les modèles de pods sont des spécifications qui déterminent comment exécuter chaque pod. Vous pouvez utiliser des fichiers de modèles de pods pour définir les configurations de pods de pilotes ou d'exécuteurs que les configurations Spark ne prennent pas en charge. Les modèles de pods sont actuellement pris en charge dans les versions 6.3.0 et supérieures d'Amazon EMR.

Pour plus d'informations sur les modèles de pods, consultez la rubrique Utilisation des modèles de pods dans le Guide de développement d'Amazon EMR on EKS.

L'exemple suivant montre comment créer un point de terminaison interactif à l'aide de modèles de pods :

```
aws emr-containers create-managed-endpoint \
    --type JUPYTER_ENTERPRISE_GATEWAY \
    --virtual-cluster-id virtual-cluster-id \
    --name example-endpoint-name \
    --execution-role-arn arn:aws:iam::aws-account-id:role/EKSClusterRole \
    --release-label emr-6.9.0-latest \
    --configuration-overrides '{
        "applicationConfiguration": [
        {
            "classification": "spark-defaults",
            "properties": {
                "spark.kubernetes.driver.podTemplateFile": "path/to/driver/
template.yaml",
                "spark.kubernetes.executor.podTemplateFile": "path/to/executor/
template.yaml"
            }
        }]
    }'
```

Déploiement d'un pod JEG sur un groupe de nœuds

Le placement du pod JEG (Jupyter Enterprise Gateway) est une fonctionnalité qui vous permet de déployer un point de terminaison interactif sur un groupe de nœuds spécifique. Grâce à cette fonctionnalité, vous pouvez configurer des paramètres tels que `instance type` pour le point de terminaison interactif.

Association d'un pod JEG à un groupe de nœuds géré

La propriété de configuration suivante vous permet de spécifier le nom d'un groupe de nœuds géré sur votre cluster Amazon EKS où le pod JEG sera déployé.

```
//payload
--configuration-overrides '{
      "applicationConfiguration": [
            {
                "classification": "endpoint-configuration",
                "properties": {
                    "managed-nodegroup-name": NodeGroupName
                }
            }
        ]
    }'
```

Le groupe de nœuds doit avoir l'étiquette Kubernetes `for-use-with-emr-containers-managed-endpoint-ng=`*`NodeGroupName`* attachée à tous les nœuds qui font partie du groupe de nœuds. Pour répertorier tous les nœuds d'un groupe de nœuds dotés de cette balise, utilisez la commande suivante :

```
kubectl get nodes --show-labels | grep for-use-with-emr-containers-managed-endpoint-
ng=NodeGroupName
```

Si le résultat de la commande ci-dessus ne renvoie pas de nœuds faisant partie de votre groupe de nœuds géré, cela signifie qu'aucun nœud du groupe de nœuds n'a l'étiquette Kubernetes `for-use-with-emr-containers-managed-endpoint-ng=`*`NodeGroupName`* attachée. Dans ce cas, suivez les étapes ci-dessous pour attacher cette étiquette aux nœuds de votre groupe de nœuds.

1. Utilisez la commande suivante pour ajouter l'étiquette Kubernetes `for-use-with-emr-containers-managed-endpoint-ng=`*`NodeGroupName`* à tous les nœuds d'un groupe de nœuds géré *`NodeGroupName`* :

```
kubectl label nodes --selector eks:nodegroup-name=NodeGroupName for-use-with-emr-containers-managed-endpoint-ng=NodeGroupName
```

2. Vérifiez que les nœuds ont été correctement étiquetés à l'aide de la commande suivante :

```
kubectl get nodes --show-labels | grep for-use-with-emr-containers-managed-endpoint-ng=NodeGroupName
```

Le groupe de nœuds géré doit être associé au groupe de sécurité d'un cluster Amazon EKS, ce qui est généralement le cas si vous avez créé votre cluster et votre groupe de nœuds gérés à l'aide de l'outil `eksctl`. Vous pouvez le vérifier dans la AWS console en procédant comme suit.

1. Accédez à votre cluster dans la console Amazon EKS.
2. Allez dans l'onglet de mise en réseau de votre cluster et notez le groupe de sécurité du cluster.
3. Accédez à l'onglet de calcul de votre cluster et cliquez sur le nom du groupe de nœuds géré.
4. Dans l'onglet Détails du groupe de nœuds géré, vérifiez que le groupe de sécurité du cluster que vous avez noté précédemment est répertorié sous Groupes de sécurité.

Si le groupe de nœuds géré n'est pas attaché au groupe de sécurité du cluster Amazon EKS, vous devez attacher la balise `for-use-with-emr-containers-managed-endpoint-sg=ClusterName/NodeGroupName` au groupe de sécurité du groupe de nœuds. Suivez les étapes ci-dessous pour attacher cette étiquette.

1. Accédez à la console Amazon EC2 et cliquez sur les groupes de sécurité dans le volet de navigation de gauche.
2. Sélectionnez le groupe de sécurité de votre groupe de nœuds géré en cochant la case.
3. Sous l'onglet Balises, ajoutez la balise `for-use-with-emr-containers-managed-endpoint-sg=ClusterName/NodeGroupName` à l'aide du bouton Gérer les balises.

Association d'un pod JEG à un groupe de nœuds autogéré

La propriété de configuration suivante vous permet de spécifier le nom d'un groupe de nœuds autogéré ou non géré sur le cluster Amazon EKS où le pod JEG sera déployé.

```
//payload
```

```
--configuration-overrides '{
      "applicationConfiguration": [
           {
               "classification": "endpoint-configuration",
               "properties": {
                   "self-managed-nodegroup-name": NodeGroupName
               }
           }
        ]
    }'
```

Le groupe de nœuds doit avoir l'étiquette Kubernetes `for-use-with-emr-containers-managed-endpoint-ng=`*`NodeGroupName`* attachée à tous les nœuds qui font partie du groupe de nœuds. Pour répertorier tous les nœuds d'un groupe de nœuds dotés de cette balise, utilisez la commande suivante :

```
kubectl get nodes --show-labels | grep for-use-with-emr-containers-managed-endpoint-ng=NodeGroupName
```

Si le résultat de la commande ci-dessus ne renvoie pas de nœuds faisant partie de votre groupe de nœuds autogéré, cela signifie qu'aucun nœud du groupe de nœuds n'a l'étiquette Kubernetes `for-use-with-emr-containers-managed-endpoint-ng=`*`NodeGroupName`* attachée. Dans ce cas, suivez les étapes ci-dessous pour attacher cette étiquette aux nœuds de votre groupe de nœuds.

1. Si vous avez créé le groupe de nœuds autogéré à l'aide de l'outil `eksctl`, utilisez la commande suivante pour ajouter l'étiquette Kubernetes `for-use-with-emr-containers-managed-endpoint-ng=`*`NodeGroupName`* à tous les nœuds du groupe de nœuds autogéré *`NodeGroupName`* en une seule fois.

   ```
   kubectl label nodes --selector alpha.eksctl.io/nodegroup-name=NodeGroupName for-use-with-emr-containers-managed-endpoint-ng=NodeGroupName
   ```

   Si vous n'avez pas utilisé `eksctl` pour créer le groupe de nœuds autogéré, vous devrez remplacer le sélecteur dans la commande ci-dessus par une étiquette Kubernetes différente qui est attachée à tous les nœuds du groupe de nœuds.
2. Utilisez la commande suivante pour vérifier que les nœuds ont été étiquetés correctement :

   ```
   kubectl get nodes --show-labels | grep for-use-with-emr-containers-managed-endpoint-ng=NodeGroupName
   ```

Le groupe de sécurité du groupe de nœuds autogérés doit avoir la balise `for-use-with-emr-containers-managed-endpoint-sg=`*`ClusterName`*`/`*`NodeGroupName`* attachée. Procédez comme suit pour attacher la balise au groupe de sécurité à partir de la AWS Management Console.

1. Accédez à la console Amazon EC2. Dans le volet de navigation de gauche, sélectionnez Groupes de sécurité.
2. Cochez la case à côté du groupe de sécurité pour votre groupe de nœuds autogéré.
3. Sous l'onglet Balises, utilisez le bouton Gérer les balises pour ajouter la balise `for-use-with-emr-containers-managed-endpoint-sg=`*`ClusterName`*`/`*`NodeGroupName`*. Remplacez *`ClusterName`* et *`NodeGroupName`* par les valeurs appropriées.

Association d'un pod JEG à un groupe de nœuds géré à l'aide d'instances à la demande

Vous pouvez également définir des étiquettes supplémentaires, appelées sélecteurs d'étiquettes Kubernetes, afin de spécifier des contraintes ou des restrictions supplémentaires pour exécuter un point de terminaison interactif sur un nœud ou un groupe de nœuds donné. L'exemple suivant montre comment utiliser des instances Amazon EC2 à la demande pour un pod JEG.

```
--configuration-overrides '{
      "applicationConfiguration": [
            {
                "classification": "endpoint-configuration",
                "properties": {
                    "managed-nodegroup-name": NodeGroupName,
                    "node-labels": "eks.amazonaws.com/capacityType:ON_DEMAND"
                }
            }
        ]
    }'
```

> **Note**
>
> Vous ne pouvez utiliser la propriété `node-labels` qu'avec une propriété `managed-nodegroup-name` ou `self-managed-nodegroup-name`.

Options de configuration de Jupyter Enterprise Gateway (JEG)

Amazon EMR on EKS utilise Jupyter Enterprise Gateway (JEG) pour activer les points de terminaison interactifs. Vous pouvez définir les valeurs suivantes pour les configurations JEG répertoriées comme autorisées lorsque vous créez le point de terminaison.

- **`RemoteMappingKernelManager.cull_idle_timeout`** : délai d'expiration en secondes (entier), après quoi un noyau est considéré comme inactif et prêt à être éliminé. Les valeurs de `0` ou moins désactivent l'élimination. Des délais d'expiration trop courts risquent d'entraîner l'élimination des noyaux pour les utilisateurs disposant de connexions réseau médiocres.
- **`RemoteMappingKernelManager.cull_interval`** : intervalle en secondes (entier) pendant lequel il faut vérifier si les noyaux inactifs dépassent la valeur du délai d'élimination.

Modification des paramètres PySpark de session

À partir d'Amazon EMR on EKS version 6.9.0, dans Amazon EMR Studio, vous pouvez ajuster la configuration Spark associée à une PySpark session en exécutant la `%%configure` commande magique dans la cellule du bloc-notes EMR.

L'exemple suivant montre un exemple de charge utile que vous pouvez utiliser pour modifier la mémoire, les cœurs et d'autres propriétés du pilote et de l'exécuteur Spark. Pour les paramètres `conf`, vous pouvez configurer n'importe quelle configuration Spark mentionnée dans la documentation de configuration d'Apache Spark.

```
%%configure -f
{
  "driverMemory": "16G",
  "driverCores": 4,
  "executorMemory" : "32G",
  "executorCores": 2,
  "conf": {
     "spark.dynamicAllocation.maxExecutors" : 10,
     "spark.dynamicAllocation.minExecutors": 1
  }
}
```

L'exemple suivant montre un exemple de charge utile que vous pouvez utiliser pour ajouter des fichiers, des pyFiles et des dépendances JAR à un moteur d'exécution Spark.

```
%%configure -f
{
  "files": "s3://amzn-s3-demo-bucket-emr-eks/sample_file.txt",
  "pyFiles": : "path-to-python-files",
  "jars" : "path-to-jars
}
```

Image de noyau personnalisée avec point de terminaison interactif

Pour garantir que vous disposez des dépendances appropriées pour votre application lorsque vous exécutez des charges de travail interactives à partir d'Amazon EMR Studio, vous pouvez personnaliser les images Docker pour les points de terminaison interactifs et exécuter des images de noyau de base personnalisées. Pour créer un point de terminaison interactif et le connecter à une image Docker personnalisée, suivez les étapes ci-dessous.

> **Note**
>
> Vous ne pouvez remplacer que les images de base. Vous ne pouvez pas ajouter de nouveaux types d'images de noyau.

1. Créez et publiez une image Docker personnalisée. L'image de base contient le moteur d'exécution Spark et les noyaux de bloc-notes qui s'exécutent avec celui-ci. Pour créer l'image, vous pouvez suivre les étapes 1 à 4 dans Instructions de personnalisation des images Docker. À l'étape 1, l'URI de l'image de base de votre fichier Docker doit utiliser `notebook-spark` à la place de `spark`.

   ```
   ECR-registry-account.dkr.ecr.Region.amazonaws.com/notebook-spark/container-image-tag
   ```

   Pour plus d'informations sur la manière de sélectionner Régions AWS et de mettre en conteneur des balises d'image, consultezDétails relatifs à la sélection d'une URI d'image de base.

2. Créez un point de terminaison interactif qui peut être utilisé avec l'image personnalisée.

   a. Créez un fichier JSON `custom-image-managed-endpoint.json` avec le contenu suivant. Cet exemple utilise la version 6.9.0 d'Amazon EMR.

Example

```
{
    "name": "endpoint-name",
    "virtualClusterId": "virtual-cluster-id",
    "type": "JUPYTER_ENTERPRISE_GATEWAY",
    "releaseLabel": "emr-6.9.0-latest",
    "executionRoleArn": "execution-role-arn",
    "configurationOverrides": {
        "applicationConfiguration": [
            {
                "classification": "jupyter-kernel-overrides",
                "configurations": [
                    {
                        "classification": "python3",
                        "properties": {
                            "container-image": "123456789012.dkr.ecr.us-
west-2.amazonaws.com/custom-notebook-python:latest"
                        }
                    },
                    {
                        "classification": "spark-python-kubernetes",
                        "properties": {
                            "container-image": "123456789012.dkr.ecr.us-
west-2.amazonaws.com/custom-notebook-spark:latest"
                        }
                    }
                ]
            }
        ]
    }
}
```

b. Créez un point de terminaison interactif avec les configurations spécifiées dans le fichier JSON, comme indiqué dans l'exemple ci-dessous. Pour de plus amples informations, veuillez consulter Création d'un point de terminaison interactif à l'aide de la commande `create-managed-endpoint`.

```
aws emr-containers create-managed-endpoint --cli-input-json custom-image-
managed-endpoint.json
```

3. Connectez-vous au point de terminaison interactif via EMR Studio. Pour plus d'informations et pour connaître les étapes à suivre, consultez la section Connexion depuis Studio dans la section Amazon EMR on EKS de la documentation de AWS Workshop Studio.

Surveillance des points de terminaison interactifs

Avec Amazon EMR on EKS version 6.10 et versions ultérieures, les points de terminaison interactifs émettent des métriques CloudWatch Amazon pour surveiller et résoudre les problèmes liés aux opérations du cycle de vie du noyau. Les métriques sont déclenchées par des clients interactifs, tels qu'EMR Studio ou des blocs-notes Jupyter auto-hébergés. Chacune des opérations prises en charge par les points de terminaison interactifs est associée à des métriques. Les opérations sont modélisées sous forme de dimensions pour chaque métrique, comme indiqué dans le tableau ci-dessous. Les métriques émises par les points de terminaison interactifs sont visibles sous un espace de noms personnalisé EMRContainers, dans votre compte.

Métrique	Description	Unit
RequestCount	Nombre cumulé de demandes d'une opération traitées par le point de terminaison interactif.	Nombre
RequestLatency	Temps écoulé entre l'arrivée d'une demande au point de terminaison interactif et l'envoi d'une réponse par le point de terminaison interactif.	Milliseconde
4 XXError	Émis lorsqu'une demande d'opération aboutit à une erreur 4xx lors du traitement.	Nombre
5 XXError	Émis lorsqu'une demande d'opération aboutit à une erreur 5Xxx du côté du serveur.	Nombre

Métrique	Description	Unit
KernelLaunchSuccess	Applicable uniquement pour l'CreateKernel opération. Cela indique le nombre cumulé de lancements de noyaux qui ont réussi jusqu'à et incluant cette demande.	Nombre
KernelLaunchFailure	Applicable uniquement pour l'CreateKernel opération. Cela indique le nombre cumulé d'échecs de lancement de noyaux jusqu'à et incluant cette demande.	Nombre

Les dimensions suivantes sont associées à chaque métrique interactive du point de terminaison :

- **`ManagedEndpointId`** : identifiant du point de terminaison interactif
- **`OperationName`** : l'opération déclenchée par le client interactif

Les valeurs possibles de la dimension **`OperationName`** sont indiquées dans le tableau suivant :

`operationName`	Description de l'opération
`CreateKernel`	Demandez au point de terminaison interactif de démarrer un noyau.
`ListKernels`	Demandez que le point de terminaison interacti f répertorie les noyaux qui ont été précédemm ent démarrés à l'aide du même jeton de session.
`GetKernel`	Demandez au point de terminaison interacti f d'obtenir des informations sur un noyau spécifique qui a déjà été démarré.

operationName	Description de l'opération
`ConnectKernel`	Demandez au point de terminaison interactif d'établir une connectivité entre le client du bloc-notes et le noyau.
`ConfigureKernel`	Publiez `%%configure magic request` sur un noyau PySpark.
`ListKernelSpecs`	Demande au point de terminaison interactif de répertorier les spécifications disponibles du noyau.
`GetKernelSpec`	Demande au point de terminaison interactif d'obtenir les spécifications d'un noyau qui a été lancé précédemment.
`GetKernelSpecResource`	Demande au point de terminaison interactif d'obtenir des ressources spécifiques associées aux spécifications du noyau précédemment lancé.

Exemples

Pour accéder au nombre total de noyaux lancés pour un point de terminaison interactif un jour donné :

1. Sélectionnez l'espace de noms personnalisé : `EMRContainers`
2. Sélectionnez votre `ManagedEndpointId`, `OperationName – CreateKernel`
3. La métrique `RequestCount` avec les statistiques `SUM` et la période `1 day` fournira toutes les demandes de lancement de noyau effectuées au cours des dernières 24 heures.
4. KernelLaunchSuccess Une métrique avec statistiques `SUM` et période `1 day` fournira toutes les demandes de lancement de noyau réussies effectuées au cours des dernières 24 heures.

Pour accéder au nombre d'échecs du noyau pour un point de terminaison interactif un jour donné :

1. Sélectionnez l'espace de noms personnalisé : EMRContainers
2. Sélectionnez votre `ManagedEndpointId`, `OperationName – CreateKernel`
3. La métrique `KernelLaunchFailure` avec les statistiques `SUM` et la période `1 day` fournira toutes les demandes de lancement de noyau échouées au cours des dernières 24 heures. Vous pouvez également sélectionner la métrique `4XXError` et `5XXError` pour savoir quel type d'échec de lancement de noyau s'est produit.

Utilisation des blocs-notes Jupyter auto-hébergés

Vous pouvez héberger et gérer Jupyter ou des JupyterLab blocs-notes sur une instance Amazon EC2 ou sur votre propre cluster Amazon EKS en tant que bloc-notes Jupyter auto-hébergé. Vous pouvez ensuite exécuter des charges de travail interactives avec vos blocs-notes Jupyter auto-hébergés. Les sections suivantes décrivent le processus de configuration et de déploiement d'un bloc-notes Jupyter auto-hébergé sur un cluster Amazon EKS.

Création d'un bloc-notes Jupyter auto-hébergé sur un cluster EKS

- Création d’un groupe de sécurité
- Création d'un point de terminaison interactif Amazon EMR on EKS
- Récupération de l'URL du serveur de passerelle de votre point de terminaison interactif
- Récupération d'un jeton d'authentification pour la connexion au point de terminaison interactif
- Exemple : déploiement d'un JupyterLab bloc-notes
- Suppression d'un bloc-notes Jupyter auto-hébergé

Création d’un groupe de sécurité

Avant de créer un point de terminaison interactif et d'exécuter un Jupyter ou un JupyterLab bloc-notes auto-hébergé, vous devez créer un groupe de sécurité pour contrôler le trafic entre votre bloc-notes et le point de terminaison interactif. Pour utiliser la console Amazon EC2 ou le SDK Amazon EC2 afin de créer le groupe de sécurité, reportez-vous aux étapes décrites dans la section Créer un groupe de sécurité dans le guide de l'utilisateur Amazon EC2. Vous devez créer le groupe de sécurité dans le VPC où vous souhaitez déployer votre serveur de bloc-notes.

Pour suivre l'exemple de ce guide, utilisez le même VPC que votre cluster Amazon EKS. Si vous souhaitez héberger votre bloc-notes dans un VPC différent de celui de votre cluster Amazon EKS, vous devrez peut-être créer une connexion d'appairage entre les deux. VPCs Pour savoir comment créer une connexion d'appairage entre deux personnes VPCs, consultez la section Créer une connexion d'appairage VPC dans le guide de démarrage Amazon VPC.

Vous avez besoin de l'identifiant du groupe de sécurité pour créer un point de terminaison interactif Amazon EMR on EKS à l'étape suivante.

Création d'un point de terminaison interactif Amazon EMR on EKS

Après avoir créé un groupe de sécurité pour votre bloc-notes, suivez les étapes décrites dans Création d'un point de terminaison interactif pour votre cluster virtuel pour créer un point de terminaison interactif. Vous devez fournir l'identifiant du groupe de sécurité que vous avez créé pour votre bloc-notes dans Création d'un groupe de sécurité.

Insérez l'ID de sécurité *your-notebook-security-group-id* à la place des paramètres de remplacement de configuration suivants :

```
--configuration-overrides '{
    "applicationConfiguration": [
        {
            "classification": "endpoint-configuration",
            "properties": {
                "notebook-security-group-id": "your-notebook-security-group-id"
            }
        }
    ],
    "monitoringConfiguration": {
    ...'
```

Récupération de l'URL du serveur de passerelle de votre point de terminaison interactif

Après avoir créé un point de terminaison interactif, récupérez l'URL du serveur de passerelle à l'aide de la commande `describe-managed-endpoint` de la AWS CLI. Vous avez besoin de cette URL pour connecter votre bloc-notes au point de terminaison. L'URL du serveur de passerelle est un point de terminaison privé.

```
aws emr-containers describe-managed-endpoint \
```

```
--region region \
--virtual-cluster-id virtualClusterId \
--id endpointId
```

Initialement, votre point de terminaison est à l'état CREATING. Après quelques minutes, il passe à l'état ACTIVE. Lorsque le point de terminaison est à l'état ACTIVE, il est prêt à être utilisé.

Prenez note de l'attribut `serverUrl` renvoyé par la commande `aws emr-containers describe-managed-endpoint` à partir du point de terminaison actif. Vous avez besoin de cette URL pour connecter votre bloc-notes au point de terminaison lorsque vous déployez votre Jupyter ou votre bloc-notes auto-hébergé. JupyterLab

Récupération d'un jeton d'authentification pour la connexion au point de terminaison interactif

Pour vous connecter à un point de terminaison interactif depuis un Jupyter ou un JupyterLab bloc-notes, vous devez générer un jeton de session avec l'`GetManagedEndpointSessionCredentials`API. Le jeton sert de preuve d'authentification pour la connexion au serveur de point de terminaison interactif.

La commande suivante est expliquée plus en détail avec un exemple de résultat ci-dessous.

```
aws emr-containers get-managed-endpoint-session-credentials \
--endpoint-identifier endpointArn \
--virtual-cluster-identifier virtualClusterArn \
--execution-role-arn executionRoleArn \
--credential-type "TOKEN" \
--duration-in-seconds durationInSeconds \
--region region
```

endpointArn

L'ARN de votre point de terminaison. Vous pouvez trouver l'ARN dans le résultat d'un appel `describe-managed-endpoint`.

virtualClusterArn

L'ARN du cluster virtuel.

executionRoleArn

L'ARN du rôle d'exécution.

durationInSeconds

La durée en secondes pendant laquelle le jeton est valide. La durée par défaut est de 15 minutes (`900`) et la durée maximale est de 12 heures (`43200`).

region

La même région que votre point de terminaison.

Votre résultat devrait ressembler à l'exemple suivant. Prenez note de la *`session-token`* valeur que vous utiliserez lorsque vous déploierez votre Jupyter ou votre bloc-notes auto-hébergé. JupyterLab

```
{
    "id": "credentialsId",
    "credentials": {
        "token": "session-token"
    },
    "expiresAt": "2022-07-05T17:49:38Z"
}
```

Exemple : déploiement d'un JupyterLab bloc-notes

Une fois les étapes ci-dessus terminées, vous pouvez essayer cet exemple de procédure pour déployer un JupyterLab bloc-notes dans le cluster Amazon EKS avec votre point de terminaison interactif.

1. Créez un espace de noms pour exécuter le serveur de bloc-notes.
2. Créez un fichier local, `notebook.yaml`, avec le contenu suivant. Le contenu du fichier est décrit ci-dessous.

```
apiVersion: v1
kind: Pod
metadata:
  name: jupyter-notebook
  namespace: namespace
spec:
  containers:
  - name: minimal-notebook
    image: jupyter/all-spark-notebook:lab-3.1.4 # open source image
    ports:
    - containerPort: 8888
```

```
    command: ["start-notebook.sh"]
    args: ["--LabApp.token=''"]
    env:
    - name: JUPYTER_ENABLE_LAB
      value: "yes"
    - name: KERNEL_LAUNCH_TIMEOUT
      value: "400"
    - name: JUPYTER_GATEWAY_URL
      value: "serverUrl"
    - name: JUPYTER_GATEWAY_VALIDATE_CERT
      value: "false"
    - name: JUPYTER_GATEWAY_AUTH_TOKEN
      value: "session-token"
```

Si vous déployez le bloc-notes Jupyter sur un cluster Fargate, étiquetez le pod Jupyter avec une étiquette `role`, comme indiqué dans l'exemple ci-dessous :

```
...
metadata:
  name: jupyter-notebook
  namespace: default
  labels:
    role: example-role-name-label
spec:
            ...
```

namespace

L'espace de noms Kubernetes dans lequel le bloc-notes est déployé.

serverUrl

Attribut `serverUrl` renvoyé par la commande `describe-managed-endpoint` dans Récupération de l'URL du serveur de passerelle de votre point de terminaison interactif.

session-token

Attribut `session-token` renvoyé par la commande `get-managed-endpoint-session-credentials` dans Récupération d'un jeton d'authentification pour la connexion au point de terminaison interactif.

KERNEL_LAUNCH_TIMEOUT

Durée en secondes pendant laquelle le point de terminaison interactif attend que le noyau passe à l'état RUNNING. Veillez à ce que le lancement du noyau ait suffisamment de temps pour se terminer en définissant le délai de lancement du noyau sur une valeur appropriée (400 secondes au maximum).

KERNEL_EXTRA_SPARK_OPTS

Vous pouvez éventuellement transmettre des configurations Spark supplémentaires pour les noyaux Spark. Définissez cette variable d'environnement avec les valeurs de la propriété de configuration Spark, comme indiqué dans l'exemple ci-dessous :

```
- name: KERNEL_EXTRA_SPARK_OPTS
  value: "--conf spark.driver.cores=2
          --conf spark.driver.memory=2G
          --conf spark.executor.instances=2
          --conf spark.executor.cores=2
          --conf spark.executor.memory=2G
          --conf spark.dynamicAllocation.enabled=true
          --conf spark.dynamicAllocation.shuffleTracking.enabled=true
          --conf spark.dynamicAllocation.minExecutors=1
          --conf spark.dynamicAllocation.maxExecutors=5
          --conf spark.dynamicAllocation.initialExecutors=1
          "
```

3. Déployez la spécification de pod sur votre cluster Amazon EKS :

```
kubectl apply -f notebook.yaml -n namespace
```

Cela démarrera un JupyterLab bloc-notes minimal connecté à votre point de terminaison interactif Amazon EMR on EKS. Attendez que le pod passe à l'état RUNNING. Vous pouvez vérifier son état à l'aide de la commande suivante :

```
kubectl get pod jupyter-notebook -n namespace
```

Lorsque le pod est prêt, la commande `get pod` renvoie un résultat similaire à celui-ci :

```
NAME              READY   STATUS    RESTARTS   AGE
jupyter-notebook  1/1     Running   0          46s
```

4. Associez le groupe de sécurité du bloc-notes au nœud sur lequel le bloc-notes est programmé.

   a. Identifiez d'abord le nœud sur lequel le pod `jupyter-notebook` est programmé à l'aide de la commande `describe pod`.

   ```
   kubectl describe pod jupyter-notebook -n namespace
   ```

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

   c. Accédez à l'onglet Calcul de votre cluster Amazon EKS et sélectionnez le nœud identifié par la commande `describe pod`. Sélectionnez l'identifiant d'instance du nœud.

   d. Dans le menu Actions, sélectionnez Sécurité > Modifier les groupes de sécurité pour associer le groupe de sécurité que vous avez créé dans Création d'un groupe de sécurité.

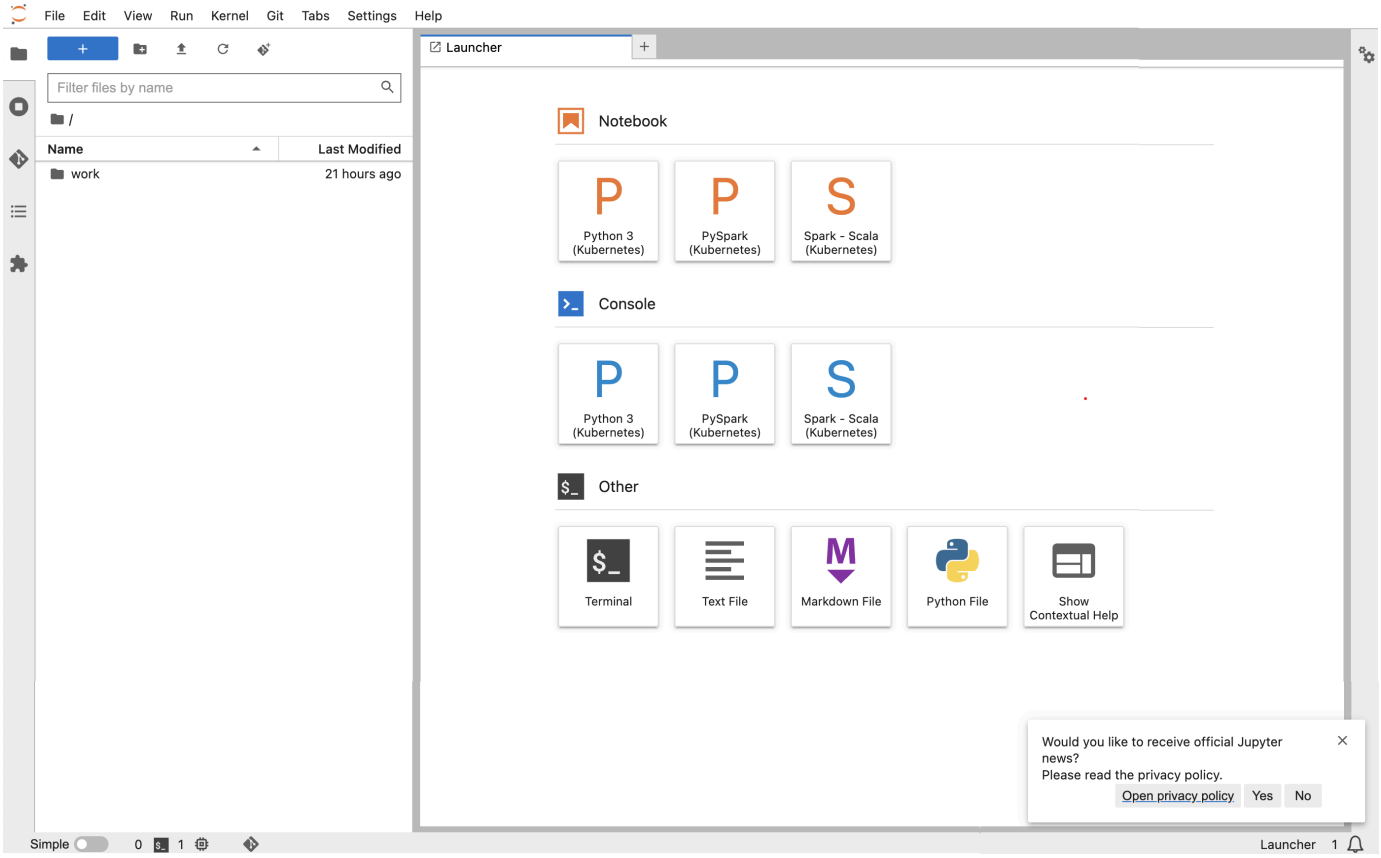
   e. Si vous déployez le module de bloc-notes Jupyter sur AWS Fargate, créez-en un `SecurityGroupPolicy`à appliquer au module de bloc-notes Jupyter avec le libellé du rôle :

   ```
   cat >my-security-group-policy.yaml <<EOF
   apiVersion: vpcresources.k8s.aws/v1beta1
   kind: SecurityGroupPolicy
   metadata:
     name: example-security-group-policy-name
     namespace: default
   spec:
     podSelector:
       matchLabels:
         role: example-role-name-label
     securityGroups:
       groupIds:
         - your-notebook-security-group-id
   EOF
   ```

5. Maintenant, redirigez le port afin que vous puissiez accéder localement à l' JupyterLab interface :

   ```
   kubectl port-forward jupyter-notebook 8888:8888 -n namespace
   ```

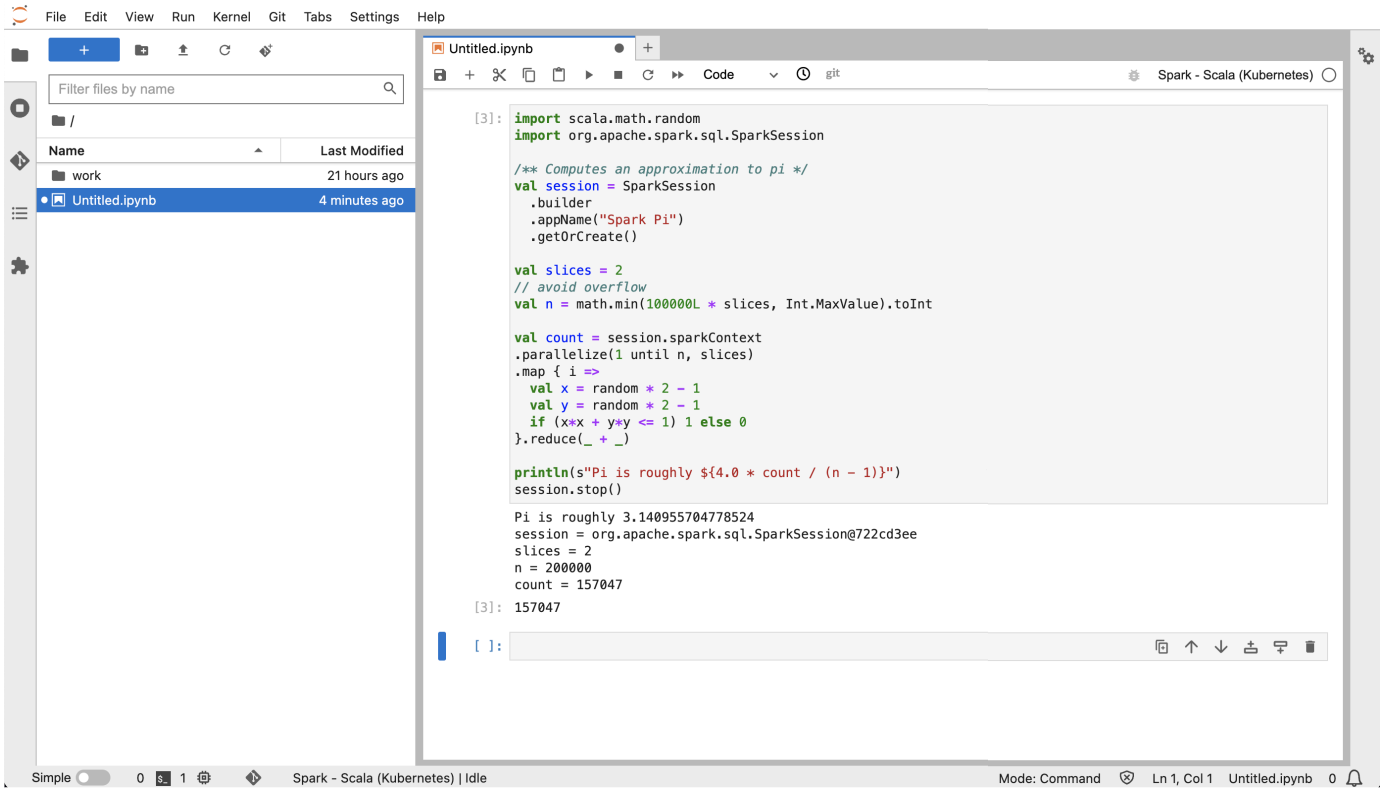
   Une fois que cela est lancé, accédez à votre navigateur local et rendez-vous `localhost:8888` pour voir l' JupyterLab interface :



6. À partir de JupyterLab, créez un nouveau bloc-notes Scala. Voici un exemple d'extrait de code que vous pouvez exécuter pour calculer approximativement la valeur de Pi :

```
import scala.math.random
import org.apache.spark.sql.SparkSession

/** Computes an approximation to pi */
val session = SparkSession
  .builder
  .appName("Spark Pi")
  .getOrCreate()

val slices = 2
// avoid overflow
val n = math.min(100000L * slices, Int.MaxValue).toInt

val count = session.sparkContext
.parallelize(1 until n, slices)
.map { i =>
  val x = random * 2 - 1
  val y = random * 2 - 1
```

```
  if (x*x + y*y <= 1) 1 else 0
}.reduce(_ + _)

println(s"Pi is roughly ${4.0 * count / (n - 1)}")
session.stop()
```



Suppression d'un bloc-notes Jupyter auto-hébergé

Lorsque vous êtes prêt à supprimer votre bloc-notes auto-hébergé, vous pouvez également supprimer le point de terminaison interactif et le groupe de sécurité. Effectuez les actions dans l'ordre suivant :

1. Utilisez la commande suivante pour supprimer le pod `jupyter-notebook` :

```
kubectl delete pod jupyter-notebook -n namespace
```

2. Supprimez ensuite votre point de terminaison interactif à l'aide de la commande `delete-managed-endpoint`. Pour savoir comment supprimer un point de terminaison interactif, consultez Suppression d'un point de terminaison interactif. Initialement, votre point de

terminaison sera à l'état TERMINATING. Une fois que toutes les ressources ont été nettoyées, elles passent à l'état TERMINATED.

3. Si vous ne prévoyez pas d'utiliser le groupe de sécurité des blocs-notes que vous avez créé dans Création d'un groupe de sécurité pour d'autres déploiements de blocs-notes Jupyter, vous pouvez le supprimer. Pour de plus amples informations, consultez la rubrique Suppression d'un groupe de sécurité dans le Guide de l'utilisateur Amazon EC2.

Obtenir des informations sur les points de terminaison interactifs à l'aide des commandes CLI

Cette rubrique couvre les opérations prises en charge sur un point de terminaison interactif autre que `create-managed-endpoint`.

Récupération des détails du point de terminaison interactif

Après avoir créé un point de terminaison interactif, vous pouvez récupérer ses détails à l'aide de la `describe-managed-endpoint` AWS CLI commande. Insérez vos propres valeurs pour *`managed-endpoint-idvirtual-cluster-id`*, et *`region`* :

```
aws emr-containers describe-managed-endpoint --id managed-endpoint-id \
 --virtual-cluster-id virtual-cluster-id --region region
```

Le résultat ressemble à ce qui suit, avec le point de terminaison spécifié, tel que l'ARN, l'identifiant et le nom.

```
{
   "id": "as3ys2xxxxxxx",
   "name": "endpoint-name",
    "arn": "arn:aws:emr-containers:us-east-1:1828xxxxxxxx:/virtualclusters/
lbhl6kwwyoxxxxxxxxxxxxxxx/endpoints/as3ysxxxxxxxx",
    "virtualClusterId": "lbhl6kwwyoxxxxxxxxxxxxxxx",
    "type": "JUPYTER_ENTERPRISE_GATEWAY",
    "state": "ACTIVE",
    "releaseLabel": "emr-6.9.0-latest",
   "executionRoleArn": "arn:aws:iam::1828xxxxxxxx:role/RoleName",
    "certificateAuthority": {
        "certificateArn": "arn:aws:acm:us-east-1:1828xxxxxxxx:certificate/zzzzzzzz-
e59b-4ed0-aaaa-bbbbbbbbbbbb",
```

```
        "certificateData": "certificate-data"
    },
    "configurationOverrides": {
        "applicationConfiguration": [
            {
                "classification": "spark-defaults",
                "properties": {
                    "spark.driver.memory": "8G"
                }
            }
        ],
        "monitoringConfiguration": {
            "persistentAppUI": "ENABLED",
            "cloudWatchMonitoringConfiguration": {
                "logGroupName": "log-group-name",
                "logStreamNamePrefix": "log-stream-name-prefix"
            },
            "s3MonitoringConfiguration": {
                "logUri": "s3-bucket-name"
            }
        }
    },
   "serverUrl": "https://internal-k8s-namespace-ingressa-aaaaaaaaaa-
zzzzzzzzzz.us-east-1.elb.amazonaws.com:18888 (https://internal-k8s-nspluto-
ingressa-51e860abbd-1620715833.us-east-1.elb.amazonaws.com:18888/)",
    "createdAt": "2022-09-19T12:37:49+00:00",
    "securityGroup": "sg-aaaaaaaaaaaaaa",
    "subnetIds": [
        "subnet-11111111111",
        "subnet-22222222222",
        "subnet-33333333333"
    ],
    "stateDetails": "Endpoint created successfully. It took 3 Minutes 15 Seconds",
    "tags": {}
 }
```


Liste de tous les points de terminaison interactifs associés à un cluster virtuel

Utilisez la `list-managed-endpoints` AWS CLI commande pour récupérer la liste de tous les points de terminaison interactifs associés à un cluster virtuel spécifié. Remplacez `virtual-cluster-id` par l'identifiant de votre cluster virtuel.

```
aws emr-containers list-managed-endpoints --virtual-cluster-id virtual-cluster-id
```

Le résultat de la commande `list-managed-endpoint` est illustré ci-dessous :


```
{
    "endpoints": [{
        "id": "as3ys2xxxxxxx",
        "name": "endpoint-name",
        "arn": "arn:aws:emr-containers:us-east-1:1828xxxxxxxx:/virtualclusters/
lbhl6kwwyoxxxxxxxxxxxxxxx/endpoints/as3ysxxxxxxxx",
        "virtualClusterId": "lbhl6kwwyoxxxxxxxxxxxxxxx",
        "type": "JUPYTER_ENTERPRISE_GATEWAY",
        "state": "ACTIVE",
        "releaseLabel": "emr-6.9.0-latest",
        "executionRoleArn": "arn:aws:iam::1828xxxxxxxx:role/RoleName",
        "certificateAuthority": {
            "certificateArn": "arn:aws:acm:us-east-1:1828xxxxxxxx:certificate/zzzzzzzz-
e59b-4ed0-aaaa-bbbbbbbbbbbb",
            "certificateData": "certificate-data"
        },
        "configurationOverrides": {
            "applicationConfiguration": [{
                "classification": "spark-defaults",
                "properties": {
                    "spark.driver.memory": "8G"
                }
            }],
            "monitoringConfiguration": {
                "persistentAppUI": "ENABLED",
                "cloudWatchMonitoringConfiguration": {
                    "logGroupName": "log-group-name",
                    "logStreamNamePrefix": "log-stream-name-prefix"
                },
                "s3MonitoringConfiguration": {
                    "logUri": "s3-bucket-name"
                }
            }
        },
        "serverUrl": "https://internal-k8s-namespace-ingressa-aaaaaaaaaa-
zzzzzzzzzz.us-east-1.elb.amazonaws.com:18888 (https://internal-k8s-nspluto-
ingressa-51e860abbd-1620715833.us-east-1.elb.amazonaws.com:18888/)",
        "createdAt": "2022-09-19T12:37:49+00:00",
        "securityGroup": "sg-aaaaaaaaaaaaa",
```

```
        "subnetIds": [
            "subnet-11111111111",
            "subnet-22222222222",
            "subnet-33333333333"
        ],
        "stateDetails": "Endpoint created successfully. It took 3 Minutes 15 Seconds",
        "tags": {}
    }]
}
```

Suppression d'un point de terminaison interactif

Pour supprimer un point de terminaison interactif associé à un cluster virtuel Amazon EMR on EKS, utilisez la `delete-managed-endpoint` AWS CLI commande. Lorsque vous supprimez un point de terminaison interactif, Amazon EMR on EKS supprime les groupes de sécurité par défaut créés pour ce point de terminaison.

Utilisez les valeurs des paramètres suivants de la commande :

- --id : identifiant du point de terminaison interactif que vous souhaitez supprimer.
- -- virtual-cluster-id — L'identifiant du cluster virtuel associé au point de terminaison interactif que vous souhaitez supprimer. Il s'agit du même identifiant de cluster virtuel que celui spécifié lors de la création du point de terminaison interactif.

```
aws emr-containers delete-managed-endpoint --id managed-endpoint-id --virtual-cluster-
id virtual-cluster-id
```

La commande renvoie un résultat similaire au suivant pour confirmer que vous avez supprimé le point de terminaison interactif :

```
{
    "id":"8gai4l4exxxxx",
    "virtualClusterId":"0b0qvauoy3ch1nqodxxxxxxxx"
}
```

Lisez, écrivez et chargez des données dans Amazon S3 Express One Zone avec Amazon EMR sur EKS

Avec les versions 7.2.0 et supérieures d'Amazon EMR, vous pouvez utiliser Amazon EMR sur EKS avec la classe de stockage Amazon S3 Express One Zone pour améliorer les performances lorsque vous exécutez des tâches et des charges de travail. S3 Express One Zone est une classe de stockage Amazon S3 à zone unique à hautes performances qui fournit un accès aux données constant à un chiffre en millisecondes pour la plupart des applications sensibles à la latence. À son lancement, S3 Express One Zone offre la latence la plus faible et les meilleures performances de stockage d'objets cloud dans Amazon S3.

Conditions préalables

Avant de pouvoir utiliser S3 Express One Zone avec Amazon EMR sur EKS, vous devez remplir les conditions préalables suivantes :

- Configuration d'Amazon EMR sur EKS terminée.
- Après avoir configuré Amazon EMR sur EKS, créez un cluster virtuel.

Bien démarrer avec S3 Express One Zone

Suivez ces étapes pour commencer à utiliser S3 Express One Zone

1. Ajoutez l'`CreateSession`autorisation à votre rôle d'exécution des tâches. Lorsque S3 Express One Zone exécute initialement une action telle que `GET` ou `PUT` sur un objet S3, la classe de stockage appelle `CreateSession` en votre nom. `LIST` Voici un exemple de la procédure à suivre pour accorder l'`CreateSession`autorisation.

   JSON

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

```
        "arn:aws:s3express:*:*:bucket/DOC-EXAMPLE-BUCKET"
      ],
      "Action": [
        "s3express:CreateSession"
      ],
      "Sid": "AllowS3EXPRESSCreatesession"
    }
  ]
}
```

2. Vous devez utiliser le connecteur Apache Hadoop S3A pour accéder aux compartiments S3 Express. Modifiez donc votre Amazon S3 URIs pour utiliser le `s3a` schéma d'utilisation du connecteur. S'ils n'utilisent pas le schéma, vous pouvez modifier l'implémentation du système de fichiers que vous utilisez pour `s3` et les `s3n` schémas.

   Pour modifier le schéma `s3`, spécifiez les configurations de cluster suivantes :

```
[
  {
    "Classification": "core-site",
    "Properties": {
      "fs.s3.impl": "org.apache.hadoop.fs.s3a.S3AFileSystem",
      "fs.AbstractFileSystem.s3.impl": "org.apache.hadoop.fs.s3a.S3A"
    }
  }
]
```

   Pour modifier le schéma s3n, spécifiez les configurations de cluster suivantes :

```
[
  {
    "Classification": "core-site",
    "Properties": {
      "fs.s3n.impl": "org.apache.hadoop.fs.s3a.S3AFileSystem",
      "fs.AbstractFileSystem.s3n.impl": "org.apache.hadoop.fs.s3a.S3A",
      "fs.s3a.endpoint.region": "us-west-2",
      "fs.s3a.change.detection.mode": "none",
      "fs.s3a.select.enabled": "false"
    }
  },
   {
    "Classification": "spark-defaults",
```

```
    "Properties": {
      "spark.hadoop.fs.s3a.aws.credentials.provider":
 "software.amazon.awssdk.auth.credentials.WebIdentityTokenFileCredentialsProvider",
      "spark.sql.sources.fastS3PartitionDiscovery.enabled": "false"
    }
  }
]
```

3. Dans votre configuration Spark-Submit, utilisez le fournisseur d'informations d'identification d'identité Web.

```
"spark.hadoop.fs.s3a.aws.credentials.provider=com.amazonaws.auth.WebIdentityTokenCredential
```

Surveillance des tâches

Vous pouvez utiliser Amazon CloudWatch Events pour suivre les tâches exécutées sur un cluster virtuel Amazon EMR on EKS. Vous pouvez utiliser les événements pour suivre l'activité et l'état des tâches que vous exécutez sur un cluster virtuel. Les rubriques suivantes vous montrent comment configurer efficacement la surveillance afin de préserver l'intégrité de vos ressources.

Rubriques

Surveillez les offres d'emploi avec Amazon CloudWatch Events

Amazon EMR on EKS émet des événements lorsque l'état d'une exécution de tâche change. Chaque événement fournit des informations, telles que la date et l'heure auxquelles l'événement s'est produit, ainsi que d'autres détails sur l'événement, tels que l'identifiant du cluster virtuel et l'identifiant de l'exécution de tâche qui a été affectée.

Vous pouvez utiliser les événements pour suivre l'activité et l'état des tâches que vous exécutez sur un cluster virtuel. Vous pouvez également utiliser Amazon CloudWatch Events pour définir une action à effectuer lorsqu'une tâche génère un événement correspondant à un modèle que vous spécifiez. Les événements sont utiles pour surveiller un événement spécifique au cours du cycle de vie d'une tâche. Par exemple, vous pouvez surveiller le moment où l'état d'une exécution de tâche passe de `submitted` à `running`. Pour plus d'informations sur CloudWatch les événements, consultez le guide de EventBridge l'utilisateur Amazon.

Le tableau suivant répertorie les événements Amazon EMR on EKS en indiquant l'état ou la modification d'état associée à chaque événement, sa gravité ainsi que les messages correspondants. Chaque événement est représenté sous la forme un objet JSON qui est automatiquement envoyé à un flux d'événements. L'objet JSON inclut des détails supplémentaires sur l'événement. L'objet JSON est particulièrement important lorsque vous configurez des règles pour le traitement des événements à l'aide d' CloudWatch Events, car les règles cherchent à correspondre aux modèles de l'objet JSON.

Pour plus d'informations, consultez les modèles d' EventBridge événements Amazon et Amazon EMR on EKS Events dans le guide de EventBridge l'utilisateur Amazon.

Événements de changement d'état d'exécution de tâche

State	Sévérité	Message
SUBMITTED	INFO	Job Run *JobRunId* (*JobRunName*) a été soumis avec succès *VirtualClusterId* au cluster virtuel à *Time* UTC.
RUNNING	INFO	Job Run *JobRunId* (*JobRunName*) dans un cluster virtuel *VirtualClusterId* a commencé à s'exécuter à*Time*.
TERMINÉ	INFO	Job Run *jobRunId* (*JobRunName*) dans un cluster virtuel *VirtualClusterId* s'est terminé à*Time*. Le Job Run a commencé à s'exécuter à *Time* et a pris *Num* quelques minutes.
CANCELLED	WARN	La demande d'annulation a abouti pour Job Run *JobRunId* (*JobRunName*) dans le cluster virtuel *VirtualClusterId* à *Time* et le Job Run est maintenant annulé.
ÉCHEC	ERROR	Job Run *JobRunId* (*JobRunName*) dans le cluster virtuel *VirtualClusterId* a échoué à*Time*.

Automatisez Amazon EMR sur EKS avec des événements CloudWatch

Vous pouvez utiliser Amazon CloudWatch Events pour automatiser vos AWS services afin de répondre aux événements du système tels que les problèmes de disponibilité des applications ou les modifications des ressources. Les événements des AWS services sont fournis à CloudWatch Events en temps quasi réel. Vous pouvez écrire des règles simples pour préciser les événements qui vous

intéressent et les actions automatisées à effectuer quand un événement correspond à une règle. Les actions pouvant être déclenchées automatiquement sont les suivantes :

- Invoquer une fonction AWS Lambda
- Appel de la fonctionnalité Exécuter la commande d'Amazon EC2
- Relais de l'événement à Amazon Kinesis Data Streams
- Activation d'une machine à AWS Step Functions états
- Notification d'un sujet Amazon Simple Notification Service (SNS) ou d'une file d'attente Amazon Simple Queue Service (SQS)

Voici quelques exemples d'utilisation d' CloudWatch Events avec Amazon EMR sur EKS :

- Activation d'une fonction Lambda lorsqu'une exécution de tâche réussit
- Notification d'une rubrique Amazon SNS lorsqu'une exécution de tâche échoue

CloudWatch Les événements pour « `detail-type:` » `EMR Job Run State Change` » sont générés par Amazon EMR sur EKS pour`SUBMITTED`, `RUNNING``CANCELLED`, `FAILED` et les changements `COMPLETED` d'état.

Exemple : configuration d'une règle qui invoque Lambda

Suivez les étapes ci-dessous pour configurer une règle d' CloudWatch événements qui invoque Lambda lorsqu'un événement « EMR Job Run State Change » se produit.

```
aws events put-rule \
--name cwe-test \
--event-pattern '{"detail-type": ["EMR Job Run State Change"]}'
```

Ajoutez la fonction Lambda que vous possédez comme nouvelle cible et autorisez CloudWatch Events à invoquer la fonction Lambda comme suit. Remplacez *123456789012* par votre ID de compte.

```
aws events put-targets \
--rule cwe-test \
--targets Id=1,Arn=arn:aws:lambda:us-east-1:123456789012:function:MyFunction
```

```
aws lambda add-permission \
--function-name MyFunction \
--statement-id MyId \
--action 'lambda:InvokeFunction' \
--principal events.amazonaws.com
```

> **Note**
>
> Vous ne pouvez pas écrire un programme qui dépend de l'ordre ou de l'existence d'événements de notification, car ces événements peuvent arriver dans un ordre différent ou être absents. Les événements sont générés dans la mesure du possible.

Surveillez le module pilote de la tâche avec une politique de nouvelle tentative à l'aide d'Amazon Events CloudWatch

À l'aide d' CloudWatch événements, vous pouvez surveiller les modules de pilotes créés dans le cadre de tâches soumises à des politiques de nouvelle tentative. Pour plus d’informations, consultez Surveillance d'une tâche à l'aide d'une politique de relance dans ce guide.

Gestion des clusters virtuels

Le cluster virtuel est un espace de noms Kubernetes que vous enregistrez sur Amazon EMR. Vous pouvez créer, décrire, répertorier et supprimer des clusters virtuels. Ils ne consomment pas de ressources supplémentaires dans votre système. Un cluster virtuel unique est mappé à un seul espace de noms Kubernetes. Compte tenu de cette relation, vous pouvez modéliser les clusters virtuels de la même façon que les espaces de noms Kubernetes pour répondre à vos besoins. Découvrez les cas d'utilisation possibles dans la documentation de présentation des concepts de Kubernetes.

Pour enregistrer Amazon EMR dans un espace de noms Kubernetes sur un cluster Amazon EKS, vous avez besoin du nom du cluster EKS et de l'espace de noms configuré pour l'exécution de votre charge de travail. Ces clusters enregistrés dans Amazon EMR sont appelés clusters virtuels, car ils ne gèrent pas le calcul physique ou le stockage, mais pointent vers un espace de noms Kubernetes dans lequel votre charge de travail est planifiée.

> **Note**
>
> Avant de créer un cluster virtuel, vous devez d'abord effectuer les étapes 1 à 8 dans Configuration d'Amazon EMR on EKS.

Rubriques

Création d'un cluster local

Exécutez la commande ci-dessous pour créer un cluster virtuel en enregistrant Amazon EMR dans un espace de noms sur un cluster EKS. *virtual_cluster_name*Remplacez-le par un nom que vous avez fourni pour votre cluster virtuel. Remplacez *eks_cluster_name* par le nom du cluster

EKS. Remplacez le *namespace_name* par l'espace de noms dans lequel vous souhaitez enregistrer Amazon EMR.

```
aws emr-containers create-virtual-cluster \
--name virtual_cluster_name \
--container-provider '{
    "id": "eks_cluster_name",
    "type": "EKS",
    "info": {
        "eksInfo": {
            "namespace": "namespace_name"
        }
    }
}'
```

Vous pouvez également créer un fichier JSON qui inclut les paramètres requis pour le cluster virtuel, comme le montre l'exemple ci-dessous.

```
{
    "name": "virtual_cluster_name",
    "containerProvider": {
        "type": "EKS",
        "id": "eks_cluster_name",
        "info": {
            "eksInfo": {
                "namespace": "namespace_name"
            }
        }
    }
}
```

Exécutez ensuite la commande `create-virtual-cluster` ci-dessous avec le chemin d'accès au fichier JSON.

```
aws emr-containers create-virtual-cluster \
--cli-input-json file://./create-virtual-cluster-request.json
```

> **Note**
>
> Pour valider la création réussie d'un cluster virtuel, consultez l'état des clusters virtuels en exécutant la commande `list-virtual-clusters` ou en accédant à la page Clusters virtuels dans la console Amazon EMR.

Liste des clusters virtuels

Exécutez la commande ci-dessous pour consulter l'état des clusters virtuels.

```
aws emr-containers list-virtual-clusters
```

Description d'un cluster virtuel

Exécutez la commande ci-dessous pour obtenir plus de détails sur un cluster virtuel, tels que l'espace de noms, l'état et la date d'enregistrement. *123456*Remplacez-le par votre ID de cluster virtuel.

```
aws emr-containers describe-virtual-cluster --id 123456
```

Suppression d'un cluster virtuel

Exécutez la commande ci-dessous pour supprimer un cluster virtuel. *123456*Remplacez-le par votre ID de cluster virtuel.

```
aws emr-containers delete-virtual-cluster --id 123456
```

États du cluster virtuel

Le tableau ci-dessous décrit les quatre états possibles d'un cluster virtuel.

State	Description
RUNNING	Le cluster virtuel est en état RUNNING.
TERMINATING	L'arrêt demandé du cluster virtuel est en cours.

State	Description
TERMINATED	L'arrêt demandé est terminé.
ARRESTED	L'arrêt demandé a échoué en raison de l'insuffisance des autorisations.

Didacticiels pour Amazon EMR on EKS

Cette section décrit les cas d'utilisation courants lorsque vous travaillez avec des applications Amazon EMR on EKS. Chaque application est spécialisée et peut être configurée selon des étapes uniques. Ces rubriques fournissent des instructions relatives à l'utilisation de chaque application.

Rubriques

Utilisation de Delta Lake avec Amazon EMR on EKS

Delta Lake est un framework de stockage open source permettant de créer une architecture Lakehouse. Ce qui suit montre comment le configurer pour son utilisation.

Utilisation de Delta Lake avec des applications Amazon EMR on EKS

1. Lorsque vous lancez une exécution de tâche pour soumettre une tâche Spark dans la configuration de l'application, incluez les fichiers JAR de Delta Lake :

```
--job-driver '{"sparkSubmitJobDriver" : {
      "sparkSubmitParameters" : "--jars local:///usr/share/aws/delta/lib/delta-
core.jar,local:///usr/share/aws/delta/lib/delta-storage.jar,local:///usr/share/aws/
delta/lib/delta-storage-s3-dynamodb.jar"}}'
```

> **Note**
>
> Les versions 7.0.0 et supérieures d'Amazon EMR utilisent Delta Lake 3.0, dont le nom est renommé en. `delta-core.jar` `delta-spark.jar` Si vous utilisez les versions 7.0.0 ou supérieures d'Amazon EMR, veillez à utiliser le nom de fichier correct, comme dans l'exemple suivant :
>
> ```
> --jars local:///usr/share/aws/delta/lib/delta-spark.jar
> ```

2. Incluez la configuration supplémentaire de Delta Lake et utilisez AWS Glue Data Catalog comme métastore.

```
--configuration-overrides '{
        "applicationConfiguration": [
        {
          "classification" : "spark-defaults",
          "properties" : {
            "spark.sql.extensions" : "io.delta.sql.DeltaSparkSessionExtension",

 "spark.sql.catalog.spark_catalog":"org.apache.spark.sql.delta.catalog.DeltaCatalog",
"spark.hadoop.hive.metastore.client.factory.class":"com.amazonaws.glue.catalog.metastore.AW
           }
        }]}'
```

Utilisation d'Apache Iceberg avec Amazon EMR on EKS

Le fichier JAR d'exécution d'Iceberg contient les classes Iceberg nécessaires à la prise en charge de l'exécution de Spark. La procédure suivante montre comment démarrer une tâche à l'aide du moteur d'exécution Iceberg Spark.

Utilisation d'Apache Iceberg avec des applications Amazon EMR on EKS

1. Lorsque vous lancez une exécution de tâche pour soumettre une tâche Spark dans la configuration de l'application, incluez le fichier JAR d'exécution Spark d'Iceberg :

```
--job-driver '{"sparkSubmitJobDriver" : {"sparkSubmitParameters" : "--jars
 local:///usr/share/aws/iceberg/lib/iceberg-spark3-runtime.jar"}}'
```

2. Incluez la configuration supplémentaire d'Iceberg :

```
--configuration-overrides '{
    "applicationConfiguration": [
    "classification" : "spark-defaults",
    "properties" : {
        "spark.sql.catalog.dev.warehouse" : "s3://amzn-s3-demo-bucket/EXAMPLE-
PREFIX/ ",
        "spark.sql.extensions ":"
 org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions ",
        "spark.sql.catalog.dev" : "org.apache.iceberg.spark.SparkCatalog",
        "spark.sql.catalog.dev.catalog-impl" :
 "org.apache.iceberg.aws.glue.GlueCatalog",
        "spark.sql.catalog.dev.io-impl": "org.apache.iceberg.aws.s3.S3FileIO"
        }
    ]
}'
```

Pour en savoir plus sur les versions Apache Iceberg d'EMR, consultez l'historique des versions Iceberg.

Configurations de session Spark pour l'intégration de catalogues

Configurations de session Spark pour l'intégration du catalogue Iceberg AWS Glue

Cet exemple montre comment intégrer Iceberg à : AWS Glue crawler

```
spark-sql \
  --conf spark.sql.catalog.rms = org.apache.iceberg.spark.SparkCatalog \
  --conf spark.sql.catalog.rms.type = glue \
  --conf spark.sql.catalog.rms.glue.id = glue RMS catalog ID \
  --conf spark.sql.catalog.rms.glue.account-id = AWS account ID \

  --conf spark.sql.extensions=
    org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions
```

Voici un exemple de requête :

```
SELECT * FROM rms.rmsdb.table1
```

Configurations de session Spark pour l'intégration du catalogue Iceberg REST AWS Glue

Cet exemple montre comment intégrer Iceberg REST à : AWS Glue crawler

```
spark-sql \
  --conf spark.sql.catalog.rms = org.apache.iceberg.spark.SparkCatalog \
  --conf spark.sql.catalog.rms.type = rest \
  --conf spark.sql.catalog.rms.warehouse = glue RMS catalog ID \
  --conf spark.sql.catalog.rms.uri = glue endpoint URI/iceberg \
  --conf spark.sql.catalog.rms.rest.sigv4-enabled = true \
  --conf spark.sql.catalog.rms.rest.signing-name = glue \

  --conf spark.sql.extensions=
    org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions
```

Voici un exemple de requête :

```
SELECT * FROM rms.rmsdb.table1
```

Cette configuration ne fonctionne que pour le stockage géré Redshift. Le FGAC pour Amazon S3 n'est pas pris en charge.

En utilisant PyFlink

Amazon EMR on EKS est compatible avec les versions 6.15.0 et supérieures. PyFlink Si vous disposez déjà d'un PyFlink script, vous pouvez effectuer l'une des opérations suivantes :

- Créez une image personnalisée avec votre PyFlink script inclus.
- Téléchargez votre script vers un emplacement Amazon S3

Si vous n'avez pas encore de script, vous pouvez utiliser l'exemple suivant pour lancer une PyFlink tâche. Cet exemple extrait le script depuis S3. Si vous utilisez une image personnalisée avec votre script déjà inclus dans l'image, vous devez mettre à jour le chemin du script à l'emplacement où vous l'avez enregistré. Si le script se trouve dans un emplacement S3, Amazon EMR on EKS le récupère et le place dans le `/opt/flink/usrlib/` répertoire du conteneur Flink.

```
apiVersion: flink.apache.org/v1beta1
```

```
kind: FlinkDeployment
metadata:
  name: python-example
spec:
  flinkVersion: v1_17
  flinkConfiguration:
    taskmanager.numberOfTaskSlots: "1"
  executionRoleArn: job-execution-role
  emrReleaseLabel: "emr-6.15.0-flink-latest"
  jobManager:
    highAvailabilityEnabled: false
    replicas: 1
    resource:
      memory: "2048m"
      cpu: 1
  taskManager:
    resource:
      memory: "2048m"
      cpu: 1
  job:
    jarURI: s3://S3 bucket with your script/pyflink-script.py
    entryClass: "org.apache.flink.client.python.PythonDriver"
    args: ["-py", "/opt/flink/usrlib/pyflink-script.py"]
    parallelism: 1
    upgradeMode: stateless
```

Utiliser AWS Glue avec Flink

Amazon EMR on EKS avec Apache Flink versions 6.15.0 et supérieures prend en charge l'utilisation du catalogue de données AWS Glue comme magasin de métadonnées pour le streaming et les flux de travail SQL par lots.

Vous devez d'abord créer une base de données AWS Glue nommée `default` qui servira de catalogue Flink SQL. Ce catalogue Flink stocke des métadonnées telles que des bases de données, des tables, des paritions, des vues, des fonctions et d'autres informations nécessaires pour accéder aux données d'autres systèmes externes.

```
aws glue create-database \
    --database-input "{\"Name\":\"default\"}"
```

Pour activer le support de AWS Glue, utilisez une `FlinkDeployment` spécification. Cet exemple de spécification utilise un script Python pour émettre rapidement des instructions Flink SQL afin d'interagir avec le catalogue AWS Glue.

```
apiVersion: flink.apache.org/v1beta1
kind: FlinkDeployment
metadata:
  name: python-example
spec:
  flinkVersion: v1_17
  flinkConfiguration:
    taskmanager.numberOfTaskSlots: "1"
    aws.glue.enabled: "true"
  executionRoleArn: job-execution-role-arn;
  emrReleaseLabel: "emr-6.15.0-flink-latest"
  jobManager:
    highAvailabilityEnabled: false
    replicas: 1
    resource:
      memory: "2048m"
      cpu: 1
  taskManager:
    resource:
      memory: "2048m"
      cpu: 1
  job:
    jarURI: s3://<S3_bucket_with_your_script/pyflink-glue-script.py
    entryClass: "org.apache.flink.client.python.PythonDriver"
    args: ["-py", "/opt/flink/usrlib/pyflink-glue-script.py"]
    parallelism: 1
    upgradeMode: stateless
```

Voici un exemple de ce à quoi pourrait ressembler votre PyFlink script.

```
import logging
import sys
from pyflink.datastream import StreamExecutionEnvironment
from pyflink.table import StreamTableEnvironment

def glue_demo():
    env = StreamExecutionEnvironment.get_execution_environment()
    t_env = StreamTableEnvironment.create(stream_execution_environment=env)
    t_env.execute_sql("""
```

```
          CREATE CATALOG glue_catalog WITH (
          'type' = 'hive',
          'default-database' = 'default',
          'hive-conf-dir' = '/glue/confs/hive/conf',
          'hadoop-conf-dir' = '/glue/confs/hadoop/conf'
          )
                      """)
    t_env.execute_sql("""
          USE CATALOG glue_catalog;
                      """)
    t_env.execute_sql("""
          DROP DATABASE IF EXISTS eks_flink_db CASCADE;
                      """)
    t_env.execute_sql("""
          CREATE DATABASE IF NOT EXISTS eks_flink_db WITH ('hive.database.location-
uri'= 's3a://S3-bucket-to-store-metadata/flink/flink-glue-for-hive/warehouse/');
                      """)
    t_env.execute_sql("""
          USE eks_flink_db;
                  """)
    t_env.execute_sql("""
          CREATE TABLE IF NOT EXISTS eksglueorders (
            order_number BIGINT,
            price        DECIMAL(32,2),
            buyer        RO first_name STRING, last_name STRING,
            order_time   TIMESTAMP(3)
          ) WITH (
            'connector' = 'datagen'
          );
                      """)
    t_env.execute_sql("""
          CREATE TABLE IF NOT EXISTS eksdestglueorders (
            order_number BIGINT,
            price        DECIMAL(32,2),
            buyer        ROW first_name STRING, last_name STRING,
            order_time   TIMESTAMP(3)
          ) WITH (
            'connector' = 'filesystem',
            'path' = 's3://S3-bucket-to-store-metadata/flink/flink-glue-for-hive/
warehouse/eksdestglueorders',
            'format' = 'json'
          );
                  """)
    t_env.execute_sql("""
```

```
          CREATE TABLE IF NOT EXISTS print_table (
            order_number BIGINT,
            price        DECIMAL(32,2),
            buyer        ROW first_name STRING, last_name STRING,
            order_time   TIMESTAMP(3)
          ) WITH (
            'connector' = 'print'
          );
                """)
    t_env.execute_sql("""
          EXECUTE STATEMENT SET
          BEGIN
          INSERT INTO eksdestglueorders SELECT * FROM  eksglueorders LIMIT 10;
          INSERT INTO print_table SELECT * FROM eksdestglueorders;
          END;
            """)


if __name__ == '__main__':
    logging.basicConfig(stream=sys.stdout, level=logging.INFO, format="%(message)s")
    glue_demo()
```

Utilisation d'Apache Hudi avec Apache Flink

Apache Hudi est un framework de gestion de données open source avec des opérations de niveau enregistrement telles que l'insertion, la mise à jour, la modification et la suppression, que vous pouvez utiliser pour simplifier la gestion des données et le développement de pipelines de données. Associé à une gestion efficace des données dans Amazon S3, Hudi vous permet d'ingérer et de mettre à jour des données en temps réel. Hudi gère les métadonnées de toutes les opérations que vous exécutez sur le jeu de données, afin que toutes les actions restent atomiques et cohérentes.

Apache Hudi est disponible sur Amazon EMR sur EKS avec Apache Flink avec Amazon EMR versions 7.2.0 et supérieures. Consultez les étapes suivantes pour savoir comment démarrer et soumettre des tâches Apache Hudi.

Soumettre une offre d'emploi Apache Hudi

Consultez les étapes suivantes pour savoir comment soumettre une tâche Apache Hudi.

1. Créez une base AWS de données Glue nomméedefault.

```
aws glue create-database --database-input "{\"Name\":\"default\"}"
```

2. Suivez l'exemple SQL de l'opérateur Flink Kubernetes pour créer le fichier. `flink-sql-runner.jar`
3. Créez un script SQL Hudi comme suit.

```
CREATE CATALOG hudi_glue_catalog WITH (
'type' = 'hudi',
'mode' = 'hms',
'table.external' = 'true',
'default-database' = 'default',
'hive.conf.dir' = '/glue/confs/hive/conf/',
'catalog.path' = 's3://<hudi-example-bucket>/FLINK_HUDI/warehouse/'
);

USE CATALOG hudi_glue_catalog;
CREATE DATABASE IF NOT EXISTS hudi_db;
use hudi_db;

CREATE TABLE IF NOT EXISTS hudi-flink-example-table(
    uuid VARCHAR(20),
    name VARCHAR(10),
    age INT,
    ts TIMESTAMP(3),
    `partition` VARCHAR(20)
)
PARTITIONED BY (`partition`)
WITH (
  'connector' = 'hudi',
  'path' = 's3://<hudi-example-bucket>/hudi-flink-example-table',
  'hive_sync.enable' = 'true',
  'hive_sync.mode' = 'glue',
  'hive_sync.table' = 'hudi-flink-example-table',
  'hive_sync.db' = 'hudi_db',
  'compaction.delta_commits' = '1',
  'hive_sync.partition_fields' = 'partition',
  'hive_sync.partition_extractor_class' =
 'org.apache.hudi.hive.MultiPartKeysValueExtractor',
  'table.type' = 'COPY_ON_WRITE'
);

EXECUTE STATEMENT SET
```

```
BEGIN

INSERT INTO hudi-flink-example-table VALUES
    ('id1','Alex',23,TIMESTAMP '1970-01-01 00:00:01','par1'),
    ('id2','Stephen',33,TIMESTAMP '1970-01-01 00:00:02','par1'),
    ('id3','Julian',53,TIMESTAMP '1970-01-01 00:00:03','par2'),
    ('id4','Fabian',31,TIMESTAMP '1970-01-01 00:00:04','par2'),
    ('id5','Sophia',18,TIMESTAMP '1970-01-01 00:00:05','par3'),
    ('id6','Emma',20,TIMESTAMP '1970-01-01 00:00:06','par3'),
    ('id7','Bob',44,TIMESTAMP '1970-01-01 00:00:07','par4'),
    ('id8','Han',56,TIMESTAMP '1970-01-01 00:00:08','par4');

END;
```

4. Téléchargez votre script Hudi SQL et le `flink-sql-runner.jar` fichier dans un emplacement S3.
5. Dans votre fichier `FlinkDeployments` YAML, définissez surhudi.enabled. `true`

```
spec:
  flinkConfiguration:
    hudi.enabled: "true"
```

6. Créez un fichier YAML pour exécuter votre configuration. Ce fichier d'exemple est nomméhudi-write.yaml.

```
apiVersion: flink.apache.org/v1beta1
kind: FlinkDeployment
metadata:
  name: hudi-write-example
spec:
  flinkVersion: v1_18
  flinkConfiguration:
    taskmanager.numberOfTaskSlots: "2"
    hudi.enabled: "true"
  executionRoleArn: "<JobExecutionRole>"
  emrReleaseLabel: "emr-7.12.0-flink-latest"
  jobManager:
    highAvailabilityEnabled: false
    replicas: 1
    resource:
      memory: "2048m"
      cpu: 1
  taskManager:
```

```
    resource:
      memory: "2048m"
      cpu: 1
  job:
    jarURI: local:///opt/flink/usrlib/flink-sql-runner.jar
    args: ["/opt/flink/scripts/hudi-write.sql"]
    parallelism: 1
    upgradeMode: stateless
  podTemplate:
    spec:
      initContainers:
        - name: flink-sql-script-download
          args:
            - s3
            - cp
            - s3://<s3_location>/hudi-write.sql
            - /flink-scripts
          image: amazon/aws-cli:latest
          imagePullPolicy: Always
          resources: {}
          terminationMessagePath: /dev/termination-log
          terminationMessagePolicy: File
          volumeMounts:
            - mountPath: /flink-scripts
              name: flink-scripts
        - name: flink-sql-runner-download
          args:
            - s3
            - cp
            - s3://<s3_location>/flink-sql-runner.jar
            - /flink-artifacts
          image: amazon/aws-cli:latest
          imagePullPolicy: Always
          resources: {}
          terminationMessagePath: /dev/termination-log
          terminationMessagePolicy: File
          volumeMounts:
            - mountPath: /flink-artifacts
              name: flink-artifact
      containers:
        - name: flink-main-container
          volumeMounts:
            - mountPath: /opt/flink/scripts
              name: flink-scripts
```

```
            - mountPath: /opt/flink/usrlib
              name: flink-artifact
      volumes:
        - emptyDir: {}
          name: flink-scripts
        - emptyDir: {}
          name: flink-artifact
```

7. Soumettez une tâche Flink Hudi à l'opérateur Flink Kubernetes.

```
kubectl apply -f hudi-write.yaml
```

Utilisation de l'accélérateur RAPIDS pour Apache Spark avec Amazon EMR on EKS

Avec Amazon EMR on EKS, vous pouvez exécuter des tâches pour l'accélérateur Nvidia RAPIDS pour Apache Spark. Ce didacticiel explique comment exécuter des tâches Spark à l'aide de RAPIDS sur des types d'instance EC2 à unité de traitement graphique (GPU). Le didacticiel utilise les versions suivantes :

- Amazon EMR on EKS en version 6.9.0 et ultérieure
- Apache Spark 3.x

Vous pouvez accélérer Spark avec des types d'instances Amazon EC2 à GPU en utilisant le plug-in Nvidia Accélérateur RAPIDS pour Apache Spark. Lorsque vous utilisez ces technologies ensemble, vous accélérez vos pipelines de science des données sans avoir à modifier le code. Cela permet de réduire le temps d'exécution nécessaire au traitement des données et à l'apprentissage des modèles. En accomplissant plus de tâches en moins de temps, vous dépensez moins sur le coût de l'infrastructure.

Avant de commencer, assurez-vous de disposer des ressources ci-dessous.

- Cluster virtuel Amazon EMR on EKS
- Cluster Amazon EKS avec un groupe de nœuds équipés de GPU

Le cluster virtuel Amazon EKS est un descripteur enregistré dans l'espace de noms Kubernetes d'un cluster Amazon EKS, et est géré par Amazon EMR on EKS. Le descripteur permet à Amazon

EMR d'utiliser l'espace de noms Kubernetes comme destination pour exécuter des tâches. Pour plus d'informations sur la configuration d'un cluster virtuel, consultez Configuration d'Amazon EMR on EKS dans ce guide.

Vous devez configurer le cluster virtuel Amazon EKS avec un groupe de nœuds doté d'instances GPU. Vous devez configurer les nœuds avec un plug-in de périphérique Nvidia. Consultez la rubrique Groupes de nœuds gérés pour en savoir plus.

Pour configurer votre cluster Amazon EKS afin d'ajouter des groupes de nœuds équipés de GPU, procédez comme suit :

Ajout de groupes de nœuds équipés de GPU

1. Créez un groupe de nœuds équipés de GPU à l'aide de la commande create-nodegroup suivante. Assurez-vous de remplacer par les paramètres corrects votre cluster Amazon EKS. Utilisez type d'instance compatible avec RAPIDS pour Spark, comme P4, P3, G5 ou G4dn.

```
aws eks create-nodegroup \
 --cluster-name EKS_CLUSTER_NAME \
 --nodegroup-name NODEGROUP_NAME \
 --scaling-config minSize=0,maxSize=5,desiredSize=2 CHOOSE_APPROPRIATELY \
 --ami-type AL2_x86_64_GPU \
 --node-role NODE_ROLE \
 --subnets SUBNETS_SPACE_DELIMITED  \
 --remote-access ec2SshKey= SSH_KEY \
 --instance-types GPU_INSTANCE_TYPE \
 --disk-size DISK_SIZE \
 --region AWS_REGION
```

2. Installez le plug-in pour appareil Nvidia dans votre cluster pour émettre le nombre de GPUs sur chaque nœud de votre cluster et pour exécuter des conteneurs compatibles avec le GPU dans votre cluster. Exécutez le code suivant pour installer le plug-in :

```
kubectl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/v0.9.0/
nvidia-device-plugin.yml
```

3. Pour valider le nombre de nœuds GPUs disponibles sur chaque nœud de votre cluster, exécutez la commande suivante :

```
kubectl get nodes  "-o=custom-
columns=NAME:.metadata.name,GPU:.status.allocatable.nvidia\.com/gpu"
```

Exécution d'une tâche RAPIDS pour Spark

1. Soumettez une tâche RAPIDS pour Spark à votre cluster Amazon EMR on EKS. Le code suivant montre un exemple de commande permettant de démarrer la tâche. La première fois que vous exécutez la tâche, le téléchargement de l'image et sa mise en cache sur le nœud peuvent prendre quelques minutes.

```
aws emr-containers start-job-run \
--virtual-cluster-id VIRTUAL_CLUSTER_ID \
--execution-role-arn JOB_EXECUTION_ROLE \
--release-label emr-6.9.0-spark-rapids-latest \
--job-driver '{"sparkSubmitJobDriver": {"entryPoint": "local:///usr/lib/
spark/examples/jars/spark-examples.jar","entryPointArguments":  ["10000"],
 "sparkSubmitParameters":"--class org.apache.spark.examples.SparkPi "}}' \
---configuration-overrides '{"applicationConfiguration": [{"classification":
 "spark-defaults","properties": {"spark.executor.instances":
 "2","spark.executor.memory": "2G"}}],"monitoringConfiguration":
 {"cloudWatchMonitoringConfiguration": {"logGroupName": "LOG_GROUP
 _NAME"},"s3MonitoringConfiguration": {"logUri": "LOG_GROUP_STREAM"}}}'
```

2. Pour vérifier que l'accélérateur RAPIDS pour Spark est activé, consultez les journaux du pilote Spark. Ces journaux sont stockés dans CloudWatch ou dans l'emplacement S3 que vous spécifiez lorsque vous exécutez la `start-job-run` commande. L'exemple suivant montre généralement à quoi ressemblent les lignes de journal :

```
22/11/15 00:12:44 INFO RapidsPluginUtils: RAPIDS Accelerator build:
 {version=22.08.0-amzn-0, user=release, url=, date=2022-11-03T03:32:45Z, revision=,
 cudf_version=22.08.0, branch=}
22/11/15 00:12:44 INFO RapidsPluginUtils: RAPIDS Accelerator JNI build:
 {version=22.08.0, user=, url=https://github.com/NVIDIA/spark-rapids-jni.git,
 date=2022-08-18T04:14:34Z, revision=a1b23cd_sample, branch=HEAD}
22/11/15 00:12:44 INFO RapidsPluginUtils: cudf build: {version=22.08.0,
 user=, url=https://github.com/rapidsai/cudf.git, date=2022-08-18T04:14:34Z,
 revision=a1b23ce_sample, branch=HEAD}
22/11/15 00:12:44 WARN RapidsPluginUtils: RAPIDS Accelerator 22.08.0-amzn-0 using
 cudf 22.08.0.
22/11/15 00:12:44 WARN RapidsPluginUtils:
 spark.rapids.sql.multiThreadedRead.numThreads is set to 20.
22/11/15 00:12:44 WARN RapidsPluginUtils: RAPIDS Accelerator is enabled, to disable
 GPU support set `spark.rapids.sql.enabled` to false.
```

```
22/11/15 00:12:44 WARN RapidsPluginUtils: spark.rapids.sql.explain is set to
 `NOT_ON_GPU`. Set it to 'NONE' to suppress the diagnostics logging about the query
 placement on the GPU.
```

3. Pour voir les opérations qui seront exécutées sur une GPU, effectuez les étapes suivantes pour activer la journalisation supplémentaire. Notez la configuration « `spark.rapids.sql.explain : ALL` ».

```
aws emr-containers start-job-run \
--virtual-cluster-id VIRTUAL_CLUSTER_ID \
--execution-role-arn JOB_EXECUTION_ROLE \
--release-label emr-6.9.0-spark-rapids-latest \
--job-driver '{"sparkSubmitJobDriver": {"entryPoint": "local:///usr/lib/
spark/examples/jars/spark-examples.jar","entryPointArguments":  ["10000"],
 "sparkSubmitParameters":"--class org.apache.spark.examples.SparkPi "}}' \
---configuration-overrides '{"applicationConfiguration":
 [{"classification": "spark-defaults","properties":
 {"spark.rapids.sql.explain":"ALL","spark.executor.instances":
 "2","spark.executor.memory": "2G"}}],"monitoringConfiguration":
 {"cloudWatchMonitoringConfiguration": {"logGroupName":
 "LOG_GROUP_NAME"},"s3MonitoringConfiguration": {"logUri": "LOG_GROUP_STREAM"}}}'
```

La commande précédente est un exemple de tâche utilisant la GPU. Son résultat ressemblerait à l'exemple ci-dessous. Reportez-vous à cette clé pour mieux comprendre le résultat :

- * : marque une opération fonctionnant sur une GPU
- ! : marque une opération qui ne peut pas être exécutée sur une GPU
- @ : marque une opération fonctionnant sur un GPU, mais qui ne sera pas exécutée parce qu'elle fait partie d'un plan qui ne peut pas être exécuté sur une GPU

```
22/11/15 01:22:58 INFO GpuOverrides: Plan conversion to the GPU took 118.64 ms
22/11/15 01:22:58 INFO GpuOverrides: Plan conversion to the GPU took 4.20 ms
22/11/15 01:22:58 INFO GpuOverrides: GPU plan transition optimization took 8.37 ms
22/11/15 01:22:59 WARN GpuOverrides:
   *Exec <ProjectExec> will run on GPU
     *Expression <Alias> substring(cast(date#149 as string), 0, 7) AS month#310
will run on GPU
       *Expression <Substring> substring(cast(date#149 as string), 0, 7) will run
on GPU
```

```
          *Expression <Cast> cast(date#149 as string) will run on GPU
      *Exec <SortExec> will run on GPU
        *Expression <SortOrder> date#149 ASC NULLS FIRST will run on GPU
        *Exec <ShuffleExchangeExec> will run on GPU
          *Partitioning <RangePartitioning> will run on GPU
            *Expression <SortOrder> date#149 ASC NULLS FIRST will run on GPU
          *Exec <UnionExec> will run on GPU
            !Exec <ProjectExec> cannot run on GPU because not all expressions can
 be replaced
              @Expression <AttributeReference> customerID#0 could run on GPU
              @Expression <Alias> Charge AS kind#126 could run on GPU
                @Expression <Literal> Charge could run on GPU
              @Expression <AttributeReference> value#129 could run on GPU
              @Expression <Alias> add_months(2022-11-15, cast(-(cast(_we0#142 as
 bigint) + last_month#128L) as int)) AS date#149 could run on GPU
                ! <AddMonths> add_months(2022-11-15, cast(-
(cast(_we0#142 as bigint) + last_month#128L) as int)) cannot run
 on GPU because GPU does not currently support the operator class
 org.apache.spark.sql.catalyst.expressions.AddMonths
                  @Expression <Literal> 2022-11-15 could run on GPU
                  @Expression <Cast> cast(-(cast(_we0#142 as bigint) +
 last_month#128L) as int) could run on GPU
                    @Expression <UnaryMinus> -(cast(_we0#142 as bigint) +
 last_month#128L) could run on GPU
                      @Expression <Add> (cast(_we0#142 as bigint) +
 last_month#128L) could run on GPU
                        @Expression <Cast> cast(_we0#142 as bigint) could run on
 GPU
                          @Expression <AttributeReference> _we0#142 could run on
 GPU
                        @Expression <AttributeReference> last_month#128L could run
 on GPU
```

Utilisation de l'intégration Amazon Redshift pour Apache Spark sur Amazon EMR on EKS

À partir de la version 6.9.0 d'Amazon EMR, chaque image de version inclut un connecteur entre Apache Spark et Amazon Redshift. Vous pouvez ainsi utiliser Spark sur Amazon EMR on EKS pour traiter des données stockées dans Amazon Redshift. L'intégration est basée sur le connecteur open-

source `spark-redshift`. Pour Amazon EMR on EKS, l'intégration Amazon Redshift pour Apache Spark est incluse en tant qu'intégration native.

Rubriques

- Lancement d'une application Spark à l'aide de l'intégration Amazon Redshift pour Apache Spark
- Authentification avec l'intégration Amazon Redshift pour Apache Spark
- Lecture et écriture à partir de et vers Amazon Redshift
- Considérations et limites relatives à l'utilisation du connecteur Spark

Lancement d'une application Spark à l'aide de l'intégration Amazon Redshift pour Apache Spark

Pour utiliser l'intégration, vous devez transmettre les dépendances Spark Redshift requises à votre tâche Spark. Vous devez utiliser `--jars` pour inclure les bibliothèques liées au connecteur Redshift. Pour connaître les autres emplacements de fichiers pris en charge par l'option `--jars`, consultez la rubrique Gestion avancée des dépendances de la documentation Apache Spark.

- `spark-redshift.jar`
- `spark-avro.jar`
- `RedshiftJDBC.jar`
- `minimal-json.jar`

Pour lancer une application Spark avec l'intégration Amazon Redshift pour Apache Spark sur Amazon EMR on EKS en version 6.9.0 ou ultérieure, utilisez la commande de l'exemple ci-dessous. Notez que les chemins répertoriés avec l'option `--conf spark.jars` sont les chemins par défaut des fichiers JAR.

```
aws emr-containers start-job-run \

--virtual-cluster-id cluster_id \
--execution-role-arn arn \
--release-label emr-6.9.0-latest\
--job-driver '{
    "sparkSubmitJobDriver": {
        "entryPoint": "s3://script_path",
            "sparkSubmitParameters":
            "--conf spark.kubernetes.file.upload.path=s3://upload_path
```

```
             --conf spark.jars=
                /usr/share/aws/redshift/jdbc/RedshiftJDBC.jar,
                /usr/share/aws/redshift/spark-redshift/lib/spark-redshift.jar,
                /usr/share/aws/redshift/spark-redshift/lib/spark-avro.jar,
                /usr/share/aws/redshift/spark-redshift/lib/minimal-json.jar"
                            }
          }'
```

Authentification avec l'intégration Amazon Redshift pour Apache Spark

Les sections suivantes présentent les options d'authentification avec Amazon Redshift lors de l'intégration à Apache Spark. Les sections montrent comment récupérer les informations de connexion ainsi que les détails concernant l'utilisation du pilote JDBC avec l'authentification IAM.

AWS Secrets Manager À utiliser pour récupérer les informations d'identification et se connecter à Amazon Redshift

Vous pouvez stocker les informations d'identification dans Secrets Manager pour vous authentifier en toute sécurité dans Amazon Redshift. Vous pouvez demander à votre tâche Spark d'appeler l'API `GetSecretValue` pour récupérer les informations d'identification :

```
from pyspark.sql import SQLContextimport boto3

sc = # existing SparkContext
sql_context = SQLContext(sc)

secretsmanager_client = boto3.client('secretsmanager',
 region_name=os.getenv('AWS_REGION'))
secret_manager_response = secretsmanager_client.get_secret_value(
    SecretId='string',
    VersionId='string',
    VersionStage='string'
)
username = # get username from secret_manager_response
password = # get password from secret_manager_response
url = "jdbc:redshift://redshifthost:5439/database?user=" + username + "&password="
 + password

# Access to Redshift cluster using Spark
```

Utilisation de l'authentification basée sur IAM avec le rôle d'exécution des tâches Amazon EMR on EKS

À partir de la version 6.9.0 d'Amazon EMR on EKS, le pilote JDBC Amazon Redshift en version 2.1 ou supérieure est intégré à l'environnement. Avec le pilote JDBC en version 2.1 et supérieure, vous pouvez spécifier l'URL JDBC et ne pas inclure le nom d'utilisateur et le mot de passe bruts. Au lieu de cela, vous pouvez indiquer un schéma `jdbc:redshift:iam://`. Cela commande au pilote JDBC d'utiliser votre rôle d'exécution de tâche Amazon EMR on EKS pour récupérer automatiquement les informations d'identification.

Pour plus d'informations, consultez la rubrique Configuration d'une connexion JDBC ou ODBC pour utiliser les informations d'identification IAM dans le Guide de gestion Amazon Redshift.

L'exemple d'URL suivant utilise un schéma `jdbc:redshift:iam://`.

```
jdbc:redshift:iam://examplecluster.abc123xyz789.us-west-2.redshift.amazonaws.com:5439/
dev
```

Les autorisations suivantes sont requises pour votre rôle d'exécution des tâches lorsqu'il remplit les conditions prévues.

Autorisations	Conditions requises pour le rôle d'exécution des tâches
`redshift:GetClusterCredentials`	Requis pour que le pilote JDBC récupère les informations d'identification à partir d'Amazon Redshift
`redshift:DescribeCluster`	Requis si vous indiquez le cluster Amazon Redshift et la Région AWS dans l'URL JDBC au lieu du point de terminaison
`redshift-serverless:GetCredentials`	Requis pour que le pilote JDBC récupère les informations d'identification à partir d'Amazon Redshift sans serveur
`redshift-serverless:GetWorkgroup`	Requis si vous utilisez Amazon Redshift sans serveur et que vous indiquez l'URL pour le nom et la région du groupe de travail

Votre politique de rôle d'exécution des tâches doit disposer des autorisations ci-dessous.

```
{
```

```
            "Effect": "Allow",
            "Action": [
                "redshift:GetClusterCredentials",
                "redshift:DescribeCluster",
                "redshift-serverless:GetCredentials",
                "redshift-serverless:GetWorkgroup"
            ],
            "Resource": [

 "arn:aws:redshift:AWS_REGION:ACCOUNT_ID:dbname:CLUSTER_NAME/DATABASE_NAME",
                "arn:aws:redshift:AWS_REGION:ACCOUNT_ID:dbuser:DATABASE_NAME/USER_NAME"
            ]
        }
```

Authentification dans Amazon Redshift à l'aide d'un pilote JDBC

Définition du nom d'utilisateur et du mot de passe dans l'URL JDBC

Pour authentifier une tâche Spark dans un cluster Amazon Redshift, vous pouvez indiquer le nom et le mot de passe de la base de données Amazon Redshift dans l'URL JDBC.

> **Note**
>
> Si vous transmettez les informations d'identification de la base de données dans l'URL, toute personne ayant accès à l'URL peut également accéder aux informations d'identification. Cette méthode n'est généralement pas recommandée, car elle n'est pas une option sécurisée.

Si la sécurité n'est pas une préoccupation pour votre application, vous pouvez utiliser le format suivant pour définir le nom d'utilisateur et le mot de passe dans l'URL JDBC :

```
jdbc:redshift://redshifthost:5439/database?user=username&password=password
```

Lecture et écriture à partir de et vers Amazon Redshift

Les exemples de code suivants permettent de lire et PySpark d'écrire des exemples de données depuis et vers une base de données Amazon Redshift à l'aide d'une API de source de données et de SparkSQL.

Data source API

PySpark À utiliser pour lire et écrire des exemples de données depuis et vers une base de données Amazon Redshift à l'aide d'une API de source de données.

```
import boto3
from pyspark.sql import SQLContext

sc = # existing SparkContext
sql_context = SQLContext(sc)

url = "jdbc:redshift:iam://redshifthost:5439/database"
aws_iam_role_arn = "arn:aws:iam::accountID:role/roleName"

df = sql_context.read \
    .format("io.github.spark_redshift_community.spark.redshift") \
    .option("url", url) \
    .option("dbtable", "tableName") \
    .option("tempdir", "s3://path/for/temp/data") \
    .option("aws_iam_role", "aws_iam_role_arn") \
    .load()

df.write \
    .format("io.github.spark_redshift_community.spark.redshift") \
    .option("url", url) \
    .option("dbtable", "tableName_copy") \
    .option("tempdir", "s3://path/for/temp/data") \
    .option("aws_iam_role", "aws_iam_role_arn") \
    .mode("error") \
    .save()
```

SparkSQL

Permet PySpark de lire et d'écrire des exemples de données depuis et vers une base de données Amazon Redshift à l'aide de SparkSQL.

```
import boto3
import json
import sys
import os
from pyspark.sql import SparkSession

spark = SparkSession \
```

```
    .builder \
    .enableHiveSupport() \
    .getOrCreate()

url = "jdbc:redshift:iam://redshifthost:5439/database"
aws_iam_role_arn = "arn:aws:iam::accountID:role/roleName"

bucket = "s3://path/for/temp/data"
tableName = "tableName" # Redshift table name

s = f"""CREATE TABLE IF NOT EXISTS {tableName} (country string, data string)
    USING io.github.spark_redshift_community.spark.redshift
    OPTIONS (dbtable '{tableName}', tempdir '{bucket}', url '{url}', aws_iam_role
 '{aws_iam_role_arn}' ); """

spark.sql(s)

columns = ["country" ,"data"]
data = [("test-country","test-data")]
df = spark.sparkContext.parallelize(data).toDF(columns)

# Insert data into table
df.write.insertInto(tableName, overwrite=False)
df = spark.sql(f"SELECT * FROM {tableName}")
df.show()
```

Considérations et limites relatives à l'utilisation du connecteur Spark

Le connecteur Spark permet de gérer les informations d'identification de différentes manières, de configurer la sécurité et de se connecter à d'autres AWS services. Familiarisez-vous avec les recommandations de cette liste afin de configurer une connexion fonctionnelle et résiliente.

- Nous vous recommandons d'activer SSL pour la connexion JDBC entre Spark sur Amazon EMR et Amazon Redshift.
- À titre de bonne pratique, nous vous recommandons de gérer les informations d'identification du cluster Amazon Redshift dans AWS Secrets Manager . Voir Utiliser AWS Secrets Manager pour récupérer les informations d'identification pour se connecter à Amazon Redshift pour un exemple.
- Nous vous recommandons de transmettre un rôle IAM à l'aide du paramètre `aws_iam_role` pour le paramètre d'authentification Amazon Redshift.
- Le paramètre `tempformat` ne prend actuellement pas en charge le format Parquet.

- L'URI `tempdir` renvoie à un emplacement Amazon S3. Ce répertoire temporaire n'est pas nettoyé automatiquement et peut donc entraîner des coûts supplémentaires.
- Tenez compte des recommandations suivantes pour Amazon Redshift :
  - Nous vous recommandons de bloquer l'accès public au cluster Amazon Redshift.
  - Nous vous recommandons d'activer la journalisation des audits d'Amazon Redshift.
  - Nous vous recommandons d'activer le chiffrement au repos d'Amazon Redshift.
- Tenez compte des recommandations suivantes pour Amazon S3 :
  - Nous vous recommandons de bloquer l'accès public aux compartiments Amazon S3.
  - Nous vous recommandons d'utiliser le chiffrement côté serveur sur Amazon S3 pour chiffrer les compartiments Amazon S3 utilisés.
  - Nous vous recommandons d'utiliser les politiques de cycle de vie d'Amazon S3 pour définir les règles de conservation du compartiment Amazon S3.
  - Amazon EMR vérifie toujours le code importé à partir d'une source ouverte dans l'image. Pour des raisons de sécurité, nous ne prenons pas en charge le codage des clés d' AWS accès dans l'`tempdir`URI comme méthode d'authentification entre Spark et Amazon S3.

Pour plus d'informations sur l'utilisation du connecteur et les paramètres qu'il prend en charge, consultez les ressources suivantes :

- Intégration d'Amazon Redshift pour Apache Spark dans le Guide de gestion Amazon Redshift
- Le référentiel communautaire `spark-redshift` sur Github

Utilisation de Volcano comme planificateur personnalisé pour Apache Spark sur Amazon EMR on EKS

Avec Amazon EMR on EKS, vous avez la possibilité d'utiliser l'opérateur Spark ou la commande spark-submit pour lancer des tâches Spark en utilisant des planificateurs personnalisés Kubernetes. Ce didacticiel explique comment exécuter des tâches Spark avec un planificateur Volcano sur une file d'attente personnalisée.

Présentation de

Volcano peut aider à gérer la planification Spark grâce à des fonctions avancées telles que la planification des files d'attente, la planification du partage équitable et la réservation de ressources.

Pour plus d'informations sur les avantages de Volcano, consultez l'article Pourquoi Spark choisit Volcano comme planificateur de lots intégré sur Kubernetes sur le blog CNCF de Linux Foundation.

Installation et configuration de Volcano

1. Choisissez l'une des commandes kubectl ci-dessous pour installer Volcano, en fonction de vos besoins architecturaux :

```
# x86_64
kubectl apply -f https://raw.githubusercontent.com/volcano-sh/volcano/v1.5.1/
installer/volcano-development.yaml
# arm64:
kubectl apply -f https://raw.githubusercontent.com/volcano-sh/volcano/v1.5.1/
installer/volcano-development-arm64.yaml
```

2. Préparez un exemple de file d'attente Volcano. Une file d'attente est un ensemble de PodGroups. La file d'attente utilise le principe FIFO et constitue la base de la répartition des ressources.

```
cat << EOF > volcanoQ.yaml
apiVersion: scheduling.volcano.sh/v1beta1
kind: Queue
metadata:
  name: sparkqueue
spec:
  weight: 4
  reclaimable: false
  capability:
    cpu: 10
    memory: 20Gi
EOF

kubectl apply -f volcanoQ.yaml
```

3. Téléchargez un exemple de PodGroup manifeste sur Amazon S3. PodGroup est un groupe de capsules fortement associées. Vous utilisez généralement un PodGroup pour la planification par lots. Soumettez l'exemple suivant PodGroup à la file d'attente que vous avez définie à l'étape précédente.

```
cat << EOF > podGroup.yaml
apiVersion: scheduling.volcano.sh/v1beta1
```

```
kind: PodGroup
spec:
  # Set minMember to 1 to make a driver pod
  minMember: 1
  # Specify minResources to support resource reservation.
  # Consider the driver pod resource and executors pod resource.
  # The available resources should meet the minimum requirements of the Spark job
  # to avoid a situation where drivers are scheduled, but they can't schedule
  # sufficient executors to progress.
  minResources:
    cpu: "1"
    memory: "1Gi"
  # Specify the queue. This defines the resource queue that the job should be
 submitted to.
  queue: sparkqueue
EOF

aws s3 mv podGroup.yaml s3://bucket-name
```

Exécution d'une application Spark avec le planificateur Volcano à l'aide de l'opérateur Spark

1. Si vous ne l'avez pas encore fait, suivez les étapes indiquées dans les sections ci-dessous pour vous préparer :

   a. Installation et configuration de Volcano

   b. Configuration de l'opérateur Spark pour Amazon EMR on EKS

   c. Installation de l'opérateur Spark

   Incluez les arguments suivants lorsque vous exécutez la commande `helm install spark-operator-demo` :

   ```
   --set batchScheduler.enable=true
   --set webhook.enable=true
   ```

2. Créez un fichier de définition `SparkApplication` `spark-pi.yaml` avec `batchScheduler` configuré.

   ```
   apiVersion: "sparkoperator.k8s.io/v1beta2"
   kind: SparkApplication
   ```

```
metadata:
  name: spark-pi
  namespace: spark-operator
spec:
  type: Scala
  mode: cluster
  image: "895885662937.dkr.ecr.us-west-2.amazonaws.com/spark/emr-6.10.0:latest"
  imagePullPolicy: Always
  mainClass: org.apache.spark.examples.SparkPi
  mainApplicationFile: "local:///usr/lib/spark/examples/jars/spark-examples.jar"
  sparkVersion: "3.3.1"
  batchScheduler: "volcano"   #Note: You must specify the batch scheduler name as
 'volcano'
  restartPolicy:
    type: Never
  volumes:
    - name: "test-volume"
      hostPath:
        path: "/tmp"
        type: Directory
  driver:
    cores: 1
    coreLimit: "1200m"
    memory: "512m"
    labels:
      version: 3.3.1
    serviceAccount: emr-containers-sa-spark
    volumeMounts:
      - name: "test-volume"
        mountPath: "/tmp"
  executor:
    cores: 1
    instances: 1
    memory: "512m"
    labels:
      version: 3.3.1
    volumeMounts:
      - name: "test-volume"
        mountPath: "/tmp"
```

3. Soumettez l'application Spark à l'aide de la commande suivante. Cela crée également un objet `SparkApplication` appelé `spark-pi` :

```
kubectl apply -f spark-pi.yaml
```

4. Vérifiez les événements de l'objet `SparkApplication` à l'aide de la commande suivante :

```
kubectl describe pods spark-pi-driver --namespace spark-operator
```

Le premier événement du pod montrera que Volcano a programmé les pods :

```
Type    Reason     Age   From               Message
----    ------     ----  ----               -------
Normal  Scheduled  23s   volcano            Successfully assigned default/spark-
pi-driver to integration-worker2
```

Exécution d'une application Spark avec le planificateur Volcano à l'aide de **`spark-submit`**

1. Effectuez d'abord les étapes décrites dans la section Configuration de spark-submit pour Amazon EMR on EKS. Vous devez créer votre distribution `spark-submit` en y incluant la prise en charge de Volcano. Pour plus d'informations, consultez la section Création de la rubrique Utilisation de Volcano comme planificateur personnalisé pour Spark sur Kubernetes dans la documentation Apache Spark.
2. Définissez les valeurs des variables d'environnement suivantes :

```
export SPARK_HOME=spark-home
export MASTER_URL=k8s://Amazon-EKS-cluster-endpoint
```

3. Soumettez l'application Spark à l'aide de la commande suivante :

```
$SPARK_HOME/bin/spark-submit \
 --class org.apache.spark.examples.SparkPi \
 --master $MASTER_URL \
 --conf spark.kubernetes.container.image=895885662937.dkr.ecr.us-
west-2.amazonaws.com/spark/emr-6.10.0:latest \
 --conf spark.kubernetes.authenticate.driver.serviceAccountName=spark \
 --deploy-mode cluster \
 --conf spark.kubernetes.namespace=spark-operator \
 --conf spark.kubernetes.scheduler.name=volcano \
```

```
 --conf spark.kubernetes.scheduler.volcano.podGroupTemplateFile=/path/to/podgroup-
template.yaml \
 --conf
 spark.kubernetes.driver.pod.featureSteps=org.apache.spark.deploy.k8s.features.VolcanoFeatu
 \
 --conf
 spark.kubernetes.executor.pod.featureSteps=org.apache.spark.deploy.k8s.features.VolcanoFea
 \
 local:///usr/lib/spark/examples/jars/spark-examples.jar 20
```

4. Vérifiez les événements de l'objet `SparkApplication` à l'aide de la commande suivante :

```
kubectl describe pod spark-pi --namespace spark-operator
```

Le premier événement du pod montrera que Volcano a programmé les pods :

```
Type    Reason     Age   From               Message
----    ------     ----  ----               -------
Normal  Scheduled  23s   volcano            Successfully assigned default/spark-
pi-driver to integration-worker2
```

Utilisation en YuniKorn tant que planificateur personnalisé pour Apache Spark sur Amazon EMR sur EKS

Avec Amazon EMR on EKS, vous avez la possibilité d'utiliser l'opérateur Spark ou la commande spark-submit pour lancer des tâches Spark en utilisant des planificateurs personnalisés Kubernetes. Ce didacticiel explique comment exécuter des tâches Spark à l'aide d'un YuniKorn planificateur sur une file d'attente personnalisée et comment planifier par groupes.

Présentation de

Apache YuniKorn peut vous aider à gérer la planification Spark grâce à une planification adaptée aux applications afin que vous puissiez contrôler avec précision les quotas et les priorités des ressources. Avec la planification en groupe, YuniKorn planifie une application uniquement lorsque la demande de ressources minimale pour l'application peut être satisfaite. Pour plus d'informations, consultez la section Qu'est-ce que la planification par groupes sur le site de YuniKorn documentation d'Apache.

Créez votre cluster et préparez-vous pour YuniKorn

Suivez les étapes ci-dessous pour déployer un cluster Amazon EKS. Vous pouvez modifier les zones Région AWS (`region`) et les zones de disponibilité (`availabilityZones`).

1. Définissez le cluster Amazon EKS :

```
cat <<EOF >eks-cluster.yaml
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: emr-eks-cluster
  region: eu-west-1

vpc:
  clusterEndpoints:
    publicAccess: true
    privateAccess: true

iam:
  withOIDC: true

nodeGroups:
  - name: spark-jobs
    labels: { app: spark }
    instanceType: m5.xlarge
    desiredCapacity: 2
    minSize: 2
    maxSize: 3
    availabilityZones: ["eu-west-1a"]
EOF
```

2. Créez le cluster :

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

3. Créez l'espace de noms `spark-job` dans lequel vous exécuterez la tâche Spark :

```
kubectl create namespace spark-job
```

4. Créez ensuite un rôle Kubernetes et une liaison de rôle. Cela est requis pour le compte de service utilisé par l'exécution de la tâche Spark.

   a. Définissez le compte de service, le rôle et la liaison de rôle pour les tâches Spark.

```
cat <<EOF >emr-job-execution-rbac.yaml
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: spark-sa
  namespace: spark-job
automountServiceAccountToken: false
---
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: spark-role
  namespace: spark-job
rules:
  - apiGroups: ["", "batch","extensions"]
    resources: ["configmaps","serviceaccounts","events","pods","pods/
exec","pods/log","pods/
portforward","secrets","services","persistentvolumeclaims"]
    verbs: ["create","delete","get","list","patch","update","watch"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: spark-sa-rb
  namespace: spark-job
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: spark-role
subjects:
  - kind: ServiceAccount
    name: spark-sa
    namespace: spark-job
EOF
```

   b. Appliquez la définition du rôle Kubernetes et de la liaison de rôle à l'aide de la commande suivante :

```
kubectl apply -f emr-job-execution-rbac.yaml
```

Installation et configuration YuniKorn

1. Utilisez la commande kubectl suivante pour créer un espace de noms `yunikorn` afin de déployer le planificateur YuniKorn :

   ```
   kubectl create namespace yunikorn
   ```

2. Pour installer le planificateur, exécutez les commandes Helm suivantes :

   ```
   helm repo add yunikorn https://apache.github.io/yunikorn-release
   ```

   ```
   helm repo update
   ```

   ```
   helm install yunikorn yunikorn/yunikorn --namespace yunikorn
   ```

Exécuter une application Spark avec un YuniKorn planificateur avec l'opérateur Spark

1. Si vous ne l'avez pas encore fait, suivez les étapes indiquées dans les sections ci-dessous pour vous préparer :

   a. Créez votre cluster et préparez-vous pour YuniKorn

   b. Installation et configuration YuniKorn

   c. Configuration de l'opérateur Spark pour Amazon EMR on EKS

   d. Installation de l'opérateur Spark

   Incluez les arguments suivants lorsque vous exécutez la commande `helm install spark-operator-demo` :

   ```
   --set batchScheduler.enable=true
   --set webhook.enable=true
   ```

2. Créez un fichier de définition `SparkApplication` `spark-pi.yaml`.

   Pour l'utiliser YuniKorn comme planificateur pour vos tâches, vous devez ajouter certaines annotations et étiquettes à la définition de votre application. Les annotations et les étiquettes indiquent la file d'attente pour votre tâche et la stratégie de planification que vous souhaitez utiliser.

   Dans l'exemple ci-dessous, l'annotation `schedulingPolicyParameters` configure la planification groupée pour l'application. L'exemple crée ensuite des groupes de tâches pour indiquer la capacité minimale qui doit être disponible avant de programmer les pods pour commencer l'exécution de la tâche. Enfin, il indique dans la définition du groupe de tâches qu'il faut utiliser des groupes de nœuds avec l'étiquette `"app": "spark"`, comme défini dans la section Créez votre cluster et préparez-vous pour YuniKorn.

```
apiVersion: "sparkoperator.k8s.io/v1beta2"
kind: SparkApplication
metadata:
  name: spark-pi
  namespace: spark-job
spec:
  type: Scala
  mode: cluster
  image: "895885662937.dkr.ecr.us-west-2.amazonaws.com/spark/emr-6.10.0:latest"
  imagePullPolicy: Always
  mainClass: org.apache.spark.examples.SparkPi
  mainApplicationFile: "local:///usr/lib/spark/examples/jars/spark-examples.jar"
  sparkVersion: "3.3.1"
  restartPolicy:
    type: Never
  volumes:
    - name: "test-volume"
      hostPath:
        path: "/tmp"
        type: Directory
  driver:
    cores: 1
    coreLimit: "1200m"
    memory: "512m"
    labels:
      version: 3.3.1
    annotations:
```

```
    yunikorn.apache.org/schedulingPolicyParameters: "placeholderTimeoutSeconds=30
gangSchedulingStyle=Hard"
    yunikorn.apache.org/task-group-name: "spark-driver"
    yunikorn.apache.org/task-groups: |-
      [{
          "name": "spark-driver",
          "minMember": 1,
          "minResource": {
            "cpu": "1200m",
            "memory": "1Gi"
          },
          "nodeSelector": {
            "app": "spark"
          }
        },
        {
          "name": "spark-executor",
          "minMember": 1,
          "minResource": {
            "cpu": "1200m",
            "memory": "1Gi"
          },
          "nodeSelector": {
            "app": "spark"
          }
      }]
  serviceAccount: spark-sa
  volumeMounts:
    - name: "test-volume"
      mountPath: "/tmp"
executor:
  cores: 1
  instances: 1
  memory: "512m"
  labels:
    version: 3.3.1
  annotations:
    yunikorn.apache.org/task-group-name: "spark-executor"
  volumeMounts:
    - name: "test-volume"
      mountPath: "/tmp"
```

3. Soumettez l'application Spark à l'aide de la commande suivante. Cela crée également un objet `SparkApplication` appelé `spark-pi` :

```
kubectl apply -f spark-pi.yaml
```

4. Vérifiez les événements de l'objet `SparkApplication` à l'aide de la commande suivante :

```
kubectl describe sparkapplication spark-pi --namespace spark-job
```

Le premier événement dédié aux pods indiquera qui YuniKorn a programmé les pods :

```
Type    Reason            Age   From                          Message
----    ------            ----  ----                          -------
Normal Scheduling        3m12s yunikorn   spark-operator/org-apache-spark-examples-
sparkpi-2a777a88b98b8a95-driver is queued and waiting for allocation
Normal GangScheduling    3m12s yunikorn   Pod belongs to the taskGroup spark-
driver, it will be scheduled as a gang member
Normal Scheduled         3m10s yunikorn   Successfully assigned spark
Normal PodBindSuccessful 3m10s yunikorn   Pod spark-operator/
Normal TaskCompleted     2m3s  yunikorn   Task spark-operator/
Normal Pulling           3m10s kubelet    Pulling
```

Exécutez une application Spark avec un YuniKorn planificateur avec **`spark-submit`**

1. Effectuez d'abord les étapes décrites dans la section Configuration de spark-submit pour Amazon EMR on EKS.
2. Définissez les valeurs des variables d'environnement suivantes :

```
export SPARK_HOME=spark-home
export MASTER_URL=k8s://Amazon-EKS-cluster-endpoint
```

3. Soumettez l'application Spark à l'aide de la commande suivante :

Dans l'exemple ci-dessous, l'annotation `schedulingPolicyParameters` configure la planification groupée pour l'application. L'exemple crée ensuite des groupes de tâches pour indiquer la capacité minimale qui doit être disponible avant de programmer les pods pour commencer l'exécution de la tâche. Enfin, il indique dans la définition du groupe de tâches qu'il faut utiliser des groupes de nœuds avec l'étiquette `"app": "spark"`, comme défini dans la section Créez votre cluster et préparez-vous pour YuniKorn.

```
$SPARK_HOME/bin/spark-submit \
 --class org.apache.spark.examples.SparkPi \
 --master $MASTER_URL \
 --conf spark.kubernetes.container.image=895885662937.dkr.ecr.us-
west-2.amazonaws.com/spark/emr-6.10.0:latest \
 --conf spark.kubernetes.authenticate.driver.serviceAccountName=spark-sa \
 --deploy-mode cluster \
 --conf spark.kubernetes.namespace=spark-job \
 --conf spark.kubernetes.scheduler.name=yunikorn \
 --conf spark.kubernetes.driver.annotation.yunikorn.apache.org/
schedulingPolicyParameters="placeholderTimeoutSeconds=30 gangSchedulingStyle=Hard"
 \
 --conf spark.kubernetes.driver.annotation.yunikorn.apache.org/task-group-
name="spark-driver" \
 --conf spark.kubernetes.executor.annotation.yunikorn.apache.org/task-group-
name="spark-executor" \
 --conf spark.kubernetes.driver.annotation.yunikorn.apache.org/task-groups='[{
           "name": "spark-driver",
           "minMember": 1,
           "minResource": {
             "cpu": "1200m",
             "memory": "1Gi"
           },
           "nodeSelector": {
             "app": "spark"
           }
         },
         {
           "name": "spark-executor",
           "minMember": 1,
           "minResource": {
             "cpu": "1200m",
             "memory": "1Gi"
           },
           "nodeSelector": {
             "app": "spark"
           }
       }]' \
 local:///usr/lib/spark/examples/jars/spark-examples.jar 20
```

4. Vérifiez les événements de l'objet `SparkApplication` à l'aide de la commande suivante :

```
kubectl describe pod spark-driver-pod --namespace spark-job
```

Le premier événement dédié aux pods indiquera qui YuniKorn a programmé les pods :

```
Type    Reason           Age   From                         Message
----    ------           ----  ----                         -------
Normal Scheduling        3m12s yunikorn   spark-operator/org-apache-spark-examples-
sparkpi-2a777a88b98b8a95-driver is queued and waiting for allocation
Normal GangScheduling    3m12s yunikorn   Pod belongs to the taskGroup spark-
driver, it will be scheduled as a gang member
Normal Scheduled         3m10s yunikorn   Successfully assigned spark
Normal PodBindSuccessful 3m10s yunikorn   Pod spark-operator/
Normal TaskCompleted     2m3s  yunikorn   Task spark-operator/
Normal Pulling           3m10s kubelet    Pulling
```

Sécurité dans Amazon EMR on EKS

La sécurité du cloud AWS est la priorité absolue. En tant que AWS client, vous bénéficiez de centres de données et d'architectures 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 ceci comme la sécurité du cloud et la sécurité dans le cloud :

- Sécurité du cloud : AWS est chargée de protéger l'infrastructure qui exécute les AWS services dans le AWS cloud. AWS vous fournit également des services que vous pouvez utiliser en toute sécurité. Des auditeurs tiers testent et vérifient régulièrement l'efficacité de notre sécurité dans le cadre des programmes de AWS conformité Programmes de de conformité. Pour en savoir plus sur les programmes de conformité qui s'appliquent à Amazon EMR, consultez la section AWS Services concernés par programme de conformitéAWS .
- Sécurité dans le cloud — Votre responsabilité est déterminée par le AWS service que vous utilisez. Vous êtes également responsable d'autres facteurs, y compris de la sensibilité de vos données, des exigences de votre entreprise, ainsi que de la législation et de la réglementation applicables.

Cette documentation vous aide à comprendre comment appliquer le modèle de responsabilité partagée lors de l'utilisation d'Amazon EMR on EKS. Les rubriques suivantes vous montrent comment configurer Amazon EMR on 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 EMR on EKS.

Rubriques

- Sécurité de l'infrastructure dans Amazon EMR on EKS
- Analyse de la configuration et des vulnérabilités
- Connexion à Amazon EMR on EKS à l'aide du point de terminaison d'un VPC d'interface
- Configuration de l'accès intercompte pour Amazon EMR on EKS

Bonnes pratiques de sécurité pour Amazon EMR on EKS

Amazon EMR on EKS fournit différentes fonctions de sécurité à prendre en compte lorsque vous développez et implémentez vos propres stratégies de sécurité. Les bonnes pratiques suivantes doivent être considérées comme des instructions générales et ne représentent pas une solution de sécurité complète. Étant donné que ces bonnes pratiques peuvent ne pas être appropriées ou suffisantes pour votre environnement, considérez-les comme des remarques utiles plutôt que comme des recommandations.

> **Note**
>
> Pour plus de bonnes pratiques de sécurité, consultez Bonnes pratiques de sécurité pour Amazon EMR on EKS.

Application du principe du moindre privilège

Amazon EMR on EKS fournit une stratégie d'accès granulaire pour les applications utilisant des rôles IAM, tels que les rôles d'exécution. Ces rôles d'exécution sont associés aux comptes de service Kubernetes via la politique d'approbation du rôle IAM. Amazon EMR on EKS crée des pods au sein d'un espace de noms Amazon EKS enregistré qui exécutent le code d'application fourni par l'utilisateur. Les modules de travail exécutant le code de l'application assument le rôle d'exécution lors de la connexion à d'autres AWS services. Nous recommandons de n'accorder aux rôles d'exécution que l'ensemble minimal de privilèges requis par la tâche, tels que la couverture de votre application et l'accès à la destination du journal. Nous recommandons également de vérifier régulièrement les autorisations des tâches et lors de toute modification du code d'application.

Liste de contrôle d'accès des points de terminaison

Les points de terminaison gérés ne peuvent être créés que pour les clusters EKS configurés pour utiliser au moins un sous-réseau privé dans votre VPC. Cette configuration restreint l'accès aux

équilibreurs de charge créés par les points de terminaison gérés afin qu'ils ne soient accessibles que depuis votre VPC. Pour renforcer davantage la sécurité, nous vous recommandons de configurer des groupes de sécurité avec ces équilibreurs de charge afin qu'ils puissent limiter le trafic entrant à un ensemble sélectionné d'adresses IP.

Obtention des dernières mises à jour de sécurité des images personnalisées

Pour utiliser des images personnalisées avec Amazon EMR on EKS, vous pouvez installer n'importe quel binaire et bibliothèque sur l'image. Vous êtes responsable de l'application des correctifs de sécurité aux binaires que vous ajoutez à l'image. Les images Amazon EMR on EKS sont régulièrement corrigées avec les derniers correctifs de sécurité. Pour obtenir l'image la plus récente, vous devez recréer les images personnalisées chaque fois qu'une nouvelle version d'image de base est disponible dans la version Amazon EMR. Pour plus d'informations, consultez Versions Amazon EMR on EKS et Détails relatifs à la sélection d'une URI d'image de base.

Limitation de l'accès aux informations d'identification du pod

Kubernetes prend en charge plusieurs méthodes d'attribution d'informations d'identification à un pod. Le provisionnement de plusieurs fournisseurs d'informations d'identification peut accroître la complexité de votre modèle de sécurité. Amazon EMR on EKS a adopté l'utilisation des rôles IAM pour les comptes de services (IRSA) comme fournisseur d'informations d'identification standard au sein d'un espace de noms EKS enregistré. Les autres méthodes ne sont pas prises en charge, notamment kube2iam, kiam et l'utilisation d'un profil d'instance EC2 de l'instance exécutée sur le cluster.

Isolation du code d'application non fiable

Amazon EMR on EKS n'inspecte pas l'intégrité du code d'application soumis par les utilisateurs du système. Si vous exécutez un cluster virtuel multilocataire configuré avec plusieurs rôles d'exécution qui peuvent être utilisés pour soumettre des tâches par des clients non fiables exécutant du code arbitraire, il y a un risque qu'une application malveillante augmente ses privilèges. Dans ce cas, envisagez d'isoler les rôles d'exécution dotés de privilèges similaires dans un cluster virtuel différent.

Autorisations de contrôle d'accès basé sur les rôles (RBAC)

Les administrateurs doivent contrôler strictement les autorisations de contrôle d'accès basé sur les rôles (RBAC) pour Amazon EMR sur les espaces de noms gérés par EKS. Au minimum, les

autorisations suivantes ne doivent pas être accordées aux soumissionnaires de tâches dans Amazon EMR sur les espaces de noms gérés par EKS.

- Autorisations Kubernetes RBAC pour modifier la carte de configuration, car Amazon EMR on EKS utilise les cartes de configuration Kubernetes pour générer des modèles de pods gérés portant le nom du compte de service géré. Cet attribut ne doit pas être modifié.
- Autorisations RBAC Kubernetes permettant d'exécuter des tâches dans Amazon EMR sur des pods EKS, afin d'éviter de donner accès aux modèles de pods gérés qui portent le nom du compte de service géré. Cet attribut ne doit pas être modifié. Cette autorisation peut également donner accès au jeton JWT monté dans le pod, qui peut ensuite être utilisé pour récupérer les informations d'identification du rôle d'exécution.
- Autorisations Kubernetes RBAC pour créer des pods - pour empêcher les utilisateurs de créer des pods à l'aide d'un Kubernetes ServiceAccount qui peut être mappé à un rôle IAM doté de plus de privilèges que l'utilisateur. AWS
- Autorisations Kubernetes RBAC pour déployer un webhook mutant - afin d'empêcher les utilisateurs d'utiliser le webhook mutant pour muter le nom Kubernetes ServiceAccount des pods créés par Amazon EMR sur EKS.
- Autorisations RBAC Kubernetes pour lire les secrets Kubernetes, afin d'empêcher les utilisateurs de lire les données confidentielles stockées dans ces secrets.

Restriction de l'accès aux informations d'identification du rôle IAM du groupe de nœuds ou du profil d'instance

- Nous vous recommandons d'attribuer AWS des autorisations minimales aux rôles IAM du groupe de nœuds. Cela permet d'éviter l'augmentation des privilèges par un code qui pourrait s'exécuter en utilisant les informations d'identification du profil d'instance des nœuds de travail EKS.
- Pour bloquer complètement l'accès aux informations d'identification du profil d'instance à tous les pods exécutés dans Amazon EMR sur des espaces de noms gérés par EKS, nous vous recommandons d'exécuter des commandes `iptables` sur les nœuds EKS. Pour plus d'informations, consultez Restriction de l'accès aux informations d'identification du profil d'instance Amazon EC2. Il est toutefois important bien définir vos rôles IAM associés aux comptes de service afin que vos pods disposent de toutes les autorisations nécessaires. Par exemple, le rôle IAM du nœud se voit attribuer des autorisations pour extraire des images de conteneurs à partir d'Amazon ECR. Si ces autorisations ne sont pas attribuées à un pod, ce dernier ne peut pas extraire d'images de conteneurs à partir d'Amazon ECR. Le plug-in VPC CNI doit également être mis à jour. Pour

plus d'informations, consultez Procédure pas à pas : mise à jour du plug-in VPC CNI pour utiliser les rôles IAM pour les comptes de service.

Protection des données

Le modèle de responsabilité AWS partagée s'applique à la protection des données dans Amazon EMR on EKS. Comme décrit dans ce modèle, AWS est responsable de la protection de l'infrastructure mondiale qui gère l'ensemble du AWS cloud. La gestion du contrôle de votre contenu hébergé sur cette infrastructure est de votre responsabilité. Ce contenu comprend les tâches de configuration et de gestion de la sécurité des services AWS que vous utilisez. Pour plus d'informations sur la confidentialité des données, consultez les FAQ sur la confidentialité des données. Pour plus d'informations sur la protection des données en Europe, consultez le modèle de responsabilité AWS partagée et le billet de blog sur le RGPD sur le blog sur la AWS sécurité.

Pour des raisons de protection des données, nous vous recommandons de protéger les informations d'identification des AWS comptes et de configurer des comptes individuels avec AWS Identity and Access Management (IAM). Ainsi, chaque utilisateur se voit attribuer uniquement les autorisations nécessaires pour exécuter ses tâches. Nous vous recommandons également de sécuriser vos données comme indiqué ci-dessous :

- Utilisez l'authentification multifactorielle (MFA) avec chaque compte.
- SSL/TLS À utiliser pour communiquer avec AWS les ressources. Nous vous recommandons le certificat TLS 1.2 ou une version ultérieure.
- Configurez l'API et la journalisation de l'activité des utilisateurs avec AWS CloudTrail.
- Utilisez des solutions de AWS chiffrement, ainsi que tous les contrôles de sécurité par défaut au sein AWS des services.
- Utilisez des services de sécurité gérés avancés tels qu'Amazon Macie, qui contribuent à la découverte et à la sécurisation des données personnelles stockées dans Amazon S3.
- Utilisez les options de chiffrement Amazon EMR on EKS pour chiffrer les données au repos et en transit.
- Si vous avez besoin de modules cryptographiques validés par la norme FIPS 140-2 pour accéder AWS via une interface de ligne de commande ou une API, utilisez un point de terminaison FIPS. Pour plus d'informations sur les points de terminaison FIPS (Federal Information Processing Standard) disponibles, consultez Federal Information Processing Standard (FIPS) 140-2 (Normes de traitement de l'information fédérale).

Nous vous recommandons vivement de ne jamais placer d'informations identifiables sensibles, telles que les numéros de compte de vos clients, dans des champs de formulaire comme Name (Nom). Cela inclut lorsque vous travaillez avec Amazon EMR sur EKS ou sur d'autres AWS services à l'aide de la console, de l'API ou. AWS CLI AWS SDKs Toutes les données que vous saisissez dans Amazon EMR on EKS ou dans d'autres services peuvent être récupérées pour être incluses dans les journaux de diagnostic. Lorsque vous fournissez une URL à un serveur externe, n'incluez pas les informations d'identification non chiffrées dans l'URL pour valider votre demande adressée au serveur.

Chiffrement au repos

Le chiffrement des données vous permet d'empêcher les utilisateurs non autorisés de lire les données d'un cluster et celles des systèmes de stockage de données associés. Cela inclut les données enregistrées sur les supports persistants (données au repos) et les données qui peuvent être interceptées alors qu'elles circulent sur le réseau (données en transit).

Le chiffrement des données nécessite des clés et des certificats. Vous pouvez choisir parmi plusieurs options, notamment les clés gérées par AWS Key Management Service, les clés gérées par Amazon S3 et les clés et certificats fournis par les fournisseurs personnalisés que vous fournissez. Lorsque vous l'utilisez en AWS KMS tant que fournisseur de clés, des frais s'appliquent pour le stockage et l'utilisation des clés de chiffrement. Pour plus d'informations, consultez Tarification d'AWS KMS.

Avant d'indiquer les options de chiffrement, choisissez les systèmes de gestion des clés et des certificats que vous souhaitez utiliser. Créez ensuite les clés et les certificats pour les fournisseurs personnalisés que vous indiquez dans le cadre des paramètres de chiffrement.

Chiffrement au repos des données EMRFS dans Amazon S3

Le chiffrement Amazon S3 fonctionne avec les objets du système de fichiers EMR (EMRFS) lus et écrits sur Amazon S3. Vous indiquez le chiffrement côté serveur (SSE) ou le chiffrement côté client (CSE) sur Amazon S3 comme Mode de chiffrement par défaut lorsque vous activez le chiffrement au repos. Le cas échéant, vous pouvez spécifier différentes méthodes de chiffrement pour les compartiments individuels à l'aide de remplacements de chiffrement par compartiment. Que le chiffrement Amazon S3 soit activé ou non, le protocole TLS (Transport Layer Security) chiffre les objets EMRFS en transit entre les nœuds de cluster EMR et Amazon S3. Pour des informations plus détaillées sur le chiffrement Amazon S3, consultez la rubrique Protection des données à l'aide du chiffrement dans le Guide du développeur Amazon Simple Storage Service.

> **Note**
>
> Lorsque vous les utilisez AWS KMS, des frais s'appliquent pour le stockage et l'utilisation des clés de chiffrement. Pour plus d'informations, consultez Tarification d'AWS KMS.

Chiffrement côté serveur sur Amazon S3

Lorsque vous configurez le chiffrement côté serveur sur Amazon S3, Amazon S3 chiffre les données au niveau de l'objet au moment où elles sont écrites sur le disque et déchiffre les données lorsqu'elles sont accédées. Pour plus d'informations sur le chiffrement SSE, consultez la rubrique Protection des données à l'aide du chiffrement côté serveur dans le Guide du développeur Amazon Simple Storage Service.

Lorsque vous indiquez le chiffrement SSE sur Amazon EMR on EKS, vous pouvez choisir entre deux systèmes de gestion de clés différents :

- SSE-S3 : Amazon S3 gère les clés pour vous.
- SSE-KMS - Vous utilisez un AWS KMS key pour configurer des politiques adaptées à Amazon EMR sur EKS.

Le chiffrement SSE avec des clés fournies par le client (SSE-C) n'est pas disponible pour une utilisation avec Amazon EMR on EKS.

Chiffrement côté client sur Amazon S3

Avec le chiffrement côté client sur Amazon S3, le chiffrement et le déchiffrement par Amazon S3 se déroulent dans le client EMRFS de votre cluster. Les objets sont chiffrés avant d'être chargés sur Amazon S3 et déchiffrés après leur chargement. Le fournisseur que vous indiquez fournit la clé de chiffrement utilisée par le client. Le client peut utiliser les clés fournies par AWS KMS (CSE-KMS) ou une classe Java personnalisée qui fournit la clé racine côté client (CSE-C). Les spécificités du chiffrement sont légèrement différentes entre CSE-KMS et CSE-C, en fonction du fournisseur indiqué et des métadonnées de l'objet à déchiffrer ou à chiffrer. Pour plus d'informations sur ces différences, consultez la rubrique Protection des données à l'aide du chiffrement côté client dans le Guide du développeur Amazon Simple Storage Service.

> **Note**
>
> Le chiffrement CSE sur Amazon S3 garantit uniquement que les données EMRFS échangées avec Amazon S3 sont chiffrées ; cela ne signifie pas que toutes les données sur les volumes des instances du cluster sont chiffrées. De plus, étant donné que Hue n'utilise pas EMRFS, les objets que le navigateur de fichiers S3 de Hue écrit sur Amazon S3 ne sont pas chiffrés.

Chiffrement de disque local

Apache Spark prend en charge le chiffrement des données temporaires écrites sur les disques locaux. Ceci s'applique aux fichiers utilisés pour la redistribution de données, aux données temporairement écrites sur le disque lorsque la mémoire est insuffisante, ainsi qu'aux blocs de données enregistrés sur disque destinés à être mis en cache ou utilisés comme variables de diffusion. Il ne couvre pas le chiffrement des données de sortie générées par des applications APIs telles que `saveAsHadoopFile` ou`saveAsTable`. Ceci peut également ne pas inclure les fichiers temporaires créés explicitement par l'utilisateur. Pour plus d'informations, consultez la rubrique Chiffrement du stockage local dans la documentation Spark. Spark ne prend pas en charge les données chiffrées sur le disque local, telles que les données intermédiaires écrites sur un disque local par un processus exécuteur lorsque les données ne rentrent pas dans la mémoire. Les données enregistrées sur le disque sont destinées à être utilisées uniquement pendant la durée d'exécution de la tâche, et la clé qui sert à chiffrer ces données est créée de manière dynamique par Spark pour chaque exécution de tâche. Une fois la tâche Spark terminée, aucun autre processus ne peut déchiffrer les données.

Pour les pods du pilote et de l'exécuteur, vous chiffrez les données au repos qui sont enregistrées sur le volume monté. Il existe trois options de stockage AWS natives différentes que vous pouvez utiliser avec Kubernetes : EBS, EFS et pour Lustre. FSx Toutes les trois offrent un chiffrement au repos à l'aide d'une clé gérée par le service ou d'une clé AWS KMS key. Pour plus d'informations, consultez le Guide des bonnes pratiques EKS. En utilisant cette méthode, toutes les données enregistrées sur le volume monté sont chiffrées.

Gestion des clés

Vous pouvez configurer KMS pour qu'il effectue automatiquement la rotation de vos clés KMS. Ce système permet d'effectuer une rotation de vos clés une fois par an tout en conservant indéfiniment

les anciennes clés, afin que vos données puissent toujours être déchiffrées. Pour plus d'informations, voir Rotation AWS KMS keys.

Chiffrement en transit

Plusieurs mécanismes de chiffrement sont activés avec le chiffrement en transit. Il s'agit de fonctionnalités open-source et spécifiques à l'application, qui peuvent varier selon la version Amazon EMR on EKS. Les fonctionnalités de chiffrement spécifiques à l'application suivantes peuvent être activées avec Amazon EMR on EKS :

- Spark
  - La communication RPC interne entre les composants Spark, tels que le service de transfert de blocs et le service de mélange externe, est chiffrée à l'aide du code AES-256 dans les versions 5.9.0 et ultérieures d'Amazon EMR. Dans les versions précédentes, les communications RPC internes étaient chiffrées à l'aide de SASL avec DIGEST- MD5 comme algorithme de chiffrement.
  - Les communications HTTP avec les interfaces utilisateur comme le serveur d'historique Spark et les serveurs de fichiers HTTPS sont chiffrées à l'aide de la configuration SSL de Spark. Pour plus d'informations, consultez Configuration SSL dans la documentation Spark.

  Pour plus d'informations, consultez la rubrique Paramètres de sécurité Spark.
- Vous devez autoriser uniquement les connexions chiffrées via HTTPS (TLS) conformément à la SecureTransport condition aws : des politiques IAM du compartiment Amazon S3.
- Les résultats de requête qui diffusent en continu vers les clients JDBC ou ODBC sont chiffrés à l'aide du protocole TLS.

Gestion de l’identité et des accès

Gestion des identités et des accès AWS (IAM) est un outil Service AWS qui permet à un administrateur de contrôler en toute sécurité l'accès aux AWS ressources. Des administrateurs IAM contrôlent les personnes qui peuvent être authentifiées (connectées) et autorisées (disposant d'autorisations) pour utiliser des ressources Amazon EMR on EKS. IAM est un Service AWS outil que vous pouvez utiliser sans frais supplémentaires.

Rubriques

- Public ciblé

- Authentification par des identités
- Gestion de l’accès à l’aide de politiques
- Fonctionnement d'Amazon EMR on EKS avec IAM
- Utilisation des rôles liés à un service pour Amazon EMR on EKS
- Politiques gérées pour Amazon EMR on EKS
- Utilisation des rôles d'exécution de tâches avec Amazon EMR on EKS
- Exemples de politiques basées sur l'identité pour Amazon EMR on EKS
- Politiques de contrôle d'accès basées sur les balises
- Résolution de problèmes concernant l'identité et l'accès Amazon EMR on EKS

Public ciblé

La façon dont vous utilisez Gestion des identités et des accès AWS (IAM) varie en fonction de votre rôle :

- Utilisateur du service : demandez des autorisations à votre administrateur si vous ne pouvez pas accéder aux fonctionnalités (voir Résolution de problèmes concernant l'identité et l'accès Amazon EMR on EKS)
- Administrateur du service : déterminez l’accès des utilisateurs et soumettez les demandes d’autorisation (voir Fonctionnement d'Amazon EMR on EKS avec IAM)
- Administrateur IAM : rédigez des politiques pour gérer l’accès (voir Exemples de politiques basées sur l'identité pour Amazon EMR on EKS)

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 en tant qu'utilisateur IAM ou en assumant un rôle IAM. Utilisateur racine d'un compte AWS

Vous pouvez vous connecter en tant qu'identité fédérée à l'aide d'informations d'identification provenant d'une source d'identité telle que AWS IAM Identity Center (IAM Identity Center), d'une authentification unique ou d'informations d'identification. Google/Facebook Pour plus d’informations sur la connexion, consultez Connexion à votre Compte AWS dans le Guide de l’utilisateur Connexion à AWS .

Pour l'accès par programmation, AWS fournit un SDK et une CLI pour signer les demandes de manière cryptographique. Pour plus d'informations, consultez Signature AWS Version 4 pour les demandes d'API dans le Guide de l'utilisateur IAM.

Compte AWS utilisateur root

Lorsque vous créez un Compte AWS, vous commencez par une seule identité de connexion appelée utilisateur Compte AWS root qui dispose d'un accès complet à toutes Services AWS les ressources. Il est vivement déconseillé d'utiliser l'utilisateur racine pour vos tâches quotidiennes. Pour les tâches qui requièrent des informations d'identification de l'utilisateur racine, consultez Tâches qui requièrent les informations d'identification de l'utilisateur racine dans le Guide de l'utilisateur IAM.

Identité fédérée

Il est recommandé d'obliger les utilisateurs humains à utiliser la fédération avec un fournisseur d'identité pour accéder à Services AWS l'aide d'informations d'identification temporaires.

Une identité fédérée est un utilisateur provenant de l'annuaire de votre entreprise, de votre fournisseur d'identité Web ou Directory Service qui y accède à Services AWS l'aide d'informations d'identification provenant d'une source d'identité. Les identités fédérées assument des rôles qui fournissent des informations d'identification temporaires.

Pour une gestion des accès centralisée, nous vous recommandons d'utiliser AWS IAM Identity Center. Pour plus d'informations, consultez Qu'est-ce que IAM Identity Center ? dans le Guide de l'utilisateur AWS IAM Identity Center .

Utilisateurs et groupes IAM

Un utilisateur IAM est une identité qui dispose d'autorisations spécifiques pour une seule personne ou application. Nous vous recommandons d'utiliser ces informations d'identification temporaires au lieu des utilisateurs IAM avec des informations d'identification à long terme. Pour plus d'informations, voir Exiger des utilisateurs humains qu'ils utilisent la fédération avec un fournisseur d'identité pour accéder à AWS l'aide d'informations d'identification temporaires dans le guide de l'utilisateur IAM.

Les groupes IAM spécifient une collection d'utilisateurs IAM et permettent de gérer plus facilement les autorisations pour de grands ensembles d'utilisateurs. Pour plus d'informations, consultez Cas d'utilisation pour les utilisateurs IAM dans le Guide de l'utilisateur IAM.

Rôles IAM

Un rôle IAM est une identité dotée d'autorisations spécifiques qui fournit des informations d'identification temporaires. Vous pouvez assumer un rôle en passant d'un rôle utilisateur à un rôle IAM (console) ou en appelant une opération AWS CLI ou AWS API. Pour plus d'informations, consultez Méthodes pour endosser un rôle dans le Guide de l'utilisateur IAM.

Les rôles IAM sont utiles pour l'accès des utilisateurs fédérés, les autorisations temporaires des utilisateurs IAM, les accès intercompte, les accès entre services et les applications exécutées sur Amazon EC2. Pour plus d'informations, consultez Accès intercompte aux ressources dans IAM dans le Guide de l'utilisateur IAM.

Gestion de l'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 définit les autorisations lorsqu'elles sont associées à une identité ou à une ressource. AWS évalue ces politiques lorsqu'un directeur fait une demande. La plupart des politiques sont stockées AWS sous forme de documents JSON. Pour plus d'informations les documents de politique JSON, consultez Vue d'ensemble des politiques JSON dans le Guide de l'utilisateur IAM.

À l'aide de politiques, les administrateurs précisent qui a accès à quoi en définissant quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

Par défaut, les utilisateurs et les rôles ne disposent d'aucune autorisation. Un administrateur IAM crée des politiques IAM et les ajoute aux rôles, que les utilisateurs peuvent ensuite assumer. Les politiques IAM définissent les autorisations quelle que soit la méthode que vous utilisez pour exécuter l'opération.

Politiques basées sur l'identité

Les stratégies basées sur l'identité sont des documents de stratégie d'autorisations JSON que vous attachez à une identité (utilisateur, groupe ou rôle). Ces politiques contrôlent les actions que peuvent exécuter ces identités, sur quelles ressources et dans quelles conditions. Pour découvrir comment créer une politique basée sur l'identité, consultez Définition d'autorisations IAM personnalisées avec des politiques gérées par le client dans le Guide de l'utilisateur IAM.

Les politiques basées sur l'identité peuvent être des politiques intégrées (intégrées directement dans une seule identité) ou des politiques gérées (politiques autonomes associées à plusieurs identités).

Pour découvrir comment choisir entre des politiques gérées et en ligne, consultez Choix entre les politiques gérées et les politiques en ligne dans le Guide de l'utilisateur IAM.

Politiques basées sur les ressources

Les politiques basées sur les ressources sont des documents de politique JSON que vous attachez à une ressource. Les exemples incluent les politiques de confiance de rôle IAM et les stratégies de compartiment Amazon S3. 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. Vous devez spécifier un principal dans une politique basée sur les ressources.

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.

Autres types de politique

AWS prend en charge des types de politiques supplémentaires qui peuvent définir les autorisations maximales accordées par les types de politiques les plus courants :

- Limites d'autorisations : une limite des autorisations définit le nombre maximum d'autorisations qu'une politique basée sur l'identité peut accorder à une entité IAM. Pour plus d'informations, consultez Limites d'autorisations pour des entités IAM dans le Guide de l'utilisateur IAM.
- Politiques de contrôle des services (SCPs) — Spécifiez les autorisations maximales pour une organisation ou une unité organisationnelle dans AWS Organizations. Pour plus d'informations, consultez Politiques de contrôle de service dans le Guide de l'utilisateur AWS Organizations .
- Politiques de contrôle des ressources (RCPs) : définissez le maximum d'autorisations disponibles pour les ressources de vos comptes. Pour plus d'informations, voir Politiques de contrôle des ressources (RCPs) dans le guide de AWS Organizations l'utilisateur.
- Politiques de session : politiques avancées que vous passez en tant que paramètre lorsque vous créez par programmation une session temporaire pour un rôle ou un utilisateur fédéré. 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 EMR on EKS avec IAM

Avant d'utiliser IAM pour gérer l'accès à Amazon EMR on EKS, découvrez les fonctionnalités IAM qui peuvent être utilisées avec Amazon EMR on EKS.

Fonctionnalités IAM que vous pouvez utiliser avec Amazon EMR on EKS

Fonctionnalité IAM	Prise en charge d'Amazon EMR on EKS
Politiques basées sur l’identité	Oui
Politiques basées sur les ressources	Non
Actions de politique	Oui
Ressources de politique	Oui
Clés de condition de politique	Oui
ACLs	Non
ABAC (étiquettes dans les politiques)	Oui
Informations d’identification temporaires	Oui
Autorisations de principal	Oui
Rôles du service	Non
Rôles liés à un service	Oui

Pour obtenir une vue d'ensemble de la façon dont Amazon EMR on EKS et d'autres AWS services fonctionnent avec la plupart des fonctionnalités IAM, consultez la section AWS Services compatibles avec IAM dans le guide de l'utilisateur IAM.

Politiques basées sur l'identité pour Amazon EMR on EKS

Prend en charge les politiques basées sur l’identité : oui

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 Définition d’autorisations IAM personnalisées avec des politiques gérées par le client dans le Guide de l’utilisateur IAM.

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. Pour découvrir 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.

Exemples de politiques basées sur l'identité pour Amazon EMR on EKS

Pour voir des exemples de politiques basées sur l'identité pour Amazon EMR on EKS, consultez Exemples de politiques basées sur l'identité pour Amazon EMR on EKS.

Politiques basées sur les ressources au sein d'Amazon EMR on EKS

Prend en charge les politiques basées sur les ressources : non

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. Services AWS

Pour permettre un accès intercompte, vous pouvez spécifier un compte entier ou des entités IAM dans un autre compte en tant que principal dans une politique basée sur les ressources. Pour plus d’informations, consultez Accès intercompte aux ressources dans IAM dans le Guide de l’utilisateur IAM.

Actions de politique pour Amazon EMR on EKS

Prend en charge les actions de politique : oui

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. Intégration d'actions dans une politique afin d'accorder l'autorisation d'exécuter les opérations associées.

Pour afficher la liste des actions Amazon EMR on EKS, consultez Actions, ressources et clés de condition pour Amazon EMR on EKS dans la Référence de l'autorisation de service.

Les actions de politique dans Amazon EMR on EKS utilisent le préfixe suivant avant l'action :

```
emr-containers
```

Pour indiquer plusieurs actions dans une seule déclaration, séparez-les par des virgules.

```
"Action": [
      "emr-containers:action1",
      "emr-containers:action2"
         ]
```

Pour voir des exemples de politiques basées sur l'identité pour Amazon EMR on EKS, consultez Exemples de politiques basées sur l'identité pour Amazon EMR on EKS.

Ressources de politique pour Amazon EMR on EKS

Prend en charge les ressources de politique : oui

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. Il est recommandé de définir une ressource à l'aide de son Amazon Resource Name (ARN). Pour les actions qui ne sont pas compatibles avec les autorisations de niveau ressource, utilisez un caractère générique (*) afin d'indiquer que l'instruction s'applique à toutes les ressources.

```
"Resource": "*"
```

Pour consulter la liste des types de ressources Amazon EMR sur EKS et leurs caractéristiques ARNs, consultez la section Ressources définies par Amazon EMR sur EKS dans le Service Authorization Reference. Pour savoir avec quelles actions vous pouvez spécifier pour l'ARN de chaque ressource, consultez Actions, ressources et clés de condition pour Amazon EMR on EKS.

Pour voir des exemples de politiques basées sur l'identité pour Amazon EMR on EKS, consultez Exemples de politiques basées sur l'identité pour Amazon EMR on EKS.

Clés de condition de politique pour Amazon EMR on EKS

Prend en charge les clés de condition de politique spécifiques au service : oui

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 `Condition` indique à quel moment les instructions s'exécutent en fonction de critères définis. Vous pouvez créer des expressions conditionnelles qui utilisent des opérateurs de condition, tels que les signes égal ou inférieur à, pour faire correspondre la condition de la politique aux valeurs de la demande. Pour voir toutes les clés de condition AWS globales, voir les clés de contexte de condition AWS globales dans le guide de l'utilisateur IAM.

Pour consulter la liste des clés de condition d'Amazon EMR on EKS et savoir quelles actions et ressources vous pouvez utiliser avec une clé de condition, consultez Actions, ressources et clés de condition pour Amazon EMR on EKS dans la Référence de l'autorisation de service.

Pour voir des exemples de politiques basées sur l'identité pour Amazon EMR on EKS, consultez Exemples de politiques basées sur l'identité pour Amazon EMR on EKS.

Listes de contrôle d'accès (ACLs) dans Amazon EMR on EKS

Supports ACLs : Non

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.

Contrôle d'accès basé sur les attributs (ABAC) avec Amazon EMR on EKS

Prend en charge ABAC (étiquettes dans les politiques)	Oui

Le contrôle d’accès par attributs (ABAC) est une stratégie d’autorisation qui définit les autorisations en fonction des attributs appelés balises. Vous pouvez associer des balises aux entités et aux AWS ressources IAM, puis concevoir des politiques ABAC pour autoriser les opérations lorsque la balise du principal correspond à la balise de la ressource.

Pour contrôler l’accès basé sur des étiquettes, vous devez fournir les informations d’étiquette dans l’élément de condition d’une politique utilisant les clés de condition `aws:ResourceTag/key-name`, `aws:RequestTag/key-name` ou `aws:TagKeys`.

Si un service prend en charge les trois clés de condition pour tous les types de ressources, alors la valeur pour ce service est Oui. Si un service prend en charge les trois clés de condition pour certains types de ressources uniquement, la valeur est Partielle.

Pour plus d’informations sur ABAC, consultez Définition d’autorisations avec l’autorisation ABAC dans le Guide de l’utilisateur IAM. Pour accéder à un didacticiel décrivant les étapes de configuration de l’ABAC, consultez Utilisation du contrôle d’accès par attributs (ABAC) dans le Guide de l’utilisateur IAM.

Utilisation d'informations d'identification temporaires avec Amazon EMR on EKS

Prend en charge les informations d’identification temporaires : oui

Les informations d'identification temporaires fournissent un accès à court terme aux AWS ressources et sont automatiquement créées lorsque vous utilisez la fédération ou que vous changez de rôle. AWS recommande de générer dynamiquement des informations d'identification temporaires au lieu d'utiliser des clés d'accès à long terme. Pour plus d’informations, consultez Informations d’identification de sécurité temporaires dans IAM et Services AWS compatibles avec IAM dans le Guide de l’utilisateur IAM.

Autorisations de principal entre services pour Amazon EMR on EKS

Prend en charge les sessions d’accès direct (FAS) : oui

Les sessions d'accès direct (FAS) utilisent les autorisations du principal appelant et Service AWS, combinées Service AWS à la demande d'envoi de demandes aux services en aval. Pour plus de détails sur la politique relative à la transmission de demandes FAS, consultez la section Sessions de transmission d'accès.

Fonctions du service Amazon EMR on EKS

Prend en charge les fonctions de service	Non

Rôles liés au service Amazon EMR on EKS

Prend en charge les rôles liés à un service.	Oui

Pour plus d'informations sur la création ou la gestion des rôles liés à un service, consultez Services AWS qui fonctionnent avec IAM. Recherchez un service dans le tableau qui inclut un `Yes` dans la colonne Rôle lié à un service. Choisissez le lien Oui pour consulter la documentation du rôle lié à ce service.

Utilisation des rôles liés à un service pour Amazon EMR on EKS

Amazon EMR sur EKS utilise des rôles liés à un Gestion des identités et des accès AWS service (IAM). Le rôle lié à un service est un type unique de rôle IAM lié directement à Amazon EMR on EKS. Les rôles liés à un service sont prédéfinis par Amazon EMR sur EKS et incluent toutes les autorisations dont le service a besoin pour appeler d'autres AWS services en votre nom.

Le rôle lié à un service simplifie la configuration d'Amazon EMR on EKS, car vous n'avez pas besoin d'ajouter manuellement les autorisations requises. Amazon EMR on EKS définit les autorisations de ses rôles liés à un service ; sauf définition contraire, seul Amazon EMR on EKS peut assumer ses rôles. Les autorisations définies comprennent la politique de confiance 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 EMR on 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 qui prennent en charge les rôles liés à un service, consultez Services AWS qui fonctionnent avec IAM et recherchez les services avec un Oui dans la

colonne Rôle lié 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 EMR on EKS

Amazon EMR sur EKS utilise le rôle lié au service nommé. AWSServiceRoleForAmazonEMRContainers

Le rôle lié à un service `AWSServiceRoleForAmazonEMRContainers` approuve les services suivants pour endosser le rôle :

- `emr-containers.amazonaws.com`

La politique d'autorisations liée au rôle `AmazonEMRContainersServiceRolePolicy` permet à Amazon EMR on EKS de réaliser un ensemble d'actions sur les ressources spécifiées, comme le montre la déclaration de politique ci-dessous.

> **Note**
>
> Le contenu de la politique gérée change, c'est pourquoi la politique présentée ici peut l'être out-of-date. Consultez la majeure partie up-to-date de la documentation sur les politiques d'Amazon EMRContainers ServiceRolePolicy dans le AWS Managed Policy Reference Guide.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "eks:DescribeCluster",
        "eks:ListNodeGroups",
        "eks:DescribeNodeGroup",
        "ec2:DescribeRouteTables",
        "ec2:DescribeSubnets",
        "ec2:DescribeSecurityGroups",
        "elasticloadbalancing:DescribeInstanceHealth",
```

```
        "elasticloadbalancing:DescribeLoadBalancers",
        "elasticloadbalancing:DescribeTargetGroups",
        "elasticloadbalancing:DescribeTargetHealth",
        "eks:ListPodIdentityAssociations",
        "eks:DescribePodIdentityAssociation"
      ],
      "Resource": [
        "*"
      ],
      "Sid": "AllowEKSDescribecluster"
    },
    {
      "Effect": "Allow",
      "Action": [
        "acm:ImportCertificate",
        "acm:AddTagsToCertificate"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/emr-container:endpoint:managed-certificate": "true"
        }
      },
      "Sid": "AllowACMImportcertificate"
    },
    {
      "Effect": "Allow",
      "Action": [
        "acm:DeleteCertificate"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/emr-container:endpoint:managed-certificate": "true"
        }
      },
      "Sid": "AllowACMDeletecertificate"
    }
  ]
```

```
}
```

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 en savoir plus, consultez Service-Linked Role Permissions (autorisations du rôle lié à un service) dans le Guide de l'utilisateur IAM.

Création d'un rôle lié à un service pour Amazon EMR on EKS

Vous n'avez pas besoin de créer manuellement un rôle lié à un service. Lorsque vous créez un cluster virtuel, Amazon EMR on 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 créez un cluster virtuel, Amazon EMR on EKS crée à nouveau le rôle lié au service pour vous.

Vous pouvez également utiliser la console IAM pour créer un rôle lié au service avec le cas d'utilisation Amazon EMR on EKS. Dans l'API AWS CLI ou dans l' AWS API, créez un rôle lié à un service avec le nom du `emr-containers.amazonaws.com` service. Pour plus d'informations, consultez Création d'un rôle lié à un service dans le Guide de l'utilisateur IAM. Si vous supprimez ce rôle lié à un service, vous pouvez utiliser ce même processus pour créer le rôle à nouveau.

Modification d'un rôle lié à un service pour Amazon EMR on EKS

Amazon EMR on EKS ne vous autorise pas à modifier le rôle lié à un service `AWSServiceRoleForAmazonEMRContainers`. 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 en savoir plus, 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 EMR on 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 les ressources de votre rôle lié à un service avant de pouvoir les supprimer manuellement.

> **Note**
>
> Si le service Amazon EMR on EKS utilise le rôle lorsque vous essayez de supprimer les ressources, la suppression peut échouer. Si cela se produit, patientez quelques minutes et réessayez.

Suppression des ressources Amazon EMR on EKS utilisées par le rôle **`AWSServiceRoleForAmazonEMRContainers`**

1. Ouvrez la console Amazon EMR.
2. Choisissez un cluster virtuel.
3. Sur la page `Virtual Cluster`, choisissez Supprimer.
4. Répétez cette procédure pour tous les autres clusters virtuels de votre compte.

Pour supprimer manuellement le rôle lié au service à l'aide d'IAM

Utilisez la console IAM, le AWS CLI, ou l' AWS API pour supprimer le rôle lié au `AWSServiceRoleForAmazonEMRContainers` service. Pour en savoir plus, 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 EMR on EKS

Amazon EMR on EKS prend en charge l'utilisation des rôles liés à un service dans toutes les régions où le service est disponible. Pour de plus amples informations, veuillez consulter Points de terminaison et quotas de service Amazon EMR on EKS.

Politiques gérées pour Amazon EMR on EKS

Consultez les détails des mises à jour apportées aux politiques AWS gérées pour Amazon EMR sur EKS depuis le 1er mars 2021.

Modifier	Description	Date
`AmazonEMRContainersServiceRolePolicy` - Ajout d'autorisations de lecture pour répertorier les	Les autorisations suivantes sont ajoutées à la politique :`eks:ListPodIdentityAssociat`	3 février 2023

Modifier	Description	Date
associations d'identité de pod EKS dans un cluster, et d'une autre autorisation de lecture pour renvoyer des informations descriptives sur les associations d'identité de pod dans un cluster. Pour plus d'informations, consultez Amazon EMRContainers ServiceRolePolicy.	`ions` ,`eks:DescribePodIdentityAssociation` .	
`AmazonEMRContainersServiceRolePolicy` – Ajout d'autorisations pour décrire et répertorier les groupes de nœuds Amazon EKS, décrire les groupes cibles des équilibreurs de charge et décrire l'état des cibles associées aux équilibreurs de charge.	Les autorisations suivantes sont ajoutées à la politique : `eks:ListNodeGroups` , `eks:DescribeNodeGroup` , `elasticloadbalancing:DescribeTargetGroups` , `elasticloadbalancing:DescribeTargetHealth` .	13 mars 2023
`AmazonEMRContainersServiceRolePolicy` - Ajout d'autorisations pour importer et supprimer des certificats dans AWS Certificate Manager.	Les autorisations suivantes sont ajoutées à la politique : `acm:ImportCertificate` , `acm:AddTagsToCertificate` , `acm:DeleteCertificate` .	3 décembre 2021
Amazon EMR on EKS a commencé à assurer le suivi des modifications	Amazon EMR on EKS a commencé à suivre les modifications apportées à ses politiques AWS gérées.	1er mars 2021

Utilisation des rôles d'exécution de tâches avec Amazon EMR on EKS

Pour utiliser la commande `StartJobRun` afin de soumettre une tâche exécutée sur un cluster EKS, vous devez d'abord intégrer un rôle d'exécution de tâche à utiliser avec un cluster virtuel. Pour plus d'informations, consultez Création d'un rôle d'exécution des tâches dans Configuration d'Amazon EMR on EKS. Vous pouvez également suivre les instructions de la section Création d'un rôle IAM pour l'exécution des tâches de l'atelier Amazon EMR on EKS.

Les autorisations suivantes doivent être incluses dans la politique d'approbation du rôle d'exécution des tâches.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sts:AssumeRoleWithWebIdentity"
      ],
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "aws:userid": "system:serviceaccount:NAMESPACE:emr-containers-sa-*-
*-AWS_ACCOUNT_ID-BASE36_ENCODED_ROLE_NAME"
        }
      },
      "Sid": "AllowSTSAssumerolewithwebidentity"
    }
  ]
}
```

La politique d'approbation décrite dans l'exemple précédent accorde des autorisations uniquement à un compte de service Kubernetes géré par Amazon EMR dont le nom correspond au modèle `emr-containers-sa-*-*-`*AWS_ACCOUNT_ID*`-`*BASE36_ENCODED_ROLE_NAME*. Les comptes de service utilisant ce modèle seront automatiquement créés lors de la soumission de la tâche et limités à l'espace de noms dans lequel vous soumettez la tâche. Cette politique d'approbation permet à ces comptes de service d'assumer le rôle d'exécution et d'obtenir les informations d'identification

temporaires du rôle d'exécution. Les comptes de service d'un autre cluster Amazon EKS ou d'un espace de noms différent au sein du même cluster EKS ne sont pas autorisés à assumer le rôle d'exécution.

Vous pouvez exécuter la commande suivante pour mettre à jour automatiquement la politique d'approbation dans le format indiqué ci-dessus.

```
aws emr-containers update-role-trust-policy \
       --cluster-name cluster \
       --namespace namespace \
       --role-name iam_role_name_for_job_execution
```

Contrôle de l'accès au rôle d'exécution

L'administrateur de votre cluster Amazon EKS peut créer un cluster virtuel Amazon EMR on EKS multilocataire auquel un administrateur IAM peut ajouter plusieurs rôles d'exécution. Étant donné que des locataires non fiables peuvent utiliser ces rôles d'exécution pour soumettre des tâches exécutant du code arbitraire, vous voudrez peut-être restreindre ces locataires afin qu'ils ne puissent pas exécuter de code qui s'arrogerait les autorisations attribuées à un ou plusieurs de ces rôles d'exécution. Pour restreindre la politique IAM attachée à une identité IAM, l'administrateur IAM peut utiliser la clé de condition Amazon Resource Name (ARN) facultative `emr-containers:ExecutionRoleArn`. Cette condition accepte une liste de rôles d'exécution ARNs autorisés à accéder au cluster virtuel, comme le montre la politique d'autorisation suivante.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "emr-containers:StartJobRun"
      ],
      "Resource": [
        "arn:aws:emr-containers:*:*:/virtualclusters/VIRTUAL_CLUSTER_ID"
      ],
      "Condition": {
        "ArnEquals": {
          "emr-containers:ExecutionRoleArn": [
```

```
            "arn:aws:iam::*:role/execution_role_name_1",
            "arn:aws:iam::*:role/execution_role_name_2"
          ]
        }
      },
      "Sid": "AllowEMRCONTAINERSStartjobrun"
    }
  ]
}
```

Si vous souhaitez autoriser tous les rôles d'exécution commençant par un préfixe particulier, par exemple `MyRole`, vous pouvez remplacer l'opérateur de condition `ArnEquals` par l'opérateur `ArnLike`, et vous pouvez remplacer la valeur `execution_role_arn` de la condition par un caractère générique `*`. Par exemple, `arn:aws:iam::AWS_ACCOUNT_ID:role/MyRole*`. Toutes les autres clés de condition ARN sont également prises en charge.

> **Note**
>
> Avec Amazon EMR on EKS, vous ne pouvez pas accorder d'autorisations aux rôles d'exécution en fonction de balises ou d'attributs. Amazon EMR on EKS ne prend pas en charge le contrôle d'accès basé sur les balises (TBAC) ou le contrôle d'accès par attributs (ABAC) pour les rôles d'exécution.

Exemples de politiques basées sur l'identité pour Amazon EMR on EKS

Par défaut, les utilisateurs et les rôles ne sont pas autorisés à créer ou à modifier des ressources Amazon EMR on EKS. 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.

Pour apprendre à créer une politique basée sur l'identité IAM à l'aide de ces exemples de documents de politique JSON, consultez Création de politiques IAM (console) dans le Guide de l'utilisateur IAM.

Pour plus de détails sur les actions et les types de ressources définis par Amazon EMR sur EKS, y compris le format du ARNs pour chacun des types de ressources, consultez la section Actions, ressources et clés de condition pour Amazon EMR sur EKS dans la référence d'autorisation de service.

Rubriques

- Bonnes pratiques en matière de politiques
- Utilisation de la console Amazon EMR on EKS
- Autorisation accordée aux utilisateurs pour afficher leurs propres autorisations

Bonnes pratiques en matière de politiques

Les politiques basées sur l'identité déterminent si une personne peut créer, consulter ou supprimer des ressources Amazon EMR on EKS dans votre compte. Ces actions peuvent entraîner des frais pour votre Compte AWS. 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 Compte AWS. 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 service spécifique Service AWS, tel que 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 politiques avec 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, activez l'authentification MFA pour une sécurité accrue. Compte AWS Pour exiger la MFA lorsque des opérations d'API sont appelées, ajoutez des conditions MFA à vos politiques. Pour plus d'informations, consultez Sécurisation de l'accès aux API avec 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 EMR on EKS

Pour accéder à la console Amazon EMR on EKS, vous devez disposer d'un ensemble minimum d'autorisations. Ces autorisations doivent vous permettre de répertorier et consulter des informations sur les ressources Amazon EMR on EKS dans votre compte Compte AWS. Si vous créez une politique basée sur l'identité qui est plus restrictive que l'ensemble minimum d'autorisations requis, la console ne fonctionnera pas comme prévu pour les entités (utilisateurs ou rôles) tributaires de cette politique.

Il n'est pas nécessaire d'accorder des autorisations de console minimales aux utilisateurs qui appellent uniquement l'API AWS CLI ou l' AWS API. Autorisez plutôt l'accès à uniquement aux actions qui correspondent à l'opération d'API qu'ils tentent d'effectuer.

Pour garantir que les utilisateurs et les rôles peuvent toujours utiliser la console Amazon EMR on EKS, associez également la politique Amazon EMR on EKS `ConsoleAccess` ou la politique `ReadOnly` AWS gérée aux entités. Pour plus d'informations, consultez Ajout d'autorisations à un utilisateur dans le Guide de l'utilisateur IAM.

Autorisation accordée aux utilisateurs pour 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 l'API AWS CLI or AWS .

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

Politiques de contrôle d'accès basées sur les balises

Vous pouvez utiliser des conditions dans votre politique basée sur l'identité pour contrôler l'accès aux clusters virtuels et aux exécutions de tâches en fonction des balises. Pour plus d'informations sur le balisage, consultez Balisage de vos ressources Amazon EMR on EKS.

Les exemples suivants illustrent différents scénarios et différentes façons d'utiliser des opérateurs de condition avec des clés de condition Amazon EMR on EKS. Ces déclarations de politique IAM sont conçues uniquement à des fins de démonstration et ne doivent pas être utilisées dans des environnements de production. Il existe plusieurs façons de combiner des instructions de

stratégie pour accorder et refuser des autorisations selon vos besoins. Pour plus d'informations sur la planification et le test des politiques IAM, consultez le Guide de l'utilisateur IAM.

> **Important**
>
> Le refus explicite d'autoriser l'attribution de balises doit être pris en considération. Cela empêche les utilisateurs de baliser une ressource et de s'accorder ainsi des autorisations que vous n'aviez pas l'intention d'accorder. Si les actions de balisage d'une ressource ne sont pas refusées, un utilisateur peut modifier les balises et contourner l'intention des politiques basées sur les balises. Pour un exemple de politique qui refuse les actions de balisage, consultez Refuser l'accès pour ajouter et supprimer des balises.

Les exemples ci-dessous illustrent les politiques d'autorisation basées sur l'identité qui sont utilisées pour contrôler les actions autorisées sur les clusters virtuels d'Amazon EMR on EKS.

Autorisation d'actions uniquement sur les ressources ayant des valeurs de balises spécifiques

Dans l'exemple de politique suivant, l'opérateur de StringEquals condition essaie de faire correspondre dev à la valeur du tag department. Si la balise « department » n'a pas été ajoutée au cluster virtuel, ou ne contient pas la valeur « dev », la stratégie ne s'applique pas et les actions ne sont pas autorisées par cette politique. Si aucune autre déclaration de politique n'autorise ces actions, l'utilisateur peut uniquement utiliser des clusters virtuels ayant cette balise avec cette valeur.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "emr-containers:DescribeVirtualCluster"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "StringEquals": {
```

```
          "aws:ResourceTag/department": "dev"
        }
      },
      "Sid": "AllowEMRCONTAINERDescribevirtualcluster"
    }
  ]
}
```

Vous pouvez également spécifier plusieurs valeurs de balise à l'aide d'un opérateur de condition. Par exemple, pour autoriser les actions sur des clusters virtuels où la balise `department` a la valeur `dev` ou `test`, vous pouvez remplacer le bloc de condition dans l'exemple précédent avec les éléments suivants.

```
"Condition": {
        "StringEquals": {
          "aws:ResourceTag/department": ["dev", "test"]
        }
      }
```

Exiger le balisage lors de la création d'une ressource

Dans l'exemple ci-dessous, la balise doit être appliquée lors de la création du cluster virtuel.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "emr-containers:CreateVirtualCluster"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/department": "dev"
        }
```

```
      },
      "Sid": "AllowEMRCONTAINERSCreatevirtualcluster"
    }
  ]
}
```

La déclaration de politique suivante permet à l'utilisateur de créer un cluster virtuel uniquement si le cluster a une balise `department`, qui peut contenir n'importe quelle valeur.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "emr-containers:CreateVirtualCluster"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "Null": {
          "aws:RequestTag/department": "false"
        }
      },
      "Sid": "AllowEMRCONTAINERSCreatevirtualcluster"
    }
  ]
}
```

Refuser l'accès pour ajouter et supprimer des balises

L'effet de cette politique consiste à refuser à un utilisateur l'autorisation d'ajouter ou de supprimer des balises sur des clusters virtuels qui comportent une balise `department` comportant la valeur `dev`.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "emr-containers:TagResource",
        "emr-containers:UntagResource"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "StringNotEquals": {
          "aws:ResourceTag/department": "dev"
        }
      },
      "Sid": "AllowEMRCONTAINERSTagresource"
    }
  ]
}
```

Résolution de problèmes concernant l'identité et l'accès Amazon EMR on EKS

Utilisez les informations suivantes pour identifier et résoudre les problèmes courants que vous pouvez rencontrer lorsque vous travaillez avec Amazon EMR on EKS et IAM.

Rubriques

- Je ne suis pas autorisé à effectuer une action dans Amazon EMR on EKS
- Je ne suis pas autorisé à effectuer iam : PassRole
- Je souhaite autoriser des personnes extérieures à mon AWS compte à accéder à mes ressources Amazon EMR on EKS

Je ne suis pas autorisé à effectuer une action dans Amazon EMR on EKS

S'il vous AWS Management Console indique que vous n'êtes pas autorisé à effectuer une action, vous devez contacter votre administrateur pour obtenir de l'aide. Votre administrateur est la personne qui vous a fourni votre nom d'utilisateur et votre mot de passe.

L'exemple d'erreur suivant se produit quand l'utilisateur `mateojackson` tente d'utiliser la console pour afficher des informations détaillées sur une ressource *`my-example-widget`* fictive, mais ne dispose pas des autorisations `emr-containers:`*`GetWidget`* fictives.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform: emr-
containers:GetWidget on resource: my-example-widget
```

Dans ce cas, Mateo demande à son administrateur de mettre à jour ses politiques pour lui permettre d'accéder à la ressource *`my-example-widget`* à l'aide de l'action `emr-containers:`*`GetWidget`*.

Je ne suis pas autorisé à effectuer iam : PassRole

Si vous recevez une erreur selon laquelle vous n'êtes pas autorisé à exécuter l'action `iam:PassRole`, vos politiques doivent être mises à jour pour vous permettre de transmettre un rôle à Amazon EMR on EKS.

Certains vous Services AWS 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 EMR on EKS. Toutefois, l'action nécessite que le service ait des autorisations accordées par un rôle de 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'exécuter 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 EMR on 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 EMR on EKS est compatible avec ces fonctionnalités, consultez Fonctionnement d'Amazon EMR on EKS avec IAM.
- Pour savoir comment fournir l'accès à vos ressources sur celles Comptes AWS que vous possédez, consultez la section Fournir l'accès à un utilisateur IAM dans un autre utilisateur Compte AWS que vous possédez dans le Guide de l'utilisateur IAM.
- Pour savoir comment fournir l'accès à vos ressources à des tiers Comptes AWS, consultez la section Fournir un accès à des ressources Comptes AWS détenues 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.

Utilisation d'Amazon EMR sur EKS avec AWS Lake Formation pour un contrôle d'accès précis

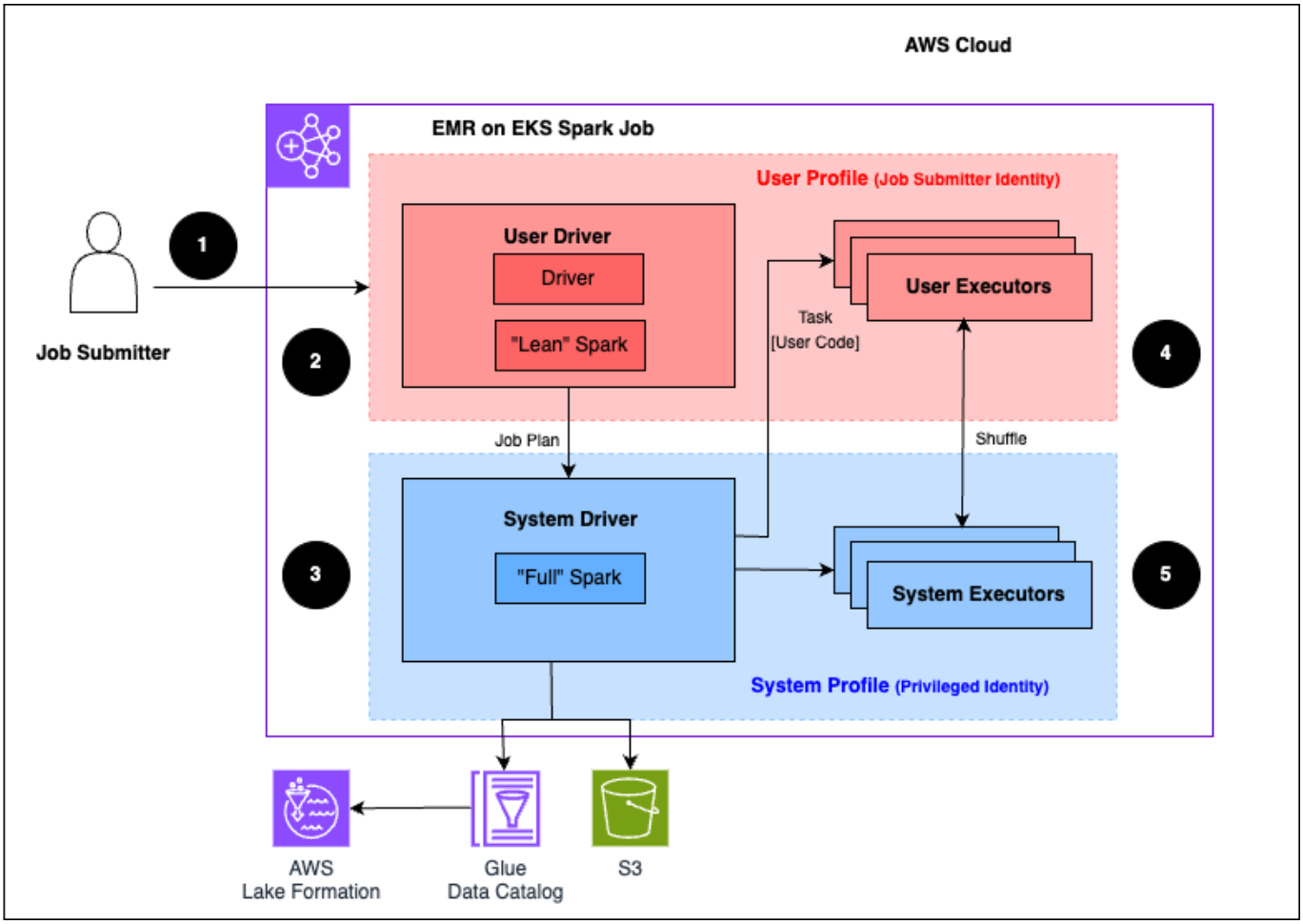
Avec les versions 7.7 et supérieures d'Amazon EMR, vous pouvez tirer parti de AWS Lake Formation pour appliquer des contrôles d'accès précis aux tables AWS Glue Data Catalog qui sont soutenues par des compartiments Amazon S3. Cette fonctionnalité vous permet de configurer des contrôles d'accès au niveau des tables, des lignes, des colonnes et des cellules pour les requêtes de lecture dans votre Amazon EMR sur EKS Spark Jobs.

Rubriques

Comment Amazon EMR sur EKS fonctionne avec Lake Formation AWS

L'utilisation d'Amazon EMR sur EKS avec Lake Formation vous permet d'appliquer une couche d'autorisations à chaque tâche Spark afin d'appliquer le contrôle des autorisations de Lake Formation lorsqu'Amazon EMR sur EKS exécute des tâches. Amazon EMR sur EKS utilise les profils de ressources Spark pour créer deux profils afin d'exécuter efficacement les tâches. Le profil utilisateur exécute le code fourni par l'utilisateur, tandis que le profil système applique les politiques de Lake Formation. Chaque Job activé par Lake Formation utilise deux pilotes Spark, l'un pour le profil utilisateur et l'autre pour le profil système. Pour plus d'informations, voir Qu'est-ce que AWS Lake Formation ?

Vous trouverez ci-dessous une présentation détaillée de la manière dont Amazon EMR on EKS accède aux données protégées par les politiques de sécurité de Lake Formation.



Les étapes suivantes décrivent ce processus :

1. Un utilisateur soumet un Spark Job à un cluster AWS virtuel Amazon EMR on EKS compatible avec Lake Formation.
2. Le service Amazon EMR on EKS configure le pilote utilisateur et exécute la tâche dans le profil utilisateur. Le pilote utilisateur exécute une version allégée de Spark qui n'est pas en mesure de lancer des tâches, des exécuteurs de requêtes, d'accéder à Amazon S3 ou au catalogue de données Glue. Il ne fait que créer un plan Job.
3. Le service Amazon EMR on EKS configure un deuxième pilote appelé pilote système et l'exécute dans le profil système (avec une identité privilégiée). Amazon EKS configure un canal TLS crypté entre les deux pilotes pour la communication. Le pilote utilisateur utilise le canal pour envoyer les plans de travail au pilote système. Le pilote système n'exécute pas le code envoyé par l'utilisateur. Il exécute Spark dans son intégralité et communique avec Amazon S3 et le catalogue de données pour accéder aux données. Il demande des exécuteurs et compile le Job Plan en une séquence d'étapes d'exécution.

4. Le service Amazon EMR on EKS exécute ensuite les étapes sur les exécuteurs. À n'importe quel stade, le code utilisateur est exécuté exclusivement sur les exécuteurs de profil utilisateur.
5. Les étapes qui lisent les données des tables du catalogue de données protégées par Lake Formation ou celles qui appliquent des filtres de sécurité sont déléguées aux exécuteurs du système.

Activez la formation de Lake avec Amazon EMR sur EKS

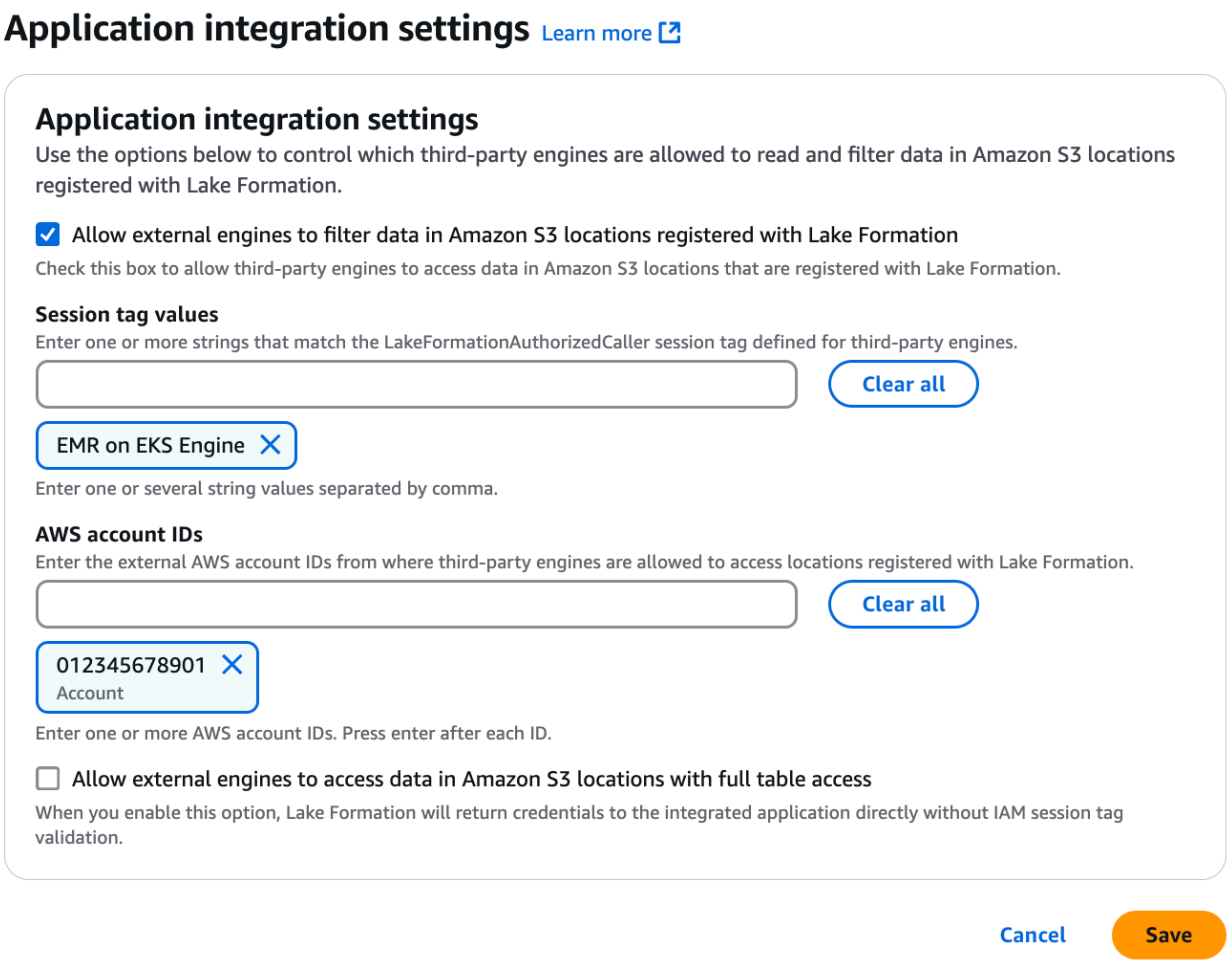
Avec les versions 7.7 et supérieures d'Amazon EMR, vous pouvez tirer parti de AWS Lake Formation pour appliquer des contrôles d'accès précis aux tables du catalogue de données soutenues par Amazon S3. Cette fonctionnalité vous permet de configurer les contrôles d'accès au niveau des tables, des lignes, des colonnes et des cellules pour les requêtes de lecture dans votre Amazon EMR sur EKS Spark Jobs.

Cette section explique comment créer une configuration de sécurité et configurer Lake Formation pour qu'il fonctionne avec Amazon EMR. Il décrit également comment créer un cluster virtuel avec la configuration de sécurité que vous avez créée pour Lake Formation. Ces sections sont destinées à être complétées dans l'ordre.

Étape 1 : configurer les autorisations au niveau des colonnes, des lignes ou des cellules basées sur Lake Formation

Tout d'abord, pour appliquer des autorisations au niveau des lignes et des colonnes à Lake Formation, l'administrateur du lac de données de Lake Formation doit définir le tag de LakeFormationAuthorizedCallersession. Lake Formation utilise cette balise de session pour autoriser les appelants et fournir l'accès au lac de données.

Accédez à la console AWS Lake Formation et sélectionnez l'option Paramètres d'intégration de l'application dans la section Administration de la barre latérale. Cochez ensuite la case Autoriser les moteurs externes à filtrer les données dans les sites Amazon S3 enregistrés auprès de Lake Formation. Ajoutez le AWS compte sur IDs lequel les tâches Spark seront exécutées et les valeurs des balises de session.



Notez que le tag de LakeFormationAuthorizedCallersession transmis ici est transmis SecurityConfigurationultérieurement lorsque vous configurez les rôles IAM, dans la section 3.

Étape 2 : Configuration des autorisations EKS RBAC

Ensuite, vous configurez des autorisations pour le contrôle d'accès basé sur les rôles.

Fournir des autorisations de cluster EKS au service Amazon EMR on EKS

Le service Amazon EMR on EKS doit disposer d'autorisations de rôle de cluster EKS afin de pouvoir créer des autorisations d'espaces de noms multiples permettant au pilote système de créer des exécuteurs utilisateur dans l'espace de noms utilisateur.

Créer un rôle de cluster

Cet exemple définit les autorisations pour un ensemble de ressources.

```
vim emr-containers-cluster-role.yaml
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: emr-containers
rules:
  - apiGroups: [""]
    resources: ["namespaces"]
    verbs: ["get"]
  - apiGroups: [""]
    resources: ["serviceaccounts", "services", "configmaps", "events", "pods", "pods/
log"]
    verbs: ["get", "list", "watch", "describe", "create", "edit", "delete",
 "deletecollection", "annotate", "patch", "label"]
  - apiGroups: [""]
    resources: ["secrets"]
    verbs: ["create", "patch", "delete", "watch"]
  - apiGroups: ["apps"]
    resources: ["statefulsets", "deployments"]
    verbs: ["get", "list", "watch", "describe", "create", "edit", "delete", "annotate",
 "patch", "label"]
  - apiGroups: ["batch"]
    resources: ["jobs"]
    verbs: ["get", "list", "watch", "describe", "create", "edit", "delete", "annotate",
 "patch", "label"]
  - apiGroups: ["extensions", "networking.k8s.io"]
    resources: ["ingresses"]
    verbs: ["get", "list", "watch", "describe", "create", "edit", "delete", "annotate",
 "patch", "label"]
  - apiGroups: ["rbac.authorization.k8s.io"]
    resources: ["clusterroles","clusterrolebindings","roles", "rolebindings"]
    verbs: ["get", "list", "watch", "describe", "create", "edit", "delete",
 "deletecollection", "annotate", "patch", "label"]
  - apiGroups: [""]
    resources: ["persistentvolumeclaims"]
    verbs: ["get", "list", "watch", "describe", "create", "edit", "delete",
 "deletecollection", "annotate", "patch", "label"]
  - apiGroups: ["kyverno.io"]
    resources: ["clusterpolicies"]
    verbs: ["create", "delete"]
---
```

```
kubectl apply -f emr-containers-cluster-role.yaml
```

Création de liaisons de rôles de cluster

```
vim emr-containers-cluster-role-binding.yaml
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: emr-containers
subjects:
- kind: User
  name: emr-containers
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: emr-containers
  apiGroup: rbac.authorization.k8s.io
---
```

```
kubectl apply -f emr-containers-cluster-role-binding.yaml
```

Fournir un accès à l'espace de noms au service Amazon EMR on EKS

Créez deux espaces de noms Kubernetes, l'un pour le pilote utilisateur et les exécuteurs, l'autre pour le pilote et les exécuteurs système, et activez l'accès au service Amazon EMR on EKS pour soumettre des tâches dans les espaces de noms utilisateur et système. Suivez le guide existant pour fournir un accès à chaque espace de noms, disponible sur Activer l'accès au cluster à l'aide `aws-auth` de.

Étape 3 : Configuration des rôles IAM pour les composants du profil utilisateur et du profil système

Troisièmement, vous définissez des rôles pour des composants spécifiques. Un Spark Job compatible avec Lake Formation comporte deux composants, l'utilisateur et le système. Le pilote utilisateur et les exécuteurs s'exécutent dans l'espace de noms utilisateur et sont liés à JobExecutionRole celui transmis dans l' StartJobRun API. Le pilote système et les exécuteurs s'exécutent dans l'espace de noms System et sont liés au QueryEnginerôle.

Configurer le rôle du moteur de requête

Le QueryEngine rôle est lié aux composants de l'espace système et serait autorisé à assumer le tag JobExecutionRolewith LakeFormationAuthorizedCallerSession. La politique d'autorisations IAM du rôle Query Engine est la suivante :

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AssumeJobRoleWithSessionTagAccessForSystemDriver",
      "Effect": "Allow",
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ],
      "Resource": [
        "arn:aws:iam::*:role/JobExecutionRole"
      ],
      "Condition": {
        "StringLike": {
          "aws:RequestTag/LakeFormationAuthorizedCaller": "EMR on EKS Engine"
        }
      }
    },
    {
      "Sid": "AssumeJobRoleWithSessionTagAccessForSystemExecutor",
      "Effect": "Allow",
      "Action": [
        "sts:AssumeRole"
      ],
      "Resource": [
        "arn:aws:iam::*:role/JobExecutionRole"
      ]
    },
    {
      "Sid": "CreateCertificateAccessForTLS",
      "Effect": "Allow",
      "Action": [
        "emr-containers:CreateCertificate"
      ],
```

```
      "Resource": [
        "*"
      ]
    }
  ]
}
```

Configurez la politique de confiance du rôle Query Engine pour faire confiance à l'espace de noms Kubernetes System.

```
aws emr-containers update-role-trust-policy \
    --cluster-name eks cluster \
    --namespace eks system namespace \
    --role-name query_engine_iam_role_name
```

Pour plus d'informations, consultez la section Mise à jour de la politique d'approbation des rôles.

Configuration du rôle d'exécution du Job

Les autorisations de Lake Formation contrôlent l'accès aux ressources du catalogue de données AWS Glue, aux sites Amazon S3 et aux données sous-jacentes de ces sites. Les autorisations IAM contrôlent l'accès à la Lake Formation and AWS Glue APIs et aux ressources. Bien que vous ayez l'autorisation Lake Formation d'accéder à une table du catalogue de données (SELECT), votre opération échoue si vous ne disposez pas de l'autorisation IAM pour les opérations d'`glue:Get*`API.

Politique d'autorisations IAM de JobExecutionRole: le JobExecutionrôle doit avoir les déclarations de politique dans sa politique d'autorisations.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "GlueCatalogAccess",
      "Effect": "Allow",
      "Action": [
        "glue:Get*",
        "glue:Create*",
```

```
        "glue:Update*"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Sid": "LakeFormationAccess",
      "Effect": "Allow",
      "Action": [
        "lakeformation:GetDataAccess"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Sid": "CreateCertificateAccessForTLS",
      "Effect": "Allow",
      "Action": [
        "emr-containers:CreateCertificate"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

Politique de confiance IAM pour JobExecutionRole:

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "TrustQueryEngineRoleForSystemDriver",
      "Effect": "Allow",
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
```

```
      ],
      "Resource": [
        "arn:aws:iam::*:role/QueryExecutionRole"
      ],
      "Condition": {
        "StringLike": {
          "aws:RequestTag/LakeFormationAuthorizedCaller": "EMR on EKS Engine"
        }
      }
    },
    {
      "Sid": "TrustQueryEngineRoleForSystemExecutor",
      "Effect": "Allow",
      "Action": [
        "sts:AssumeRole"
      ],
      "Resource": [
        "arn:aws:iam::*:role/QueryEngineRole"
      ]
    }
  ]
}
```

Configurez la politique de confiance du rôle d'exécution de Job pour faire confiance à l'espace de noms utilisateur Kubernetes :

```
aws emr-containers update-role-trust-policy \
    --cluster-name eks cluster \
    --namespace eks User namespace \
    --role-name job_execution_role_name
```

Pour plus d'informations, voir Mettre à jour la politique de confiance du rôle d'exécution des tâches.

Étape 4 : Configuration de la configuration de sécurité

Pour exécuter une tâche compatible avec Lake Formation, vous devez créer une configuration de sécurité.

```
aws emr-containers create-security-configuration \
    --name 'security-configuration-name' \
    --security-configuration '{
```

```
        "authorizationConfiguration": {
            "lakeFormationConfiguration": {
                "authorizedSessionTagValue": "SessionTag configured in LakeFormation",
                "secureNamespaceInfo": {
                    "clusterId": "eks-cluster-name",
                    "namespace": "system-namespace-name"
                },
                "queryEngineRoleArn": "query-engine-IAM-role-ARN"
            }
        }
    }'
```

Assurez-vous que le tag de session transmis dans le champ authorizedSessionTagValue peut autoriser Lake Formation. Définissez la valeur sur celle configurée dans Lake Formation, dansÉtape 1 : configurer les autorisations au niveau des colonnes, des lignes ou des cellules basées sur Lake Formation.

Étape 5 : Création d'un cluster virtuel

Créez un cluster virtuel Amazon EMR sur EKS avec une configuration de sécurité.

```
aws emr-containers create-virtual-cluster \
--name my-lf-enabled-vc \
--container-provider '{
    "id": "eks-cluster",
    "type": "EKS",
    "info": {
        "eksInfo": {
            "namespace": "user-namespace"
        }
    }
}' \
--security-configuration-id SecurityConfiguraionId
```

Assurez-vous que l'SecurityConfigurationidentifiant de l'étape précédente est transmis, afin que la configuration d'autorisation de Lake Formation soit appliquée à tous les jobs exécutés sur le cluster virtuel. Pour plus d'informations, consultez Enregistrer le cluster Amazon EKS auprès d'Amazon EMR.

Étape 6 : Soumettre un job dans le FGAC activé VirtualCluster

Le processus de soumission des offres d'emploi est le même pour les tâches autres que celles liées à Lake Formation et à Lake Formation. Pour plus d'informations, voir Soumettre une tâche exécutée avec `StartJobRun`.

Le pilote Spark, l'exécuteur et les journaux d'événements du pilote système sont stockés dans le compartiment S3 du compte de AWS service à des fins de débogage. Nous recommandons de configurer une clé KMS gérée par le client dans le Job Run pour chiffrer tous les journaux stockés dans le AWS service bucket. Pour plus d'informations sur l'activation du chiffrement des journaux, consultez la section Chiffrement d'Amazon EMR sur les journaux EKS.

Considérations et restrictions

Tenez compte des considérations et limitations suivantes lorsque vous utilisez Lake Formation avec Amazon EMR sur EKS :

- Amazon EMR on EKS prend en charge un contrôle d'accès précis via Lake Formation uniquement pour les formats de table Apache Hive, Apache Iceberg, Apache Hudi et Delta. Les formats Apache Hive incluent Parquet, ORC et XSv.
- `DynamicResourceAllocation`est activé par défaut et vous ne pouvez pas le désactiver `DynamicResourceAllocation` pour les tâches de Lake Formation. La valeur par défaut de la `spark.dynamicAllocation.maxExecutors` configuration DRA étant infinie, veuillez configurer une valeur appropriée en fonction de votre charge de travail.
- Les tâches compatibles avec Lake Formation ne prennent pas en charge l'utilisation d'EMR personnalisé sur les images EKS dans le pilote système et les exécuteurs système.
- Vous ne pouvez utiliser Lake Formation qu’avec des tâches Spark.
- EMR sur EKS with Lake Formation ne prend en charge qu'une seule session Spark pendant toute la durée d'une tâche.
- EMR sur EKS with Lake Formation ne prend en charge que les requêtes de table entre comptes partagées via des liens de ressources.
- Les éléments suivants ne sont pas pris en charge :
  - Jeux de données distribués résilients (RDD)
  - Spark Streaming
  - Écriture avec les autorisations accordées par Lake Formation
  - Contrôle d’accès pour les colonnes imbriquées

- L'EMR sur EKS bloque les fonctionnalités susceptibles de compromettre l'isolation complète du pilote système, notamment les suivantes :
  - UDTs, Hive UDFs et toute fonction définie par l'utilisateur impliquant des classes personnalisées
  - Sources de données personnalisées
  - Fourniture de fichiers JAR supplémentaires pour l'extension Spark, le connecteur ou la commande Metastore `ANALYZE TABLE`
- Pour appliquer les contrôles d’accès, les opérations `EXPLAIN PLAN` et DDL telles que `DESCRIBE TABLE` n’exposent pas les informations restreintes.
- Amazon EMR on EKS restreint l'accès aux journaux Spark du pilote système pour les tâches compatibles avec Lake Formation. Étant donné que le pilote système s’exécute avec plus d’accès, les événements et les journaux générés par le pilote système peuvent inclure des informations sensibles. Pour empêcher les utilisateurs ou le code non autorisés d'accéder à ces données sensibles, EMR on EKS a désactivé l'accès aux journaux des pilotes du système. Pour le dépannage, contactez AWS le support.
- Si vous avez enregistré l'emplacement d'une table auprès de Lake Formation, le chemin d'accès aux données passe par les informations d'identification stockées dans Lake Formation, indépendamment de l'autorisation IAM pour le rôle d'exécution de la tâche EMR sur EKS. Si vous configurez mal le rôle enregistré avec l'emplacement de la table, les tâches soumises qui utilisent le rôle avec l'autorisation S3 IAM sur l'emplacement de la table échoueront.
- L’écriture dans une table Lake Formation utilise l’autorisation IAM plutôt que les autorisations accordées par Lake Formation. Si votre rôle d'exécution de tâches dispose des autorisations S3 nécessaires, vous pouvez l'utiliser pour exécuter des opérations d'écriture.

Voici les restrictions et les considérations à prendre en compte lors de l’utilisation d’Apache Iceberg :

- Vous ne pouvez utiliser Apache Iceberg qu’avec un catalogue de sessions et non avec des catalogues nommés arbitrairement.
- Les tables Iceberg enregistrées dans Lake Formation ne prennent en charge que les tables de métadonnées `history`,`metadata_log_entries`,`snapshots`,, `filesmanifests`, et`refs`. Amazon EMR masque les colonnes susceptibles de contenir des données sensibles, telles que `partitionspath`, et. `summaries` Cette restriction ne s’applique pas aux tables Iceberg qui ne sont pas enregistrées dans Lake Formation.
- Les tables que vous n’enregistrez pas dans Lake Formation prennent en charge toutes les procédures stockées par Iceberg. Les procédures `register_table` et `migrate` ne sont prises en charge pour aucune table.

- Nous vous recommandons d'utiliser Iceberg DataFrameWriter V2 au lieu de V1.

Pour plus d'informations, consultez Comprendre les concepts et la terminologie d'Amazon EMR on EKS et Activer l'accès au cluster pour Amazon EMR on EKS.

Avertissement pour les administrateurs de données

> **Note**
>
> Lorsque vous accordez l'accès aux ressources de Lake Formation à un rôle IAM pour EMR sur EKS, vous devez vous assurer que l'administrateur ou l'opérateur du cluster EMR est un administrateur de confiance. Cela est particulièrement pertinent pour les ressources de Lake Formation partagées entre plusieurs organisations et AWS comptes.

Responsabilités des administrateurs d'EKS

- L'`System`espace de noms doit être protégé. Aucun utilisateur, ressource, entité ou outil ne serait autorisé à disposer d'autorisations Kubernetes RBAC sur les ressources Kubernetes de l'espace de noms. `System`
- Aucun utilisateur, ressource ou entité à l'exception du service EMR sur EKS ne doit avoir `CREATE` accès à POD, CONFIG_MAP et SECRET dans l'espace de noms. `User`
- `System`les pilotes et les `System` exécuteurs contiennent des données sensibles. Ainsi, les événements Spark, les journaux des pilotes Spark et les journaux de l'exécuteur Spark présents dans l'espace de `System` noms ne doivent pas être transférés vers des systèmes de stockage de journaux externes.

Résolution des problèmes

Logging

EMR sur EKS utilise les profils de ressources Spark pour diviser l'exécution des tâches. Amazon EMR on EKS utilise le profil utilisateur pour exécuter le code que vous avez fourni, tandis que le profil système applique les politiques de Lake Formation. Vous pouvez accéder aux journaux des conteneurs exécutés en tant que profil utilisateur en configurant la StartJobRun demande avec MonitoringConfiguration.

Serveur d'historique Spark

Le serveur d'historique Spark contient tous les événements Spark générés à partir du profil utilisateur et les événements expurgés générés à partir du pilote du système. Vous pouvez voir tous les conteneurs provenant des pilotes utilisateur et système dans l'onglet Executors. Toutefois, les liens du journal ne sont disponibles que pour le profil utilisateur.

Échec de la tâche en raison d’autorisations insuffisantes pour Lake Formation

Assurez-vous que votre rôle d'exécution de tâches dispose des autorisations nécessaires pour s'exécuter `SELECT` et `DESCRIBE` sur la table à laquelle vous accédez.

Échec de l’exécution de la tâche avec RDD

EMR sur EKS ne prend actuellement pas en charge les opérations de jeu de données distribué résilient (RDD) sur les tâches compatibles avec Lake Formation.

Impossible d’accéder aux fichiers de données dans Amazon S3

Assurez-vous d’avoir enregistré l’emplacement du lac de données dans Lake Formation.

Exception de validation de sécurité

EMR sur EKS a détecté une erreur de validation de sécurité. Contactez le AWS support pour obtenir de l'aide.

Partage du catalogue de données et des tableaux AWS Glue entre les comptes

Vous pouvez partager des bases de données et des tables entre les comptes tout en continuant à utiliser Lake Formation. Pour plus d'informations, voir Partage de données entre comptes dans Lake Formation et How do I share AWS Glue Data Catalog and tables cross-account using AWS Lake Formation ? .

Iceberg Job lance une erreur d'initialisation et ne définit pas la région AWS

Le message est le suivant :

```
25/02/25 13:33:19 ERROR SparkFGACExceptionSanitizer: Client received error with id =
 b921f9e6-f655-491f-b8bd-b2842cdc20c7,
reason = IllegalArgumentException, message = Cannot initialize
```

```
LakeFormationAwsClientFactory, please set client.region to a valid aws region
```

Assurez-vous que la configuration de Spark `spark.sql.catalog.catalog_name.client.region` est définie sur une région valide.

Iceberg Job lance SparkUnsupportedOperationException

Le message est le suivant :

```
25/02/25 13:53:15 ERROR SparkFGACExceptionSanitizer: Client received error with id =
 921fef42-0800-448b-bef5-d283d1278ce0,
reason = SparkUnsupportedOperationException, message = Either glue.id or glue.account-
id is set with non-default account.
Cross account access with fine-grained access control is only supported with AWS
 Resource Access Manager.
```

Assurez-vous que la configuration Spark `spark.sql.catalog.catalog_name.glue.account-id` est définie sur un identifiant de compte valide.

Iceberg Job échoue avec « 403 accès refusé » lors de l'opération MERGE

Le message est le suivant :

```
software.amazon.awssdk.services.s3.model.S3Exception: Access Denied (Service: S3,
 Status Code: 403,
...
 at
 software.amazon.awssdk.services.s3.DefaultS3Client.deleteObject(DefaultS3Client.java:3365)
 at org.apache.iceberg.aws.s3.S3FileIO.deleteFile(S3FileIO.java:162)
 at org.apache.iceberg.io.FileIO.deleteFile(FileIO.java:86)
 at
 org.apache.iceberg.io.RollingFileWriter.closeCurrentWriter(RollingFileWriter.java:129)
```

Désactivez les opérations S3 Delete dans Spark en ajoutant la propriété suivante. `--conf spark.sql.catalog.s3-table-name.s3.delete-enabled=false`.

Journalisation et surveillance

Pour détecter les incidents, recevoir des alertes en cas d'incident et y répondre, utilisez ces options avec Amazon EMR on EKS :

- Surveillez Amazon EMR sur EKS avec AWS CloudTrail - AWS CloudTrailfournit un enregistrement des actions entreprises par un utilisateur, un rôle ou un AWS service dans Amazon EMR sur EKS. Il capture les appels à partir de la console Amazon EMR et les appels de code aux opérations d'API Amazon EMR on EKS en tant qu'événements. Cela vous permet de déterminer quelle demande a été envoyée à Amazon EMR on EKS, l'adresse IP source à partir de laquelle la demande a été effectuée, l'auteur de la demande, la date de la demande, ainsi que d'autres informations. Pour de plus amples informations, veuillez consulter Journalisation d'Amazon EMR sur les appels d'API EKS à l'aide de AWS CloudTrail.
- CloudWatch Use Events with Amazon EMR on EKS - CloudWatch Events fournit un flux quasi en temps réel d'événements système décrivant les modifications apportées aux AWS ressources. CloudWatch Events prend connaissance des changements opérationnels au fur et à mesure qu'ils se produisent, y répond et prend les mesures correctives nécessaires, en envoyant des messages pour répondre à l'environnement, en activant des fonctions, en apportant des modifications et en capturant des informations d'état. Pour utiliser CloudWatch Events with Amazon EMR on EKS, créez une règle qui se déclenche lors d'un appel d'API Amazon EMR on EKS via. CloudTrail Pour de plus amples informations, veuillez consulter Surveillez les offres d'emploi avec Amazon CloudWatch Events.

Chiffrement des journaux Amazon EMR sur EKS avec un stockage géré

Les sections suivantes expliquent comment configurer le chiffrement des journaux.

Activer le chiffrement

Pour chiffrer les journaux du stockage géré à l'aide de votre propre clé KMS, utilisez la configuration suivante lorsque vous soumettez une exécution de tâche.

```
"monitoringConfiguration": {
            "managedLogs": {
                "allowAWSToRetainLogs":"ENABLED",
                "encryptionKeyArn":"KMS key arn"
            },
            "persistentAppUI": "ENABLED"
        }
```

La `allowAWSToRetainLogs` configuration permet de AWS conserver les journaux des espaces de noms du système lors de l'exécution d'une tâche à l'aide du FGAC natif. La `persistentAppUI` configuration permet AWS de sauvegarder les journaux d'événements qui sont utilisés pour générer

l'interface utilisateur de Spark. Le `encryptionKeyArn` est utilisé pour spécifier l'ARN de la clé KMS que vous souhaitez utiliser pour chiffrer les journaux stockés par AWS.

Autorisations requises pour le chiffrement des journaux

L'utilisateur qui soumet la tâche ou consulte l'interface utilisateur de Spark doit être autorisé à effectuer `kms:DescribeKey` les `kms:GenerateDataKey` actions et `kms:Decrypt` à obtenir la clé de chiffrement. Ces autorisations sont utilisées pour vérifier la validité de la clé et vérifier que l'utilisateur dispose des autorisations nécessaires pour lire et écrire des journaux chiffrés avec la clé KMS. Si l'utilisateur qui soumet la tâche ne dispose pas des autorisations clés nécessaires, Amazon EMR sur EKS rejette la soumission de l'exécution de la tâche.

Exemple de politique IAM pour le rôle utilisé pour appeler StartJobRun

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Action": [
        "emr-containers:StartJobRun"
      ],
      "Resource": [
        "*"
      ],
      "Effect": "Allow",
      "Sid": "AllowEMRCONTAINERSStartjobrun"
    },
    {
      "Action": [
        "kms:DescribeKey",
        "kms:Decrypt",
        "kms:GenerateDataKey"
      ],
      "Resource": [
        "arn:aws:kms:*:*:key/key-id"
      ],
      "Effect": "Allow",
      "Sid": "AllowKMSDescribekey"
    }
  ]
```

```
}
```

Vous devez également configurer la clé KMS pour autoriser les `persistentappui.elasticmapreduce.amazonaws.com` et les principaux `elasticmapreduce.amazonaws.com` de service à accéder à `kms:GenerateDataKey` et`kms:Decrypt`. Cela permet à EMR de lire et d'écrire des journaux chiffrés à l'aide de la clé KMS pour accéder au stockage géré.

Exemple de politique clé KMS

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:DescribeKey"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "StringLike": {
          "kms:viaService": "emr-containers.*.amazonaws.com"
        }
      },
      "Sid": "AllowKMSDescribekey"
    },
    {
      "Effect": "Allow",
      "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "StringLike": {
```

```
          "kms:viaService": "emr-containers.*.amazonaws.com",
          "kms:EncryptionContext:aws:emr-containers:virtualClusterId": "virtual
 cluster id"
        }
      },
      "Sid": "AllowKMSDecryptGenerate"
    },
    {
      "Effect": "Allow",
      "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "StringLike": {
          "kms:EncryptionContext:aws:emr-containers:virtualClusterId": "virtual
 cluster id"
        },
        "ArnLike": {
          "aws:SourceArn": "arn:aws:emr-containers:*:*:/
virtualclusters/virtual_cluster_id"
        }
      },
      "Sid": "AllowKMSDecryptService"
    }
  ]
}
```

Pour des raisons de sécurité, nous vous recommandons d'ajouter les `aws:SourceArn` conditions `kms:viaServicekms:EncryptionContext`, et. Ces conditions permettent de garantir que la clé est uniquement utilisée par Amazon EMR sur EKS et uniquement utilisée pour les journaux générés par des tâches exécutées dans un cluster virtuel spécifique.

Journalisation d'Amazon EMR sur les appels d'API EKS à l'aide de AWS CloudTrail

Amazon EMR on 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 EMR on EKS.

CloudTrail capture tous les appels d'API pour Amazon EMR sur EKS sous forme d'événements. Ces captures incluent les appels de la console Amazon EMR on EKS et les appels de code vers les opérations d'API Amazon EMR on EKS. Si vous créez un suivi, vous pouvez activer la diffusion continue d' CloudTrail événements vers un compartiment Amazon S3, y compris des événements pour Amazon EMR sur EKS. Si vous ne configurez pas de suivi, vous pouvez toujours consulter les événements les plus récents dans la CloudTrail console dans Historique des événements. À l'aide des informations collectées par CloudTrail, vous pouvez déterminer la demande envoyée à Amazon EMR sur EKS, l'adresse IP à partir de laquelle la demande a été faite, l'auteur de la demande, la date à laquelle elle a été faite, ainsi que des informations supplémentaires.

Pour en savoir plus CloudTrail, consultez le guide de AWS CloudTrail l'utilisateur.

Informations Amazon EMR sur EKS dans CloudTrail

CloudTrail est activé sur votre AWS compte lorsque vous le créez. Lorsqu'une activité se produit dans Amazon EMR on EKS, cette activité est enregistrée dans un CloudTrail événement avec d'autres événements de AWS service dans l'historique des événements. Vous pouvez consulter, rechercher et télécharger les événements récents dans votre AWS compte. Pour plus d'informations, consultez la section Affichage des événements à l'aide de l'historique des CloudTrail événements.

Pour un enregistrement continu des événements de votre AWS compte, y compris des événements relatifs à Amazon EMR sur EKS, créez un historique. Un suivi permet CloudTrail de fournir des fichiers journaux à un compartiment Amazon S3. Par défaut, lorsque vous créez un parcours dans la console, celui-ci s'applique à toutes les AWS régions. Le journal enregistre les événements de toutes les régions de la AWS partition et transmet les fichiers journaux au compartiment Amazon S3 que vous spécifiez. En outre, vous pouvez configurer d'autres AWS services pour analyser plus en détail les données d'événements collectées dans les CloudTrail journaux et agir en conséquence. Pour plus d’informations, consultez les ressources suivantes :

- Présentation de la création d’un journal de suivi
- CloudTrail services et intégrations pris en charge
- Configuration des notifications Amazon SNS pour CloudTrail
- Réception de fichiers CloudTrail journaux de plusieurs régions et réception de fichiers CloudTrail journaux de plusieurs comptes

Toutes les actions Amazon EMR sur EKS sont enregistrées CloudTrail et documentées dans la documentation de l'API Amazon EMR on EKS. Par exemple, les appels

au`CreateVirtualCluster`, `StartJobRun` et les `ListJobRuns` actions génèrent des entrées dans les fichiers CloudTrail journaux.

Chaque événement ou entrée de journal contient des informations sur la personne ayant initié la demande. Les informations relatives à l'identité permettent de déterminer les éléments suivants :

- Si la demande a été faite avec les informations d'identification de l'utilisateur root ou Gestion des identités et des accès AWS (IAM).
- Si la demande a été effectuée avec les informations d'identification de sécurité temporaires d'un rôle ou d'un utilisateur fédéré.
- Si la demande a été faite par un autre AWS service.

Pour plus d'informations, consultez l'élément Identité de l'CloudTrail utilisateur.

Présentation des entrées du fichier journal d'Amazon EMR on EKS

Un suivi est une configuration qui permet de transmettre des événements sous forme de fichiers journaux à un compartiment Amazon S3 que vous spécifiez. CloudTrail les fichiers journaux contiennent une ou plusieurs entrées de journal. Un événement représente une demande unique provenant de n'importe quelle source et inclut des informations sur l'action demandée, la date et l'heure de l'action, les paramètres de la demande, etc. CloudTrail les fichiers journaux ne constituent pas une trace ordonnée des appels d'API publics, ils n'apparaissent donc pas dans un ordre spécifique.

L'exemple suivant montre une entrée de CloudTrail journal illustrant l'`ListJobRuns`action.

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AIDACKCEVSQ6C2EXAMPLE:admin",
    "arn": "arn:aws:sts::012345678910:assumed-role/Admin/admin",
    "accountId": "012345678910",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE",
        "arn": "arn:aws:iam::012345678910:role/Admin",
```

```
        "accountId": "012345678910",
        "userName": "Admin"
      },
      "webIdFederationData": {},
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2020-11-04T21:49:36Z"
      }
    }
  },
  "eventTime": "2020-11-04T21:52:58Z",
  "eventSource": "emr-containers.amazonaws.com",
  "eventName": "ListJobRuns",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "203.0.113.1",
  "userAgent": "aws-cli/1.11.167 Python/2.7.10 Darwin/16.7.0 botocore/1.7.25",
  "requestParameters": {
    "virtualClusterId": "1K48XXXXXXHCB"
  },
  "responseElements": null,
  "requestID": "890b8639-e51f-11e7-b038-EXAMPLE",
  "eventID": "874f89fa-70fc-4798-bc00-EXAMPLE",
  "readOnly": true,
  "eventType": "AwsApiCall",
  "recipientAccountId": "012345678910"
}
```


Utilisation Amazon S3 Access Grants avec Amazon EMR sur EKS

Présentation de S3 Access Grants pour Amazon EMR sur EKS

Avec Amazon EMR 6.15.0 et versions ultérieures, Amazon S3 Access Grants fournit une solution de contrôle d’accès évolutive que vous pouvez utiliser pour augmenter l’accès à vos données Amazon S3 depuis Amazon EMR sur EKS. Si vos exigences en matière de données S3 nécessitent une configuration d’autorisations complexe ou importante, vous pouvez utiliser S3 Access Grants pour mettre à l’échelle les autorisations de données S3 pour les utilisateurs, les groupes, les rôles et les applications.

Utilisez S3 Access Grants pour augmenter l’accès aux données Amazon S3 au-delà des autorisations accordées par le rôle d’exécution ou les rôles IAM associés aux identités ayant accès à votre cluster Amazon EMR sur EKS.

Pour plus d'informations, voir la rubrique Gestion des accès avec S3 Access Grants pour Amazon EMR du Guide de gestion Amazon EMR et la rubrique Gestion des accès avec S3 Access Grants du Guide de l'utilisateur Amazon Simple Storage Service.

Cette page présente les conditions requises pour exécuter une tâche Spark dans Amazon EMR sur EKS avec l'intégration de S3 Access Grants. Avec Amazon EMR sur EKS, S3 Access Grants nécessite une instruction de politique IAM supplémentaire dans le rôle d'exécution de votre tâche, ainsi qu'une configuration de remplacement supplémentaire pour l'API `StartJobRun`. Pour savoir comment configurer S3 Access Grants avec d'autres déploiements Amazon EMR, consultez la documentation suivante :

- Utilisation de S3 Access Grants avec Amazon EMR
- Utilisation de S3 Access Grants avec EMR sans serveur

Lancement d'un cluster Amazon EMR sur EKS avec S3 Access Grants pour la gestion des données

Vous pouvez activer S3 Access Grants dans Amazon EMR sur EKS et lancer une tâche Spark. Lorsque votre application demande à accéder aux données S3, Amazon S3 fournit des informations d'identification temporaires limitées au compartiment, au préfixe ou à l'objet concerné.

1. Configurez un rôle d'exécution de tâches pour votre cluster Amazon EMR sur EKS. Ajoutez les autorisations IAM `s3:GetDataAccess` et `s3:GetAccessGrantsInstanceForPrefix` requises pour l'exécution des tâches Spark :

```
{
    "Effect": "Allow",
    "Action": [
    "s3:GetDataAccess",
    "s3:GetAccessGrantsInstanceForPrefix"
    ],
    "Resource": [     //LIST ALL INSTANCE ARNS THAT THE ROLE IS ALLOWED TO QUERY
         "arn:aws_partition:s3:Region:account-id1:access-grants/default",
         "arn:aws_partition:s3:Region:account-id2:access-grants/default"
    ]
}
```

> **Note**
>
> Si vous spécifiez des rôles IAM qui, pour l'exécution des tâches, disposent d'autorisations supplémentaires pour accéder directement à S3, les utilisateurs peuvent être en mesure d'accéder aux données, quelles que soient les autorisations que vous définissez dans S3 Access Grants.

2. Soumettez une tâche à votre cluster Amazon EMR sur EKS avec une étiquette de version Amazon EMR 6.15 ou version ultérieure et la classification `emrfs-site`, comme illustré dans l'exemple suivant. Remplacez les valeurs dans *`red text`* par les valeurs appropriées pour votre scénario d'utilisation.

```
{
  "name": "myjob",
  "virtualClusterId": "123456",
  "executionRoleArn": "iam_role_name_for_job_execution",
  "releaseLabel": "emr-7.12.0-latest",
  "jobDriver": {
    "sparkSubmitJobDriver": {
      "entryPoint": "entryPoint_location",
      "entryPointArguments": ["argument1", "argument2"],
       "sparkSubmitParameters": "--class main_class"
    }
  },
  "configurationOverrides": {
    "applicationConfiguration": [
      {
        "classification": "emrfs-site",
        "properties": {
          "fs.s3.s3AccessGrants.enabled": "true",
          "fs.s3.s3AccessGrants.fallbackToIAM": "false"
         }
      }
    ],
  }
}
```

Considérations relatives à S3 Acces Grants avec Amazon EMR sur EKS

Pour obtenir des informations importantes sur la prise en charge, la compatibilité et le comportement lorsque vous utilisez Amazon S3 Access Grants avec Amazon EMR sur EKS, consultez la rubrique Considérations relatives à S3 Access Grants avec Amazon EMR du Guide de gestion d'Amazon EMR.

Validation de la conformité d'Amazon EMR on EKS

Des auditeurs tiers évaluent la sécurité et la conformité d'Amazon EMR sur EKS dans le cadre de plusieurs programmes de AWS conformité. Il s'agit notamment des certifications SOC, PCI, FedRAMP, HIPAA et d'autres.

Résilience dans Amazon EMR on EKS

L'infrastructure AWS mondiale est construite autour des AWS régions et des zones de disponibilité. AWS Les régions fournissent plusieurs zones de disponibilité physiquement séparées et isolées, connectées par un réseau à faible latence, à haut débit et hautement redondant. Avec les zones de disponibilité, vous pouvez concevoir et exploiter des applications et des bases de données qui basculent automatiquement d'une zone à l'autre sans interruption. Les zones de disponibilité sont davantage disponibles, tolérantes aux pannes et ont une plus grande capacité de mise à l'échelle que les infrastructures traditionnelles à un ou plusieurs centres de données.

Pour plus d'informations sur AWS les régions et les zones de disponibilité, consultez la section Infrastructure AWS mondiale.

Outre l'infrastructure AWS mondiale, Amazon EMR on EKS propose une intégration avec Amazon S3 via EMRFS pour répondre à vos besoins en matière de résilience des données et de sauvegarde.

Sécurité de l'infrastructure dans Amazon EMR on EKS

En tant que service géré, Amazon EMR est protégé par la sécurité du réseau AWS mondial. Pour plus d'informations sur les services AWS de sécurité et sur la manière dont AWS l'infrastructure est protégée, consultez la section Sécurité du AWS cloud. Pour concevoir votre AWS environnement en utilisant les meilleures pratiques en matière de sécurité de l'infrastructure, consultez la section Protection de l'infrastructure dans le cadre AWS bien architecturé du pilier de sécurité.

Vous utilisez des appels d'API AWS publiés pour accéder à Amazon EMR 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.

Analyse de la configuration et des vulnérabilités

AWS gère les tâches de sécurité de base telles que l'application de correctifs au système d'exploitation client (OS) et aux bases de données, la configuration du pare-feu et la reprise après sinistre. Ces procédures ont été vérifiées et certifiées par les tiers appropriés. Pour plus de détails, consultez les ressources suivantes :

- Validation de la conformité d'Amazon EMR on EKS
- Modèle de responsabilité partagée
- Amazon Web Services : Présentation des procédures de sécurité (livre blanc)

Connexion à Amazon EMR on EKS à l'aide du point de terminaison d'un VPC d'interface

Vous pouvez vous connecter directement à Amazon EMR sur EKS à l'aide des points de terminaison Interface VPC (AWS PrivateLink) de votre Virtual Private Cloud (VPC) au lieu de vous connecter via Internet. Lorsque vous utilisez un point de terminaison VPC d'interface, la communication entre votre VPC et Amazon EMR sur EKS s'effectue entièrement au sein du réseau. AWS Chaque point de terminaison VPC est représenté par une ou plusieurs interfaces réseau élastiques (ENIs) avec des adresses IP privées dans vos sous-réseaux VPC.

Le point de terminaison VPC de l'interface connecte votre VPC directement à Amazon EMR sur EKS sans passerelle Internet, périphérique NAT, connexion VPN ou connexion Direct Connect. AWS Les instances de votre VPC ne nécessitent pas d'adresses IP publiques pour communiquer avec l'API Amazon EMR on EKS.

Vous pouvez créer un point de terminaison VPC d'interface pour vous connecter à Amazon EMR sur EKS à l'aide des commandes AWS Management Console or AWS Command Line Interface ().AWS CLI Pour plus d’informations, consultez Création d’un point de terminaison d’interface.

Une fois que vous avez créé un point de terminaison d'un VPC d'interface, si vous activez les noms d'hôte DNS privés pour le point de terminaison, le point de terminaison Amazon EMR on EKS par défaut est résolu par votre point de terminaison de VPC. Le point de terminaison par défaut du nom de service Amazon EMR on EKS a le format suivant.

```
emr-containers.Region.amazonaws.com
```

Si vous n'activez pas les noms d'hôte DNS privés, Amazon VPC fournit un nom de point de terminaison DNS que vous pouvez utiliser au format suivant.

```
VPC_Endpoint_ID.emr-containers.Region.vpce.amazonaws.com
```

Pour plus d'informations, consultez Interface VPC Endpoints (AWS PrivateLink) dans le guide de l'utilisateur Amazon VPC. Amazon EMR on EKS prend en charge l'exécution d'appels en direction de toutes ses actions d'API à l'intérieur de votre VPC.

Vous pouvez attacher des politiques de point de terminaison de VPC au point de terminaison d'un VPC pour contrôler l'accès des principaux IAM. Vous pouvez également associer des groupes de sécurité à un point de terminaison VPC pour contrôler l'accès entrant et sortant en fonction de l'origine et de la destination du trafic réseau, comme une plage d'adresses IP. Pour plus d'informations, consultez Contrôle de l'accès aux services avec des points de terminaison de VPC.

Création d'une politique de point de terminaison d'un VPC pour Amazon EMR on EKS

Vous pouvez créer une politique pour les points de terminaison Amazon VPC pour Amazon EMR sur EKS afin de spécifier ce qui suit :

- Principal qui peut ou ne peut pas effectuer des actions
- Les actions qui peuvent être effectuées.
- Les ressources sur lesquelles les actions peuvent être exécutées.

Pour en savoir plus, consultez la rubrique Contrôle de l'accès aux services avec des points de terminaison d'un VPC du Guide de l'utilisateur Amazon VPC.

Example Politique de point de terminaison VPC visant à refuser tout accès à un compte spécifié AWS

La politique de point de terminaison VPC suivante refuse au AWS compte *123456789012* tout accès aux ressources utilisant le point de terminaison.

```
{
    "Statement": [
        {
            "Action": "*",
            "Effect": "Allow",
            "Resource": "*",
            "Principal": "*"
        },
        {
            "Action": "*",
            "Effect": "Deny",
            "Resource": "*",
            "Principal": {
                "AWS": [
                    "123456789012"
                ]
            }
        }
    ]
}
```

Example Politique du point de terminaison d'un VPC pour autoriser l'accès VPC uniquement à un principal (utilisateur) IAM spécifié

La politique de point de terminaison VPC suivante autorise un accès complet uniquement à l'utilisateur *lijuan* IAM inscrit dans le compte. AWS *123456789012* Toutes les autres entités IAM se voient refuser l'accès à l'aide du point de terminaison.

```
{
    "Statement": [
        {
            "Action": "*",
            "Effect": "Allow",
            "Resource": "*",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::123456789012:user/lijuan"
```

```
                ]
            }
        }
    ]
}
```

Example Politique du point de terminaison d'un VPC pour autoriser les opérations Amazon EMR on EKS en lecture seule

La politique de point de terminaison VPC suivante autorise uniquement le AWS compte *123456789012* à effectuer les actions Amazon EMR sur EKS spécifiées.

Les actions spécifiées fournissent l'équivalent d'un accès en lecture seule pour Amazon EMR on EKS. Toutes les autres actions sur le VPC sont refusées pour le compte spécifié. Tous les autres comptes se voient refuser tout accès. Pour obtenir une liste des actions d'Amazon EMR on EKS, consultez la rubrique Actions, ressources et clés de condition pour Amazon EMR on EKS.

```
{
    "Statement": [
        {
            "Action": [
                "emr-containers:DescribeJobRun",
                "emr-containers:DescribeVirtualCluster",
                "emr-containers:ListJobRuns",
                "emr-containers:ListTagsForResource",
                "emr-containers:ListVirtualClusters"
            ],
            "Effect": "Allow",
            "Resource": "*",
            "Principal": {
                "AWS": [
                    "123456789012"
                ]
            }
        }
    ]
}
```

Example Politique du point de terminaison d'un VPC refusant l'accès à un cluster virtuel spécifié

La politique de point de terminaison VPC suivante autorise un accès complet à tous les comptes et à tous les principaux, mais refuse tout accès du AWS compte *123456789012* aux actions effectuées

sur le cluster virtuel avec l'ID de cluster. *A1B2CD34EF5G* D'autres actions Amazon EMR on EKS qui ne prennent pas en charge les autorisations au niveau des ressources pour les clusters virtuels sont toujours autorisées. Pour obtenir une liste des actions Amazon EMR on EKS et de leur type de ressource correspondant, consultez la rubrique Actions, ressources et clés de condition pour Amazon EMR on EKS du Guide de l'utilisateur Gestion des identités et des accès AWS .

```
{
    "Statement": [
        {
            "Action": "*",
            "Effect": "Allow",
            "Resource": "*",
            "Principal": "*"
        },
        {
            "Action": "*",
            "Effect": "Deny",
            "Resource": "arn:aws:emr-containers:us-west-2:123456789012:/
virtualclusters/A1B2CD34EF5G",
            "Principal": {
                "AWS": [
                    "123456789012"
                ]
            }
        }
    ]
}
```

Configuration de l'accès intercompte pour Amazon EMR on EKS

Vous pouvez configurer l'accès intercompte pour Amazon EMR on EKS. L'accès entre comptes permet aux utilisateurs d'un AWS compte d'exécuter des tâches Amazon EMR sur EKS et d'accéder aux données sous-jacentes appartenant à AWS un autre compte.

Conditions préalables

Pour configurer l'accès entre comptes pour Amazon EMR sur EKS, vous devez effectuer des tâches en étant connecté aux AWS comptes suivants :

- `AccountA`- Un AWS compte sur lequel vous avez créé un cluster virtuel Amazon EMR sur EKS en enregistrant Amazon EMR avec un espace de noms sur un cluster EKS.

- `AccountB`- Un AWS compte contenant un compartiment Amazon S3 ou une table DynamoDB auxquels vous souhaitez que vos tâches Amazon EMR on EKS aient accès.

Vous devez disposer des éléments suivants dans vos AWS comptes avant de configurer l'accès entre comptes :

- Un cluster virtuel Amazon EMR on EKS dans le `AccountA` où vous souhaitez exécuter des tâches.
- Un rôle d'exécution de tâches dans le `AccountA` qui dispose des autorisations nécessaires pour exécuter des tâches dans le cluster virtuel. Pour plus d'informations, consultez Création d'un rôle d'exécution des tâches et Utilisation des rôles d'exécution de tâches avec Amazon EMR on EKS.

Procédure d'accès à un compartiment Amazon S3 ou à une table DynamoDB intercompte

Pour configurer l'accès intercompte pour Amazon EMR on EKS, procédez comme suit.

1. Créez un compartiment Amazon S3, `cross-account-bucket`, dans `AccountB`. Pour plus d'informations, consultez Créer un compartiment. Si vous souhaitez bénéficier d'un accès intercompte à DynamoDB, vous pouvez également créer une table DynamoDB dans `AccountB`. Pour plus d'informations, consultez Création d'une table DynamoDB.
2. Créez un rôle IAM `Cross-Account-Role-B` dans le `AccountB` qui peut accéder au `cross-account-bucket`.
    1. Connectez-vous à la console IAM.
    2. Choisissez Rôles et créez un nouveau rôle : `Cross-Account-Role-B`. Pour plus d'informations sur la création de rôles IAM, consultez Création de rôles IAM dans le Guide de l'utilisateur IAM.
    3. Créez une politique IAM qui spécifie les autorisations du `Cross-Account-Role-B` à accéder au compartiment S3 `cross-account-bucket`, comme le montre la déclaration de politique suivante. Attachez ensuite la politique IAM au `Cross-Account-Role-B`. Pour plus d'informations, consultez Création d'une nouvelle politique dans le Guide de l'utilisateur IAM.

        JSON

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

```
    {
      "Effect": "Allow",
      "Action": [
        "s3:*"
      ],
      "Resource": [
        "arn:aws:s3:::cross-account-bucket",
        "arn:aws:s3:::cross-account-bucket/*"
      ],
      "Sid": "AllowS3"
    }
  ]
}
```

Si l'accès à DynamoDB est nécessaire, créez une politique IAM qui spécifie les autorisations d'accès à la table DynamoDB intercompte. Attachez ensuite la politique IAM au `Cross-Account-Role-B`. Pour plus d'informations, consultez Création d'une table DynamoDB dans le Guide de l'utilisateur IAM.

Voici une politique d'accès à une table DynamoDB, `CrossAccountTable`.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:*"
      ],
      "Resource": [
        "arn:aws:dynamodb:us-east-1:*:table/CrossAccountTable"
      ],
      "Sid": "AllowDYNAMODB"
    }
  ]
}
```

3. Modifiez la relation de confiance du rôle `Cross-Account-Role-B`.

   1. Pour configurer la relation de confiance du rôle, choisissez l'onglet Relations de confiance dans la console IAM pour le rôle créé à l'étape 2 : `Cross-Account-Role-B`.

2. Sélectionnez Modifier la relation de confiance.
3. Ajoutez le document de politique suivant, qui permet à `Job-Execution-Role-A` dans `AccountA` d'assumer ce rôle `Cross-Account-Role-B`.

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sts:AssumeRole"
      ],
      "Resource": [
        "arn:aws:iam::*:role/Job-Execution-Role-A"
      ],
      "Sid": "AllowSTSAssumerole"
    }
  ]
}
```

4. Accordez à `Job-Execution-Role-A` dans `AccountA` l'autorisation d'assumer le `Cross-Account-Role-B` dans le cadre du rôle STS.

   1. Dans la console IAM du AWS compte`AccountA`, sélectionnez`Job-Execution-Role-A`.
   2. Ajoutez la déclaration de politique générale suivante au rôle `Job-Execution-Role-A` pour autoriser l'action `AssumeRole` sur le rôle `Cross-Account-Role-B`.

      JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sts:AssumeRole"
      ],
      "Resource": [
        "arn:aws:iam::*:role/Cross-Account-Role-B"
      ],
```

```
      "Sid": "AllowSTSAssumerole"
    }
  ]
}
```

5. Pour accéder à Amazon S3, définissez les paramètres `spark-submit` suivants (`spark conf`) lors de la soumission de la tâche à Amazon EMR on EKS.

> **Note**
>
> Par défaut, EMRFS utilise le rôle d'exécution de la tâche pour accéder au compartiment S3 à partir la tâche. Mais lorsque `customAWSCredentialsProvider` est défini sur `AssumeRoleAWSCredentialsProvider`, EMRFS utilise le rôle correspondant que vous spécifiez avec `ASSUME_ROLE_CREDENTIALS_ROLE_ARN` au lieu du rôle `Job-Execution-Role-A` pour l'accès à Amazon S3.

- `--conf spark.hadoop.fs.s3.customAWSCredentialsProvider=com.amazonaws.emr.AssumeRol`
- `--conf spark.kubernetes.driverEnv.ASSUME_ROLE_CREDENTIALS_ROLE_ARN=arn:aws:iam::Ac Cross-Account-Role-B \`
- `--conf spark.executorEnv.ASSUME_ROLE_CREDENTIALS_ROLE_ARN=arn:aws:iam::AccountB:ro Cross-Account-Role-B \`

> **Note**
>
> Vous devez définir `ASSUME_ROLE_CREDENTIALS_ROLE_ARN` à la fois pour l'exécuteur et le pilote `env` dans la configuration de la tâche Spark.

Pour l'accès intercompte DynamoDB, vous devez configurer `--conf spark.dynamodb.customAWSCredentialsProvider=com.amazonaws.emr.AssumeRoleAWSCr`

6. Exécutez la tâche Amazon EMR on EKS avec un accès intercompte, comme le montre l'exemple suivant.

```
aws emr-containers start-job-run \
--virtual-cluster-id 123456 \
--name myjob \
--execution-role-arn execution-role-arn \
--release-label emr-6.2.0-latest \
--job-driver '{"sparkSubmitJobDriver": {"entryPoint": "entryPoint_location",
 "entryPointArguments": ["arguments_list"], "sparkSubmitParameters": "--class
 <main_class> --conf spark.executor.instances=2 --conf spark.executor.memory=2G
 --conf spark.executor.cores=2 --conf spark.driver.cores=1 --conf
 spark.hadoop.fs.s3.customAWSCredentialsProvider=com.amazonaws.emr.AssumeRoleAWSCredentials
 --conf
 spark.kubernetes.driverEnv.ASSUME_ROLE_CREDENTIALS_ROLE_ARN=arn:aws:iam::AccountB:role/
Cross-Account-Role-B --conf
 spark.executorEnv.ASSUME_ROLE_CREDENTIALS_ROLE_ARN=arn:aws:iam::AccountB:role/
Cross-Account-Role-B"}} ' \
--configuration-overrides '{"applicationConfiguration": [{"classification":
 "spark-defaults", "properties": {"spark.driver.memory": "2G"}}],
 "monitoringConfiguration": {"cloudWatchMonitoringConfiguration":
 {"logGroupName": "log_group_name", "logStreamNamePrefix": "log_stream_prefix"},
 "persistentAppUI":"ENABLED",  "s3MonitoringConfiguration": {"logUri": "s3://
my_s3_log_location" }}}'
```

Balisage de vos ressources Amazon EMR on EKS

Pour vous aider à gérer vos ressources Amazon EMR on EKS, vous pouvez attribuer vos propres métadonnées à chaque ressource à l'aide de balises. Cette rubrique présente une vue d'ensemble de la fonction des balises et vous montre comment créer des balises.

Rubriques

Principes de base des étiquettes

Une étiquette est une étiquette que vous attribuez à une AWS ressource. Chaque balise est constituée d'une clé et d'une valeur facultative que vous définissez.

Les balises vous permettent de classer vos AWS ressources en fonction d'attributs tels que l'objectif, le propriétaire ou l'environnement. Lorsque vous avez de nombreuses ressources de même type, vous pouvez rapidement identifier une ressource spécifique en fonction des balises que vous lui avez attribuées. Par exemple, vous pouvez définir un ensemble de balises pour vos clusters Amazon EMR on 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.

Les balises ne sont pas automatiquement affectées à vos ressources. 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 balises n'ont pas de signification sémantique pour Amazon EMR on EKS et sont interprétées strictement comme des chaînes de caractères.

Une valeur de balise peut être une chaîne vide, mais pas null. Une clé de balise ne peut pas être une chaîne vide. 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 Gestion des identités et des accès AWS (IAM), vous pouvez contrôler quels utilisateurs de votre AWS compte sont autorisés à gérer les tags.

Pour des exemples de politique de contrôle d'accès basée sur les balises, consultez Politiques de contrôle d'accès basées sur les balises.

Baliser vos ressources

Vous pouvez baliser des clusters virtuels nouveaux ou existants et des exécutions de tâches qui sont dans des états actifs. Les états actifs des exécutions de tâches sont les suivants : `PENDING`, `SUBMITTED`, `RUNNING` et `CANCEL_PENDING`. Les états actifs des clusters virtuels sont les suivants : `RUNNING`, `TERMINATING` et `ARRESTED`. Pour plus d'informations, consultez États d'exécution de la tâche et États du cluster virtuel.

Lorsqu'un cluster virtuel est arrêté, les balises sont effacées et ne sont plus accessibles.

Si vous utilisez l'API Amazon EMR on EKS, le ou un AWS SDK AWS CLI, vous pouvez appliquer des balises aux nouvelles ressources à l'aide du paramètre tags de l'action d'API correspondante. Vous pouvez également appliquer des identifications aux ressources à l'aide de l'action d'API `TagResource`.

Vous pouvez utiliser certaines actions de création de ressources pour spécifier des balises pour une ressource lors de la création de cette dernière. Dans ce cas, si les balises ne peuvent pas être appliquées pendant la création de la ressource, cette dernière n'est pas créée. Ce mécanisme garantit que les ressources que vous vouliez étiqueter lors de la création sont créées avec des identifications spécifiées ou ne sont pas créées du tout. Si vous étiqueter des ressources au moment de la création, vous n'avez pas besoin d'exécuter de scripts d'étiquetage personnalisés après la création des ressources.

Le tableau suivant décrit les ressources Amazon EMR on EKS qui peuvent être balisées.

Ressource	Prend en charge les étiquettes	Prend en charge la propagation des étiquettes	Prend en charge le balisage lors de la création (Amazon EMR sur EKS API AWS CLI et SDK) AWS	API de création (des balises peuvent être ajoutées lors de la création)
Cluster virtuel	Oui	Non. Les balises associées à un cluster virtuel ne se propagent pas aux exécution s de tâches soumises à ce cluster virtuel.	Oui	CreateVir tualCluster
Job exécutés	Oui	Non	Oui	StartJobRun

Restrictions liées aux étiquettes

Les restrictions de base suivantes s'appliquent aux balises :

- Nombre maximal de balises par ressource : 50
- Pour chaque ressource, chaque clé de balise doit être unique, et chaque clé de balise peut avoir une seule valeur.
- Longueur de clé maximale : 128 caractères Unicode en UTF-8
- Longueur de valeur maximale : 256 caractères Unicode en UTF-8
- Si votre schéma de balisage est utilisé pour plusieurs AWS services et ressources, n'oubliez pas que d'autres services peuvent imposer des restrictions quant aux caractères autorisés. Les caractères généralement autorisés sont les lettres, les chiffres et les espaces représentables en UTF-8, ainsi que les caractères suivants : + - = . _ : / @.
- Les clés et les valeurs des balises distinguent les majuscules et minuscules.
- Une valeur de balise peut être une chaîne vide, mais pas null. Une clé de balise ne peut pas être une chaîne vide.

- N'utilisez pas `aws:`, `AWS:` ou n'importe quelle combinaison de majuscules ou minuscules de ce préfixe pour des clés ou des valeurs Ils sont réservés uniquement à AWS l'usage.

Travaillez avec des balises à l'aide de l'API Amazon EMR on EKS AWS CLI et de l'API Amazon EMR

Utilisez les AWS CLI commandes suivantes ou les opérations d'API Amazon EMR sur EKS pour ajouter, mettre à jour, répertorier et supprimer les balises de vos ressources.

Sous-tâche	AWS CLI	Action d'API
Ajouter ou remplacer une ou plusieurs balises	tag-resource	TagResource
Répertorie les balises d'une ressource.	list-tags-for-resource	ListTagsForResource
Supprimer une ou plusieurs balises.	untag-resource	UntagResource

Les exemples suivants montrent comment ajouter ou supprimer les étiquettes d'une ressource à l'aide de l' AWS CLI.

Exemple 1 : Baliser un cluster virtuel existant

La commande suivante permet de baliser un cluster virtuel existant.

```
aws emr-containers tag-resource --resource-arn resource_ARN --tags team=devs
```

Exemple 2 : Supprimer la balise d'un cluster virtuel existant

La commande suivante permet de supprimer une balise d'un cluster virtuel existant.

```
aws emr-containers untag-resource --resource-arn resource_ARN --tag-keys tag_key
```

Exemple 3 : Afficher la liste des balises d'une ressource

La commande suivante permet de répertorier l'ensemble des étiquettes associées à une ressource existante.

```
aws emr-containers list-tags-for-resource --resource-arn resource_ARN
```

Résolution des problèmes d'Amazon EMR on EKS

Cette section explique comment résoudre les problèmes liés à Amazon EMR on EKS. Pour plus d'informations sur la manière de résoudre les problèmes généraux liés à Amazon EMR, consultez la rubrique Résolution des problèmes liés à un cluster dans le Guide de gestion d'Amazon EMR.

Rubriques

Résolution des problèmes liés à l'utilisation de PersistentVolumeClaims (PVC)

Si vous devez créer, répertorier ou supprimer PersistentVolumeClaims (PVC) pour une tâche sans ajouter d'autorisations PVC au rôle Kubernetes par défaut emr-containers, la tâche échoue lorsque vous la soumettez. Sans ces autorisations, le rôle emr-containers ne peut pas créer les rôles nécessaires pour le pilote Spark ou le client Spark. Il ne suffit pas d'ajouter des autorisations aux rôles du pilote ou du client Spark, comme le suggèrent les messages d'erreur. Le rôle principal emr-containers doit également inclure les autorisations requises. Cette section explique comment ajouter les autorisations requises au rôle principal emr-containers.

Vérification

Pour vérifier si votre rôle emr-containers dispose des autorisations nécessaires, définissez la variable NAMESPACE avec votre propre valeur, puis exécutez la commande suivante :

```
export NAMESPACE=YOUR_VALUE
kubectl describe role emr-containers -n ${NAMESPACE}
```

En outre, pour vérifier si les rôles Spark et client disposent des autorisations nécessaires, exécutez la commande suivante :

```
kubectl describe role emr-containers-role-spark-driver -n ${NAMESPACE}
kubectl describe role emr-containers-role-spark-client -n ${NAMESPACE}
```

Si les autorisations ne sont pas disponibles, procédez au correctif comme suit.

Correctif

1. Si les tâches non autorisées sont en cours d'exécution, arrêtez-les.
2. Créez un fichier nommé RBAC_Patch.py comme suit :

```
import os
import subprocess as sp
import tempfile as temp
import json
import argparse
import uuid

def delete_if_exists(dictionary: dict, key: str):
    if dictionary.get(key, None) is not None:
        del dictionary[key]

def doTerminalCmd(cmd):
    with temp.TemporaryFile() as f:
        process = sp.Popen(cmd, stdout=f, stderr=f)
        process.wait()
        f.seek(0)
        msg = f.read().decode()
    return msg

def patchRole(roleName, namespace, extraRules, skipConfirmation=False):
    cmd = f"kubectl get role {roleName} -n {namespace} --output json".split(" ")
    msg = doTerminalCmd(cmd)
    if "(NotFound)" in msg and "Error" in msg:
        print(msg)
        return False
    role = json.loads(msg)
    rules = role["rules"]
    rulesToAssign = extraRules[::]
    passedRules = []
    for rule in rules:
        apiGroups = set(rule["apiGroups"])
        resources = set(rule["resources"])
        verbs = set(rule["verbs"])
        for extraRule in extraRules:
            passes = 0
            apiGroupsExtra = set(extraRule["apiGroups"])
```

```
            resourcesExtra = set(extraRule["resources"])
            verbsExtra = set(extraRule["verbs"])
            passes += len(apiGroupsExtra.intersection(apiGroups)) >=
len(apiGroupsExtra)
            passes += len(resourcesExtra.intersection(resources)) >=
len(resourcesExtra)
            passes += len(verbsExtra.intersection(verbs)) >= len(verbsExtra)
            if passes >= 3:
                if extraRule not in passedRules:
                    passedRules.append(extraRule)
                    if extraRule in rulesToAssign:
                        rulesToAssign.remove(extraRule)
                break
    prompt_text = "Apply Changes?"
    if len(rulesToAssign) == 0:
        print(f"The role {roleName} seems to already have the necessary
permissions!")
        prompt_text = "Proceed anyways?"
    for ruleToAssign in rulesToAssign:
        role["rules"].append(ruleToAssign)
    delete_if_exists(role, "creationTimestamp")
    delete_if_exists(role, "resourceVersion")
    delete_if_exists(role, "uid")
    new_role = json.dumps(role, indent=3)
    uid = uuid.uuid4()
    filename = f"Role-{roleName}-New_Permissions-{uid}-TemporaryFile.json"
    try:
        with open(filename, "w+") as f:
            f.write(new_role)
            f.flush()
        prompt = "y"
        if not skipConfirmation:
            prompt = input(
                doTerminalCmd(f"kubectl diff -f {filename}".split(" ")) +
f"\n{prompt_text} y/n: "
            ).lower().strip()
            while prompt != "y" and prompt != "n":
                prompt = input("Please make a valid selection. y/n:
").lower().strip()
        if prompt == "y":
            print(doTerminalCmd(f"kubectl apply -f {filename}".split(" ")))
    except Exception as e:
        print(e)
    os.remove(f"./{filename}")
```

```
if __name__ == '__main__':
    parser = argparse.ArgumentParser()
    parser.add_argument("-n", "--namespace",
                        help="Namespace of the Role. By default its the
 VirtualCluster's namespace",
                        required=True,
                        dest="namespace"
                        )

    parser.add_argument("-p", "--no-prompt",
                        help="Applies the patches without asking first",
                        dest="no_prompt",
                        default=False,
                        action="store_true"
                        )
    args = parser.parse_args()

    emrRoleRules = [
        {
            "apiGroups": [""],
            "resources": ["persistentvolumeclaims"],
            "verbs": ["list", "create", "delete", "patch"]
         }

    ]

    driverRoleRules = [
        {
            "apiGroups": [""],
            "resources": ["persistentvolumeclaims"],
            "verbs": ["list", "create", "delete", "patch", "deletecollection"]
        },
        {
            "apiGroups": [""],
            "resources": ["services"],
            "verbs": ["get", "list", "describe", "create", "delete", "watch",
 "deletecollection"]
        },
        {
            "apiGroups": [""],
            "resources": ["configmaps", "pods"],
            "verbs": ["deletecollection"]
        }
```

```
    ]

    clientRoleRules = [
        {
            "apiGroups": [""],
            "resources": ["persistentvolumeclaims"],
            "verbs": ["list", "create", "delete", "patch"]
        }
    ]

    patchRole("emr-containers", args.namespace, emrRoleRules, args.no_prompt)
    patchRole("emr-containers-role-spark-driver", args.namespace, driverRoleRules,
args.no_prompt)
    patchRole("emr-containers-role-spark-client", args.namespace, clientRoleRules,
args.no_prompt)
```

3. Exécutez le script Python :

```
python3 RBAC_Patch.py -n ${NAMESPACE}
```

4. Une différence kubectl entre les nouvelles autorisations et les anciennes apparaît. Appuyez sur y pour corriger le rôle.
5. Vérifiez les trois rôles dotés d'autorisations supplémentaires comme suit :

```
kubectl describe role -n ${NAMESPACE}
```

6. Exécutez le script Python :

```
python3 RBAC_Patch.py -n ${NAMESPACE}
```

7. Après avoir exécuté la commande, elle affichera une différence kubectl entre les nouvelles autorisations et les anciennes. Appuyez sur y pour corriger le rôle.
8. Vérifiez les trois rôles dotés d'autorisations supplémentaires :

```
kubectl describe role -n ${NAMESPACE}
```

9. Soumettez à nouveau la tâche.

Correctif manuel

Si l'autorisation requise par votre application s'applique à autre chose que les règles PVC, vous pouvez ajouter manuellement des autorisations Kubernetes pour votre cluster virtuel Amazon EMR selon vos besoins.

> **Note**
>
> Le rôle emr-containers est un rôle principal. Cela signifie qu'il doit fournir toutes les autorisations nécessaires avant que vous puissiez modifier vos rôles de pilote ou de client sous-jacents.

1. Téléchargez les autorisations actuelles dans les fichiers yaml en exécutant les commandes ci-dessous :

```
kubectl get role -n ${NAMESPACE} emr-containers -o yaml >> emr-containers-role-patch.yaml
kubectl get role -n ${NAMESPACE} emr-containers-role-spark-driver -o yaml >> driver-role-patch.yaml
kubectl get role -n ${NAMESPACE} emr-containers-role-spark-client -o yaml >> client-role-patch.yaml
```

2. En fonction de l'autorisation requise par votre application, modifiez chaque fichier et ajoutez des règles supplémentaires telles que les suivantes :

   - emr-containers-role-patch.yaml

```
- apiGroups:
  - ""
  resources:
  - persistentvolumeclaims
  verbs:
  - list
  - create
  - delete
  - patch
```

   - driver-role-patch.yaml

```
- apiGroups:
```

```
  - ""
  resources:
  - persistentvolumeclaims
  verbs:
  - list
  - create
  - delete
  - patch
  - deletecollection
- apiGroups:
  - ""
  resources:
  - services
  verbs:
  - get
  - list
  - describe
  - create
  - delete
  - watch
  - deletecollection
- apiGroups:
  - ""
  resources:
  - configmaps
  - pods
  verbs:
  - deletecollection
```

- client-role-patch.yaml

```
- apiGroups:
  - ""
  resources:
  - persistentvolumeclaims
  verbs:
  - list
  - create
  - delete
  - patch
```

3. Supprimez les attributs suivants avec leurs valeurs. Cela est nécessaire pour appliquer la mise à jour.

- creationTimestamp
- resourceVersion
- uid

4. Enfin, exécutez le correctif :

```
kubectl apply -f emr-containers-role-patch.yaml
kubectl apply -f driver-role-patch.yaml
kubectl apply -f client-role-patch.yaml
```

Résolution des problèmes de mise à l'échelle automatique verticale d'Amazon EMR on EKS

Consultez les sections suivantes si vous rencontrez des problèmes lors de la configuration de l'opérateur de mise à l'échelle automatique verticale d'Amazon EMR on EKS sur un cluster Amazon EKS avec Operator Lifecycle Manager. Pour plus d'informations, y compris les étapes à suivre pour finaliser l'installation, consultez Utilisation de la mise à l'échelle automatique verticale avec les tâches Spark sur Amazon EMR.

Erreur 403 : accès interdit

Si vous avez suivi les étapes de la rubrique Installation d'Operator Lifecycle Manager (OLM) sur votre cluster Amazon EKS, exécuté la commande `olm status` et reçu une erreur `403 Forbidden` comme celle ci-dessous, il se peut que vous n'ayez pas obtenu les jetons d'authentification pour le référentiel Amazon ECR de l'opérateur.

Pour résoudre ce problème, répétez l'étape de la rubrique Installation de l'opérateur de mise à l'échelle automatique verticale d'Amazon EMR on EKS pour obtenir les jetons. Réessayez ensuite l'installation.

```
Error: FATA[0002] Failed to run bundle: pull bundle image: error pulling image IMAGE.
error resolving name : unexpected status code [manifests latest]: 403 Forbidden
```

L'espace de noms Kubernetes est introuvable

Lorsque vous configurez l'opérateur de mise à l'échelle automatique verticale Amazon EMR on EKS sur un cluster Amazon EKS, une erreur `namespaces not found` comme celle illustrée ici peut s'afficher :

```
FATA[0020] Failed to run bundle: create catalog: error creating catalog source:
 namespaces "NAME" not found.
```

Si l'espace de noms que vous avez spécifié n'existe pas, OLM n'installera pas l'opérateur de mise à l'échelle automatique verticale. Pour résoudre ce problème, utilisez la commande suivante pour créer l'espace de noms. Réessayez ensuite l'installation.

```
kubectl create namespace NAME
```

Erreur lors de l'enregistrement des informations d'identification Docker

Pour configurer la mise à l'échelle automatique verticale, vous devez vous authentifier et récupérer vos images Docker d'Amazon EMR on EKS liées à la mise à l'échelle automatique verticale. Ce faisant, vous risquez de recevoir une erreur comme celle-ci si Docker n'est pas en cours d'exécution :

```
aws ecr get-login-password \
 --region $REGION | docker login \
 --username AWS \
 --password-stdin $ACCOUNT_ID.dkr.ecr.$REGION.amazonaws.com

Error saving credentials: error storing credentials - err: exit status 1
out: 'Post "http://ipc/registry/credstore-updated": dial unix backend.sock: connect: no
 such file or directory'
```

Pour résoudre ce problème, vérifiez que Docker est en cours d'exécution ou ouvrez Docker Desktop. Réessayez ensuite d'enregistrer vos informations d'identification.

Résolution des problèmes liés à l'opérateur Spark d'Amazon EMR on EKS

Consultez les sections suivantes si vous rencontrez des problèmes avec l'opérateur Spark d'Amazon EMR on EKS. Pour plus d'informations, y compris les étapes à suivre pour finaliser l'installation, consultez Exécution de tâches Spark à l'aide de l'opérateur Spark.

Erreur lors de l'installation des Charts de Helm

Si vous avez suivi les étapes de la rubrique Installation de l'opérateur Spark et que vous avez reçu une erreur `INSTALLATION FAILED` comme celle ci-dessous lorsque vous avez essayé d'installer ou de vérifier les Charts de Helm, il se peut que vous n'ayez pas obtenu les jetons d'authentification pour le référentiel Amazon ECR de l'opérateur.

Pour résoudre ce problème, répétez l'étape de la rubrique Installation de l'opérateur Spark pour authentifier votre client Helm dans le registre Amazon ECR. Réessayez ensuite l'étape d'installation.

```
Error: INSTALLATION FAILED: Kubernetes cluster unreachable: the server has asked for
 the client to provide credentials
```

UnsupportedFileSystemException: Non FileSystem pour le schéma « s3 »

Il se peut que vous rencontriez l'exception suivante dans le thread « main » :

```
org.apache.hadoop.fs.UnsupportedFileSystemException: No FileSystem for scheme "s3"
```

Dans ce cas, ajoutez les exceptions suivantes à la spécification `SparkApplication` :

```
 hadoopConf:
   # EMRFS filesystem
   fs.s3.customAWSCredentialsProvider:
 com.amazonaws.auth.WebIdentityTokenCredentialsProvider
   fs.s3.impl: com.amazon.ws.emr.hadoop.fs.EmrFileSystem
   fs.AbstractFileSystem.s3.impl: org.apache.hadoop.fs.s3.EMRFSDelegate
   fs.s3.buffer.dir: /mnt/s3
   fs.s3.getObject.initialSocketTimeoutMilliseconds: "2000"
   mapreduce.fileoutputcommitter.algorithm.version.emr_internal_use_only.EmrFileSystem:
 "2"
   mapreduce.fileoutputcommitter.cleanup-
failures.ignored.emr_internal_use_only.EmrFileSystem: "true"
 sparkConf:
   # Required for EMR Runtime
   spark.driver.extraClassPath: /usr/lib/hadoop-lzo/lib/*:/usr/lib/hadoop/hadoop-
aws.jar:/usr/share/aws/aws-java-sdk/*:/usr/share/aws/emr/emrfs/conf:/usr/share/aws/
emr/emrfs/lib/*:/usr/share/aws/emr/emrfs/auxlib/*:/usr/share/aws/emr/security/conf:/
usr/share/aws/emr/security/lib/*:/usr/share/aws/hmclient/lib/aws-glue-datacatalog-
spark-client.jar:/usr/share/java/Hive-JSON-Serde/hive-openx-serde.jar:/usr/share/aws/
sagemaker-spark-sdk/lib/sagemaker-spark-sdk.jar:/home/hadoop/extrajars/*
```

```
   spark.driver.extraLibraryPath: /usr/lib/hadoop/lib/native:/usr/lib/hadoop-lzo/lib/
native:/docker/usr/lib/hadoop/lib/native:/docker/usr/lib/hadoop-lzo/lib/native
   spark.executor.extraClassPath: /usr/lib/hadoop-lzo/lib/*:/usr/lib/hadoop/hadoop-
aws.jar:/usr/share/aws/aws-java-sdk/*:/usr/share/aws/emr/emrfs/conf:/usr/share/aws/
emr/emrfs/lib/*:/usr/share/aws/emr/emrfs/auxlib/*:/usr/share/aws/emr/security/conf:/
usr/share/aws/emr/security/lib/*:/usr/share/aws/hmclient/lib/aws-glue-datacatalog-
spark-client.jar:/usr/share/java/Hive-JSON-Serde/hive-openx-serde.jar:/usr/share/aws/
sagemaker-spark-sdk/lib/sagemaker-spark-sdk.jar:/home/hadoop/extrajars/*
   spark.executor.extraLibraryPath: /usr/lib/hadoop/lib/native:/usr/lib/hadoop-lzo/lib/
native:/docker/usr/lib/hadoop/lib/native:/docker/usr/lib/hadoop-lzo/lib/native
```

Points de terminaison et quotas de service Amazon EMR on EKS

Voici les points de terminaison et les Service Quotas pour Amazon EMR on EKS. Pour vous connecter par programmation à un AWS service, vous utilisez un point de terminaison. Outre les points de AWS terminaison standard, certains AWS services proposent des points de terminaison FIPS dans certaines régions. Pour plus d'informations, consultez Points de terminaison du service AWS. Service Quotas, également appelés limites, représentent le nombre maximal de ressources ou d'opérations de service pour votre compte AWS . Pour plus d'informations, consultez Quotas de service AWS.

Points de terminaison de service

Région AWS nom	Code	Endpoint	Protocole
USA Est (Virginie du Nord)	us-east-1	`emr-containers.us-east-1.amazonaws.com`	HTTPS
USA Est (Ohio)	us-east-2	`emr-containers.us-east-2.amazonaws.com`	HTTPS
USA Ouest (Californie du Nord)	us-west-1	`emr-containers.us-west-1.amazonaws.com`	HTTPS
USA Ouest (Oregon)	us-west-2	`emr-containers.us-west-2.amazonaws.com`	HTTPS
Asie Pacifique (Tokyo)	ap-northeast-1	`emr-containers.ap-northeast-1.amazonaws.com`	HTTPS
Asie-Pacifique (Séoul)	ap-northeast-2	`emr-containers.ap-northeast-2.amazonaws.com`	HTTPS
Asie-Pacifique (Osaka)	ap-northeast-3	`emr-containers.ap-northeast-3.amazonaws.com`	HTTPS

Région AWS nom	Code	Endpoint	Protocole
Asie-Pacifique (Mumbai)	ap-south-1	emr-containers.ap-south-1.amazonaws.com	HTTPS
Asie-Pacifique (Hyderabad)	ap-south-2	emr-containers.ap-south-2.amazonaws.com	HTTPS
Asie-Pacifique (Singapour)	ap-southeast-1	emr-containers.ap-southeast-1.amazonaws.com	HTTPS
Asie-Pacifique (Sydney)	ap-southeast-2	emr-containers.ap-southeast-2.amazonaws.com	HTTPS
Asie-Pacifique (Jakarta)	ap-southeast-3	emr-containers.ap-southeast-3.amazonaws.com	HTTPS
Asie-Pacifique (Hong Kong)	ap-east-1	emr-containers.ap-east-1.amazonaws.com	HTTPS
Afrique (Le Cap)	af-south-1	emr-containers.af-south-1.amazonaws.com	HTTPS
Canada (Centre)	ca-central-1	emr-containers.ca-central-1.amazonaws.com	HTTPS
Chine (Ningxia)	cn-northwest-1	emr-containers.cn-northwest-1.amazonaws.com.cn	HTTPS
Chine (Pékin)	cn-north-1	emr-containers.cn-north-1.amazonaws.com.cn	HTTPS
Europe (Francfort)	eu-central-1	emr-containers.eu-central-1.amazonaws.com	HTTPS
Europe (Zurich)	eu-central-2	emr-containers.eu-central-2.amazonaws.com	HTTPS

Région AWS nom	Code	Endpoint	Protocole
Europe (Irlande)	eu-west-1	`emr-containers.eu-west-1.amazonaws.com`	HTTPS
Europe (Londres)	eu-west-2	`emr-containers.eu-west-2.amazonaws.com`	HTTPS
Europe (Paris)	eu-west-3	`emr-containers.eu-west-3.amazonaws.com`	HTTPS
Europe (Stockholm)	eu-north-1	`emr-containers.eu-north-1.amazonaws.com`	HTTPS
Europe (Milan)	eu-south-1	`emr-containers.eu-south-1.amazonaws.com`	HTTPS
Europe (Espagne)	eu-south-2	`emr-containers.eu-south-2.amazonaws.com`	HTTPS
Israël (Tel Aviv)	il-central-1	`emr-containers.il-central-1.amazonaws.com`	HTTPS
Amérique du Sud (São Paulo)	sa-east-1	`emr-containers.sa-east-1.amazonaws.com`	HTTPS
Moyen-Orient (EAU)	me-central-1	`emr-containers.me-central-1.amazonaws.com`	HTTPS
Middle East (Bahrain)	me-south-1	`emr-containers.me-south-1.amazonaws.com`	HTTPS
AWS GovCloud (USA Est)	us-gov-east-1	`emr-containers.us-gov-east-1.amazonaws.com`	HTTPS
AWS GovCloud (US-Ouest)	us-gov-west-1	`emr-containers.us-gov-west-1.amazonaws.com`	HTTPS

Quotas de service

Amazon EMR on EKS limite les demandes d'API suivantes pour chaque AWS compte, région par région. Pour plus d'informations sur la façon dont la régulation est appliquée, consultez la section API Request Throttling dans le Amazon API Reference. EC2 Vous pouvez demander une augmentation des quotas de limitation des API pour votre AWS compte, en suivant le guide ci-dessous.

Action d'API	Capacité maximale du compartiment	Taux de remplissage du compartiment (par seconde)
CancelJobRun	25	1
CreateJobTemplate	25	1
CreateManagedEndpoint	25	1
CreateSecurityConfiguration	25	1
CreateVirtualCluster	25	1
DeleteJobTemplate	25	1
DeleteManagedEndpoint	25	1
DeleteVirtualCluster	25	1
DescribeJobRun	100	20
DescribeJobTemplate	25	1
DescribeManagedEndpoint	100	5
DescribeSecurityConfiguration	25	1
DescribeVirtualCluster	100	5
GetManagedEndpoint SessionCredentials	25	1
ListJobRuns	100	5

Action d'API	Capacité maximale du compartiment	Taux de remplissage du compartiment (par seconde)
ListJobTemplates	25	1
ListManagedEndpoints	25	1
ListSecurityConfigurations	25	1
ListVirtualClusters	100	5
StartJobRun	25	1
Throttle quota for all EMR on EKS API requests	200	20

Lorsque vous créez votre AWS compte, nous définissons des quotas par défaut (également appelés limites) pour vos AWS ressources par région. Si vous tentez de dépasser le quota d'une ressource, la demande échoue. Dans ce cas, vous pouvez réduire votre utilisation des ressources ou demander une augmentation du quota.

La console Service Quotas est un emplacement central où vous pouvez consulter et gérer vos quotas de AWS services, et demander des augmentations de quotas pour la plupart des ressources que vous utilisez. Utilisez les informations de quota fournies ici pour gérer votre AWS infrastructure. Prévoyez de demander les augmentations de quota avant le moment où vous en aurez besoin.

Afficher les quotas et demander des augmentations de quotas

Vous pouvez consulter vos quotas de service actuels pour chaque région à l'aide de la console Service Quotas. Pour obtenir des instructions détaillées, consultez la section Affichage des quotas de service dans le Guide de l'utilisateur des quotas de service.

Vous pouvez demander une augmentation de quota à partir de la console de AWS gestion ou à l'aide de la AWS CLI. Les étapes sont détaillées dans la section Demande d'augmentation de quota.

Versions Amazon EMR on EKS

Une version Amazon EMR est un ensemble d'applications open-source issues de l'écosystème big data. Chaque version comprend différentes applications, composants et fonctionnalités big data que vous choisissez de déployer et de configurer avec Amazon EMR on EKS lorsque vous exécutez votre tâche.

À partir des versions 5.32.0 et 6.2.0 d'Amazon EMR, vous pouvez déployer Amazon EMR on EKS. Cette option de déploiement n'est pas disponible avec les versions antérieures d'Amazon EMR. Vous devez spécifier une version prise en charge lorsque vous soumettez votre tâche.

Amazon EMR on EKS utilise le type d'étiquette de version suivant : `emr-x.x.x-latest` ou `emr-x.x.x-yyyymmdd` avec une date de version spécifique. Par exemple, `emr-7.12.0-latest` ou `emr-7.12.0-20210129`. Lorsque vous utilisez le suffixe `-latest`, vous vous assurez que votre version Amazon EMR inclut toujours les dernières mises à jour de sécurité.

> **Note**
>
> Pour une comparaison entre Amazon EMR sur EKS et Amazon EMR exécuté sur EKS, EC2 consultez l'Amazon EMR sur le site Web. FAQs AWS

Rubriques

- Versions 7.0.0 d'Amazon EMR sur EKS
- Versions 6.15.0 d'Amazon EMR sur EKS
- Versions 6.14.0 d'Amazon EMR sur EKS
- Versions 6.13.0 d'Amazon EMR on EKS
- Versions 6.12.0 d'Amazon EMR on EKS
- Versions 6.11.0 d'Amazon EMR on EKS
- Versions 6.10.0 d'Amazon EMR on EKS
- Versions 6.9.0 d'Amazon EMR on EKS
- Versions 6.8.0 d'Amazon EMR on EKS
- Versions 6.7.0 d'Amazon EMR on EKS
- Versions 6.6.0 d'Amazon EMR on EKS
- Versions 6.5.0 d'Amazon EMR on EKS
- Versions 6.4.0 d'Amazon EMR on EKS
- Versions 6.3.0 d'Amazon EMR on EKS
- Versions 6.2.0 d'Amazon EMR on EKS
- Versions 5.36.0 d'Amazon EMR on EKS
- Versions 5.35.0 d'Amazon EMR on EKS
- Versions 5.34.0 d'Amazon EMR on EKS
- Versions 5.33.0 d'Amazon EMR on EKS
- Versions 5.32.0 d'Amazon EMR on EKS

Versions d'Amazon EMR on EKS 7.12.0

Cette page décrit les fonctionnalités nouvelles et mises à jour d'Amazon EMR spécifiques au déploiement d'Amazon EMR on EKS. Pour en savoir plus sur Amazon EMR exécuté sur Amazon EC2 et sur la version 7.12.0 d'Amazon EMR en général, consultez Amazon EMR 7.12.0 dans le guide de mise à jour d'Amazon EMR.

Amazon EMR sur les versions 7.12 d'EKS

Les versions 7.12.0 d'Amazon EMR suivantes sont disponibles pour Amazon EMR sur EKS. Sélectionnez une version EMR-7.12.0-xxxx spécifique pour afficher plus de détails, tels que la balise d'image du conteneur associée.

Flink releases

Les versions 7.12.0 d'Amazon EMR suivantes sont disponibles pour Amazon EMR sur EKS lorsque vous exécutez des applications Flink.

- emr-7.12.0-flink-latest
- emr-7.12.0-flink-20251111

Spark releases

Les versions 7.12.0 d'Amazon EMR suivantes sont disponibles pour Amazon EMR sur EKS lorsque vous exécutez des applications Spark.

- emr-7.12.0-dernier
- emr-7.12.0-20251111
- emr-7.12.0-spark-rapids-latest
- emr-7.12.0-spark-rapids-20251111
- emr-7.12.0-java11-latest
- emr-7.12.0-java11-20251111
- emr-7.12.0-java8-latest
- emr-7.12.0-java8-20251111
- emr-7.12.0-spark-rapids-java8-latest
- emr-7.12.0-spark-rapids-java8-20251111
- notebook-spark/emr-7.12.0-latest
- notebook-spark/emr-7.12.0-20251111
- notebook-spark/emr-7.12.0-spark-rapids-latest
- notebook-spark/emr-7.12.0-spark-rapids-20251111
- notebook-spark/emr-7.12.0-java11-latest
- notebook-spark/emr-7.12.0-java11-20251111
- notebook-spark/emr-7.12.0-java8-latest
- notebook-spark/emr-7.12.0-java8-20251111
- notebook-spark/emr-7.12.0-spark-rapids-java8-latest
- notebook-spark/emr-7.12.0-spark-rapids-java8-20251111
- notebook-python/emr-7.12.0-latest

- notebook-python/emr-7.12.0-20251111
- notebook-python/emr-7.12.0-spark-rapids-latest
- notebook-python/emr-7.12.0-spark-rapids-20251111
- notebook-python/emr-7.12.0-java11-latest
- notebook-python/emr-7.12.0-java11-20251111
- notebook-python/emr-7.12.0-java8-latest
- notebook-python/emr-7.12.0-java8-20251111
- notebook-python/emr-7.12.0-spark-rapids-java8-latest
- notebook-python/emr-7.12.0-spark-rapids-java8-20251111
- livy/emr-7.12.0-latest
- livy/emr-7.12.0-20251111
- livy/emr-7.12.0-java11-latest
- livy/emr-7.12.0-java11-20251111
- livy/emr-7.12.0-java8-latest
- livy/emr-7.12.0-java8-20251111

Notes de mise à jour

Notes de mise à jour pour Amazon EMR sur EKS 7.12.0 :

- Applications prises en charge – AWS SDK pour Java 2.35.5 and 1.12.792, Apache Spark 3.5.6-amzn-1, Apache Hudi 1.0.2-amzn-1, Apache Iceberg 1.10.0-amzn-0, Delta 3.3.2-amzn-1, Apache Spark RAPIDS 25.04.0-amzn-0, Apache Flink 1.20.0-amzn-6
- Composants pris en charge - `emr-ddb` `emr-goodiesemr-s3-select`,,`emrfs`,`hadoop-client`,`hudi`,`hudi-spark`,`iceberg`,`spark-kubernetes`.
- Classifications de configuration prises en charge

  À utiliser avec StartJobRunet CreateManagedEndpoint APIs:

Classifications	Descriptions
`core-site`	Modifiez les valeurs dans le fichier Hadoop `core-site.xml` .

Classifications	Descriptions
`emrfs-site`	Modifiez les paramètres EMRFS.
`spark-metrics`	Modifiez les valeurs dans le fichier Spark `metrics.properties` .
`spark-defaults`	Modifiez les valeurs dans le fichier Spark `spark-defaults.conf` .
`spark-env`	Modifiez les valeurs dans l'environnement Spark.
`spark-hive-site`	Modifiez les valeurs dans le fichier Spark `hive-site.xml` .
`spark-log4j2`	Modifiez les valeurs dans le fichier Spark `log4j2.properties` .
`emr-job-submitter`	Configuration pour le pod soumissionnaire de tâches.

À utiliser spécifiquement avec CreateManagedEndpoint APIs:

Classifications	Descriptions
`jeg-config`	Modifiez les valeurs dans le fichier `jupyter_enterprise_gateway_config.py` Jupyter Enterprise Gateway.
`jupyter-kernel-overrides`	Modifiez la valeur de l'image du noyau dans le fichier Jupyter Kernel Spec.

Les classifications de configuration vous permettent de personnaliser les applications. Elles correspondent souvent à un fichier XML de configuration de l'application, tel que `spark-hive-site.xml`. Pour plus d'informations, consultez la rubrique Configuration des applications.

Changements et fonctionnalités

Les fonctionnalités suivantes sont incluses dans la version 7.12.0 d'Amazon EMR sur EKS :

- Vues matérialisées d'icebergs — À partir de la version 7.12.0 d'EMR, EMR Spark prend en charge la création et la gestion de vues matérialisées d'iceberg (MV).
- Accès complet aux tables à Hudi — Depuis EMR 7.12.0, EMR prend désormais en charge le contrôle FTA (Full Table Access) pour Apache Hudi dans Apache Spark en fonction de vos politiques définies dans Lake Formation. Cette fonctionnalité permet d'effectuer des opérations de lecture et d'écriture à partir de vos tâches Amazon EMR Spark sur les tables enregistrées de Lake Formation lorsque le rôle de tâche dispose d'un accès complet aux tables.
- Mise à niveau de la version Iceberg — EMR 7.12.0 prend en charge la version 1.10 d'Apache Iceberg.
- Journalisation des charges de travail interactives de Livy : à partir de la version 7.12.0 d'EMR, EMR prend en charge une journalisation complète des principaux composants du système afin d'améliorer le dépannage en cas d'échec des tâches de Livy Spark. Cette fonctionnalité permettra au service EMR d'accéder à des Livy et à des SecretAgent journaux supplémentaires afin de simplifier le dépannage.

emr-7.12.0-dernier

Notes de mise à jour : `emr-7.12.0-latest` renvoie actuellement à `emr-7.12.0-20251111`.

Régions : `emr-7.12.0-latest` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-7.12.0:latest`

emr-7.12.0-20251111

Notes de publication : `emr-7.12.0-20251111` a été publié en novembre 2025. Il s'agit de la version initiale d'Amazon EMR 7.12.0 (Spark).

Régions : `emr-emr-7.12.0-20251111` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-7.12.0-20251111`

emr-7.12.0-flink-latest

Notes de publication : pointe `emr-7.12.0-flink-latest` actuellement vers `emr-7.12.0-flink-20251111`

Régions : `emr-7.12.0-flink-latest` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-7.12.0-flink:latest`

emr-7.12.0-flink-20251111

Notes de publication : `7.12.0-flink-20251111` a été publié en novembre 2025. Il s'agit de la version initiale d'Amazon EMR 7.12.0 (Flink).

Régions : `emr-7.12.0-flink-20251111` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-7.12.0-flink:20251111`

Versions d'Amazon EMR on EKS 7.11.0

Cette page décrit les fonctionnalités nouvelles et mises à jour d'Amazon EMR spécifiques au déploiement d'Amazon EMR on EKS. Pour en savoir plus sur Amazon EMR exécuté sur Amazon EC2 et sur la version 7.11.0 d'Amazon EMR en général, consultez Amazon EMR 7.11.0 dans le guide de mise à jour d'Amazon EMR.

Versions d'Amazon EMR on EKS 7.11

Les versions 7.11.0 d'Amazon EMR suivantes sont disponibles pour Amazon EMR sur EKS. Sélectionnez une version EMR-7.11.0-xxxx spécifique pour afficher plus de détails, tels que la balise d'image du conteneur associée.

Flink releases

Les versions 7.11.0 d'Amazon EMR suivantes sont disponibles pour Amazon EMR sur EKS lorsque vous exécutez des applications Flink.

- emr-7.11.0-flink-latest

- emr-7.11.0-flink-20251020

Spark releases

Les versions 7.11.0 d'Amazon EMR suivantes sont disponibles pour Amazon EMR sur EKS lorsque vous exécutez des applications Spark.

- emr-7.11.0-version la plus récente
- emr-7.11.0-20251020
- emr-7.11.0-spark-rapids-latest
- emr-7.11.0-spark-rapids-20251020
- emr-7.11.0-java11-latest
- emr-7.11.0-java11-20251020
- emr-7.11.0-java8-latest
- emr-7.11.0-java8-20251020
- emr-7.11.0-spark-rapids-java8-latest
- emr-7.11.0-spark-rapids-java8-20251020
- notebook-spark/emr-7.11.0-latest
- notebook-spark/emr-7.11.0-20251020
- notebook-spark/emr-7.11.0-spark-rapids-latest
- notebook-spark/emr-7.11.0-spark-rapids-20251020
- notebook-spark/emr-7.11.0-java11-latest
- notebook-spark/emr-7.11.0-java11-20251020
- notebook-spark/emr-7.11.0-java8-latest
- notebook-spark/emr-7.11.0-java8-20251020
- notebook-spark/emr-7.11.0-spark-rapids-java8-latest
- notebook-spark/emr-7.11.0-spark-rapids-java8-20251020
- notebook-python/emr-7.11.0-latest
- notebook-python/emr-7.11.0-20251020
- notebook-python/emr-7.11.0-spark-rapids-latest
- notebook-python/emr-7.11.0-spark-rapids-20251020
- notebook-python/emr-7.11.0-java11-latest

- notebook-python/emr-7.11.0-java11-20251020
- notebook-python/emr-7.11.0-java8-latest
- notebook-python/emr-7.11.0-java8-20251020
- notebook-python/emr-7.11.0-spark-rapids-java8-latest
- notebook-python/emr-7.11.0-spark-rapids-java8-20251020
- livy/emr-7.11.0-latest
- livy/emr-7.11.0-20251020
- livy/emr-7.11.0-java11-latest
- livy/emr-7.11.0-java11-20251020
- livy/emr-7.11.0-java8-latest
- livy/emr-7.11.0-java8-20251020

Notes de mise à jour

Notes de mise à jour pour Amazon EMR sur EKS 7.11.0 :

- Applications prises en charge – AWS SDK pour Java 2.35.5 and 1.12.792, Apache Spark 3.5.6-amzn-0, Apache Hudi 1.0.2-amzn-0, Apache Iceberg 1.9.1-amzn-0, Delta 3.3.2-amzn-0, Apache Spark RAPIDS 25.04.0-amzn-0, Apache Flink 1.20.0-amzn-5
- Composants pris en charge - `emr-ddb` `emr-goodies``emr-s3-select`,,`emrfs`,`hadoop-client`,`hudi`,`hudi-spark`,`iceberg`,`spark-kubernetes`.
- Classifications de configuration prises en charge

  À utiliser avec StartJobRunet CreateManagedEndpoint APIs:

Classifications	Descriptions
`core-site`	Modifiez les valeurs dans le fichier Hadoop `core-site.xml` .
`emrfs-site`	Modifiez les paramètres EMRFS.
`spark-metrics`	Modifiez les valeurs dans le fichier Spark `metrics.properties` .

Classifications	Descriptions
`spark-defaults`	Modifiez les valeurs dans le fichier Spark `spark-defaults.conf` .
`spark-env`	Modifiez les valeurs dans l'environnement Spark.
`spark-hive-site`	Modifiez les valeurs dans le fichier Spark `hive-site.xml` .
`spark-log4j2`	Modifiez les valeurs dans le fichier Spark `log4j2.properties` .
`emr-job-submitter`	Configuration pour le pod soumissionnaire de tâches.

À utiliser spécifiquement avec CreateManagedEndpoint APIs:

Classifications	Descriptions
`jeg-config`	Modifiez les valeurs dans le fichier `jupyter_enterprise_gateway_config.py` Jupyter Enterprise Gateway.
`jupyter-kernel-overrides`	Modifiez la valeur de l'image du noyau dans le fichier Jupyter Kernel Spec.

Les classifications de configuration vous permettent de personnaliser les applications. Elles correspondent souvent à un fichier XML de configuration de l'application, tel que `spark-hive-site.xml`. Pour plus d'informations, consultez la rubrique Configuration des applications.

Changements et fonctionnalités

Les modifications suivantes sont incluses dans la version 7.11.0 d'Amazon EMR sur EKS :

Amazon EMR on EKS prend désormais en charge l'intégration avec SageMaker Unified Studio via une solution Livy gérée comprenant :

- Sessions gérées : nouveau type de ressource de session interactive qui fournit un point de terminaison HTTPS personnalisé pour exécuter des sessions Spark sur votre cluster EKS via SageMaker Unified Studio
- Intégration de Lake Formation : prend en charge le contrôle d'accès aux données grâce à deux modes a) Contrôle d'accès détaillé b) Accès complet aux tables (mode de compatibilité)
- Gestion des identités : options d'authentification flexibles a) Contrôle d'accès basé sur les rôles IAM b) contrôle d'accès basé sur les rôles.
- Sessions d'arrière-plan utilisateur avec intégration : prend en charge les charges de travail Spark de longue durée pour qu'elles continuent de fonctionner même après que les utilisateurs se soient déconnectés d' SageMaker Unified Studio, prenant en charge les sessions d'une durée maximale de 90 jours

emr-7.11.0-version la plus récente

Notes de mise à jour : `emr-7.11.0-latest` renvoie actuellement à `emr-7.11.0-20251020`.

Régions : `emr-7.11.0-latest` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-7.11.0:latest`

emr-7.11.0-20251020

Notes de publication : `emr-7.11.0-20251020` a été publié en novembre 2025. Il s'agit de la version initiale d'Amazon EMR 7.11.0 (Spark).

Régions : `emr-emr-7.11.0-20251020` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-7.11.0-20251020`

emr-7.11.0-flink-latest

Notes de publication : pointe `emr-7.11.0-flink-latest` actuellement vers `emr-7.11.0-flink-20251020`

Régions : `emr-7.11.0-flink-latest` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-7.11.0-flink:latest`

emr-7.11.0-flink-20251020

Notes de publication : `7.11.0-flink-20251020` a été publié en novembre 2025. Il s'agit de la version initiale d'Amazon EMR 7.11.0 (Flink).

Régions : `emr-7.11.0-flink-20251020` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-7.11.0-flink:20251020`

Versions d'Amazon EMR on EKS 7.10.0

Cette page décrit les fonctionnalités nouvelles et mises à jour d'Amazon EMR spécifiques au déploiement d'Amazon EMR on EKS. Pour en savoir plus sur Amazon EMR exécuté sur Amazon EC2 et sur la version 7.10.0 d'Amazon EMR en général, consultez Amazon EMR 7.10.0 dans le guide de mise à jour d'Amazon EMR.

Amazon EMR sur EKS versions 7.10

Les versions 7.10.0 d'Amazon EMR suivantes sont disponibles pour Amazon EMR sur EKS. Sélectionnez une version EMR-7.10.0-xxxx spécifique pour afficher plus de détails, tels que la balise d'image du conteneur associée.

Flink releases

Les versions 7.10.0 d'Amazon EMR suivantes sont disponibles pour Amazon EMR sur EKS lorsque vous exécutez des applications Flink.

- emr-7.10.0-flink-latest

- emr-7.10.0-flink-20250801

Spark releases

Les versions 7.10.0 d'Amazon EMR suivantes sont disponibles pour Amazon EMR sur EKS lorsque vous exécutez des applications Spark.

- emr-7.10.0-version la plus récente
- emr-7.10.0-20250801
- emr-7.10.0-spark-rapids-latest
- emr-7.10.0-spark-rapids-20250801
- emr-7.10.0-java11-latest
- emr-7.10.0-java11-20250801
- emr-7.10.0-java8-latest
- emr-7.10.0-java8-20250801
- emr-7.10.0-spark-rapids-java8-latest
- emr-7.10.0-spark-rapids-java8-20250801
- notebook-spark/emr-7.10.0-latest
- notebook-spark/emr-7.10.0-20250801
- notebook-spark/emr-7.10.0-spark-rapids-latest
- notebook-spark/emr-7.10.0-spark-rapids-20250801
- notebook-spark/emr-7.10.0-java11-latest
- notebook-spark/emr-7.10.0-java11-20250801
- notebook-spark/emr-7.10.0-java8-latest
- notebook-spark/emr-7.10.0-java8-20250801
- notebook-spark/emr-7.10.0-spark-rapids-java8-latest
- notebook-spark/emr-7.10.0-spark-rapids-java8-20250801
- notebook-python/emr-7.10.0-latest
- notebook-python/emr-7.10.0-20250801
- notebook-python/emr-7.10.0-spark-rapids-latest
- notebook-python/emr-7.10.0-spark-rapids-20250801
- notebook-python/emr-7.10.0-java11-latest

- notebook-python/emr-7.10.0-java11-20250801
- notebook-python/emr-7.10.0-java8-latest
- notebook-python/emr-7.10.0-java8-20250801
- notebook-python/emr-7.10.0-spark-rapids-java8-latest
- notebook-python/emr-7.10.0-spark-rapids-java8-20250801
- livy/emr-7.10.0-latest
- livy/emr-7.10.0-20250801
- livy/emr-7.10.0-java11-latest
- livy/emr-7.10.0-java11-20250801
- livy/emr-7.10.0-java8-latest
- livy/emr-7.10.0-java8-20250801

Notes de mise à jour

Notes de mise à jour pour Amazon EMR sur EKS 7.10.0 :

- Applications prises en charge – AWS SDK pour Java 2.31.48 and 1.12.782, Apache Spark 3.5.5-amzn-1, Apache Hudi 0.15.0-amzn-7, Apache Iceberg 1.8.1-amzn-0, Delta 3.3.0-amzn-2, Apache Spark RAPIDS 25.04.0-amzn-0, Apache Flink 1.20.0-amzn-4, Flink Kubernetes Operator 1.10.0-amzn-4
- Composants pris en charge - `emr-ddb` `emr-goodies``emr-s3-select`,,`emrfs`,`hadoop-client`,`hudi`,`hudi-spark`,`iceberg`,`spark-kubernetes`.
- Classifications de configuration prises en charge

  À utiliser avec StartJobRunet CreateManagedEndpoint APIs:

Classifications	Descriptions
`core-site`	Modifiez les valeurs dans le fichier Hadoop `core-site.xml` .
`emrfs-site`	Modifiez les paramètres EMRFS.
`spark-metrics`	Modifiez les valeurs dans le fichier Spark `metrics.properties` .

Classifications	Descriptions
`spark-defaults`	Modifiez les valeurs dans le fichier Spark `spark-defaults.conf` .
`spark-env`	Modifiez les valeurs dans l'environnement Spark.
`spark-hive-site`	Modifiez les valeurs dans le fichier Spark `hive-site.xml` .
`spark-log4j2`	Modifiez les valeurs dans le fichier Spark `log4j2.properties` .
`emr-job-submitter`	Configuration pour le pod soumissionnaire de tâches.

À utiliser spécifiquement avec CreateManagedEndpoint APIs:

Classifications	Descriptions
`jeg-config`	Modifiez les valeurs dans le fichier `jupyter_enterprise_gateway_config.py` Jupyter Enterprise Gateway.
`jupyter-kernel-overrides`	Modifiez la valeur de l'image du noyau dans le fichier Jupyter Kernel Spec.

Les classifications de configuration vous permettent de personnaliser les applications. Elles correspondent souvent à un fichier XML de configuration de l'application, tel que `spark-hive-site.xml`. Pour plus d'informations, consultez la rubrique Configuration des applications.

Changements et fonctionnalités

Les fonctionnalités suivantes sont incluses dans la version 7.10.0 d'Amazon EMR sur EKS :

- Système de fichiers S3A — À partir de la version 7.10.0, le système de fichiers S3A a remplacé EMRFS en tant que connecteur EMR S3 par défaut. Pour plus d'informations, voir Système de fichiers EMR (EMRFS).

emr-7.10.0-version la plus récente

Notes de mise à jour : `emr-7.10.0-latest` renvoie actuellement à `emr-7.10.0-20250801`.

Régions : `emr-7.10.0-latest` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-7.10.0:latest`

emr-7.10.0-20250801

Notes de publication : `emr-7.10.0-20250801` a été publié en février 2025. Il s'agit de la version initiale d'Amazon EMR 7.10.0 (Spark).

Régions : `emr-emr-7.10.0-20250801` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-7.10.0-20250801`

emr-7.10.0-flink-latest

Notes de publication : pointe `emr-7.10.0-flink-latest` actuellement vers `emr-7.10.0-flink-20250801`

Régions : `emr-7.10.0-flink-latest` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-7.10.0-flink:latest`

emr-7.10.0-flink-20250801

Notes de publication : `7.10.0-flink-20250801` a été publié en février 2025. Il s'agit de la version initiale d'Amazon EMR 7.10.0 (Flink).

Régions : `emr-7.10.0-flink-20250801` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-7.10.0-flink:20250801`

Versions d'Amazon EMR on EKS 7.9.0

Cette page décrit les fonctionnalités nouvelles et mises à jour d'Amazon EMR spécifiques au déploiement d'Amazon EMR on EKS. Pour en savoir plus sur Amazon EMR exécuté sur Amazon EC2 et sur la version 7.9.0 d'Amazon EMR en général, consultez Amazon EMR 7.9.0 dans le guide de mise à jour d'Amazon EMR.

Amazon EMR sur EKS versions 7.9

Les versions 7.9.0 d'Amazon EMR suivantes sont disponibles pour Amazon EMR sur EKS. Sélectionnez une version EMR-7.9.0-xxxx spécifique pour afficher plus de détails, tels que la balise d'image du conteneur associée.

Flink releases

Les versions 7.9.0 d'Amazon EMR suivantes sont disponibles pour Amazon EMR sur EKS lorsque vous exécutez des applications Flink.

- emr-7.9.0-flink-latest
- emr-7.9.0-flink-20250425

Spark releases

Les versions 7.9.0 d'Amazon EMR suivantes sont disponibles pour Amazon EMR sur EKS lorsque vous exécutez des applications Spark.

- emr-7.9.0 - Dernière version
- emr-7.9.0-20250425
- emr-7.9.0-spark-rapids-latest
- emr-7.9.0-spark-rapids-20250425
- emr-7.9.0-java11-latest

- emr-7.9.0-java11-20250425
- emr-7.9.0-java8-latest
- emr-7.9.0-java8-20250425
- emr-7.9.0-spark-rapids-java8-latest
- emr-7.9.0-spark-rapids-java8-20250425
- notebook-spark/emr-7.9.0-latest
- notebook-spark/emr-7.9.0-20250425
- notebook-spark/emr-7.9.0-spark-rapids-latest
- notebook-spark/emr-7.9.0-spark-rapids-20250425
- notebook-spark/emr-7.9.0-java11-latest
- notebook-spark/emr-7.9.0-java11-20250425
- notebook-spark/emr-7.9.0-java8-latest
- notebook-spark/emr-7.9.0-java8-20250425
- notebook-spark/emr-7.9.0-spark-rapids-java8-latest
- notebook-spark/emr-7.9.0-spark-rapids-java8-20250425
- notebook-python/emr-7.9.0-latest
- notebook-python/emr-7.9.0-20250425
- notebook-python/emr-7.9.0-spark-rapids-latest
- notebook-python/emr-7.9.0-spark-rapids-20250425
- notebook-python/emr-7.9.0-java11-latest
- notebook-python/emr-7.9.0-java11-20250425
- notebook-python/emr-7.9.0-java8-latest
- notebook-python/emr-7.9.0-java8-20250425
- notebook-python/emr-7.9.0-spark-rapids-java8-latest
- notebook-python/emr-7.9.0-spark-rapids-java8-20250425
- livy/emr-7.9.0-latest
- livy/emr-7.9.0-20250425
- livy/emr-7.9.0-java11-latest
- livy/emr-7.9.0-java11-20250425

- livy/emr-7.9.0-java8-latest
- livy/emr-7.9.0-java8-20250425

Notes de mise à jour

Notes de mise à jour pour Amazon EMR sur EKS 7.9.0

- Applications prises en charge – AWS SDK pour Java 2.31.16 and 1.12.782, Apache Spark 3.5.5, Apache Hudi 0.15.0-amzn-6, Apache Iceberg 1.7.1-amzn-2, Delta 3.3.0-amzn-1, Apache Spark RAPIDS 25.02.1-amzn-0, Jupyter Enterprise Gateway 2.6.0, Apache Flink 1.20.0-amzn-3, Flink Operator 1.10.0-amzn-3
- Composants pris en charge - `emr-ddb` `emr-goodiesemr-s3-select`,,`emrfs`,`hadoop-client`,`hudi`,`hudi-spark`,`iceberg`,`spark-kubernetes`.
- Classifications de configuration prises en charge

  À utiliser avec StartJobRunet CreateManagedEndpoint APIs:

Classifications	Descriptions
`core-site`	Modifiez les valeurs dans le fichier Hadoop `core-site.xml` .
`emrfs-site`	Modifiez les paramètres EMRFS.
`spark-metrics`	Modifiez les valeurs dans le fichier Spark `metrics.properties` .
`spark-defaults`	Modifiez les valeurs dans le fichier Spark `spark-defaults.conf` .
`spark-env`	Modifiez les valeurs dans l'environnement Spark.
`spark-hive-site`	Modifiez les valeurs dans le fichier Spark `hive-site.xml` .
`spark-log4j2`	Modifiez les valeurs dans le fichier Spark `log4j2.properties` .

Classifications	Descriptions
`emr-job-submitter`	Configuration pour le pod soumissionnaire de tâches.

À utiliser spécifiquement avec CreateManagedEndpoint APIs:

Classifications	Descriptions
`jeg-config`	Modifiez les valeurs dans le fichier `jupyter_enterprise_gateway_config.py` Jupyter Enterprise Gateway.
`jupyter-kernel-overrides`	Modifiez la valeur de l'image du noyau dans le fichier Jupyter Kernel Spec.

Les classifications de configuration vous permettent de personnaliser les applications. Elles correspondent souvent à un fichier XML de configuration de l'application, tel que `spark-hive-site.xml`. Pour plus d'informations, consultez la rubrique Configuration des applications.

Modifications

Les modifications suivantes sont incluses dans la version 7.9.0 d'Amazon EMR sur EKS :

- Aucune modification pour la version.

emr-7.9.0 - Dernière version

Notes de mise à jour : `emr-7.9.0-latest` renvoie actuellement à `emr-7.9.0-20250425`.

Régions : `emr-7.9.0-latest` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-7.9.0:latest`

emr-7.9.0-20250425

Notes de publication : `emr-7.9.0-20250425` a été publié en février 2025. Il s'agit de la version initiale d'Amazon EMR 7.9.0 (Spark).

Régions : `emr-emr-7.9.0-20250425` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-7.9.0-20250425`

emr-7.9.0-flink-latest

Notes de publication : pointe `emr-7.9.0-flink-latest` actuellement vers `emr-7.9.0-flink-20250425`

Régions : `emr-7.9.0-flink-latest` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-7.9.0-flink:latest`

emr-7.9.0-flink-20250425

Notes de publication : `7.9.0-flink-20250425` a été publié en février 2025. Il s'agit de la version initiale d'Amazon EMR 7.9.0 (Flink).

Régions : `emr-7.9.0-flink-20250425` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-7.9.0-flink:20250425`

Versions d'Amazon EMR on EKS 7.8.0

Cette page décrit les fonctionnalités nouvelles et mises à jour d'Amazon EMR spécifiques au déploiement d'Amazon EMR on EKS. Pour en savoir plus sur Amazon EMR exécuté sur Amazon EC2 et sur la version 7.8.0 d'Amazon EMR en général, consultez Amazon EMR 7.8.0 dans le guide de mise à jour d'Amazon EMR.

Amazon EMR sur EKS versions 7.8

Les versions 7.8.0 d'Amazon EMR suivantes sont disponibles pour Amazon EMR sur EKS. Sélectionnez une version EMR-7.8.0-xxxx spécifique pour afficher plus de détails, tels que la balise d'image du conteneur associée.

Flink releases

Les versions 7.8.0 d'Amazon EMR suivantes sont disponibles pour Amazon EMR sur EKS lorsque vous exécutez des applications Flink.

- emr-7.8.0-flink-latest
- emr-7.8.0-flink-20250228

Spark releases

Les versions 7.8.0 d'Amazon EMR suivantes sont disponibles pour Amazon EMR sur EKS lorsque vous exécutez des applications Spark.

- emr-7.8.0 - Dernière version
- emr-7.8.0-20250228
- emr-7.8.0-spark-rapids-latest
- emr-7.8.0-spark-rapids-20250228
- emr-7.8.0-java11-latest
- emr-7.8.0-java11-20250228
- emr-7.8.0-java8-latest
- emr-7.8.0-java8-20250228
- emr-7.8.0-spark-rapids-java8-latest
- emr-7.8.0-spark-rapids-java8-20250228
- notebook-spark/emr-7.8.0-latest
- notebook-spark/emr-7.8.0-20250228
- notebook-spark/emr-7.8.0-spark-rapids-latest
- notebook-spark/emr-7.8.0-spark-rapids-20250228
- notebook-spark/emr-7.8.0-java11-latest
- notebook-spark/emr-7.8.0-java11-20250228

- notebook-spark/emr-7.8.0-java8-latest
- notebook-spark/emr-7.8.0-java8-20250228
- notebook-spark/emr-7.8.0-spark-rapids-java8-latest
- notebook-spark/emr-7.8.0-spark-rapids-java8-20250228
- notebook-python/emr-7.8.0-latest
- notebook-python/emr-7.8.0-20250228
- notebook-python/emr-7.8.0-spark-rapids-latest
- notebook-python/emr-7.8.0-spark-rapids-20250228
- notebook-python/emr-7.8.0-java11-latest
- notebook-python/emr-7.8.0-java11-20250228
- notebook-python/emr-7.8.0-java8-latest
- notebook-python/emr-7.8.0-java8-20250228
- notebook-python/emr-7.8.0-spark-rapids-java8-latest
- notebook-python/emr-7.8.0-spark-rapids-java8-20250228
- livy/emr-7.8.0-latest
- livy/emr-7.8.0-20250228
- livy/emr-7.8.0-java11-latest
- livy/emr-7.8.0-java11-20250228
- livy/emr-7.8.0-java8-latest
- livy/emr-7.8.0-java8-20250228

Notes de mise à jour

Notes de mise à jour pour Amazon EMR sur EKS 7.8.0

- Applications prises en charge – AWS SDK pour Java 2.29.52 and 1.12.780, Apache Spark 3.5.4, Apache Hudi 0.15.0-amzn-5, Apache Iceberg 1.7.1-amzn-1, Delta 3.3.0-amzn-0, Apache Spark RAPIDS 24.12.0-amzn-0, Jupyter Enterprise Gateway 2.6.0, Apache Flink 1.20.0-amzn-2, Flink Operator 1.10.0-amzn-2
- Composants pris en charge - `emr-ddb` `emr-goodies`,`emr-s3-select`,`emrfs`,`hadoop-client`,`hudi`,`hudi-spark`,`iceberg`,`spark-kubernetes`.
- Classifications de configuration prises en charge

À utiliser avec StartJobRunet CreateManagedEndpoint APIs:

Classifications	Descriptions
`core-site`	Modifiez les valeurs dans le fichier Hadoop `core-site.xml` .
`emrfs-site`	Modifiez les paramètres EMRFS.
`spark-metrics`	Modifiez les valeurs dans le fichier Spark `metrics.properties` .
`spark-defaults`	Modifiez les valeurs dans le fichier Spark `spark-defaults.conf` .
`spark-env`	Modifiez les valeurs dans l'environnement Spark.
`spark-hive-site`	Modifiez les valeurs dans le fichier Spark `hive-site.xml` .
`spark-log4j2`	Modifiez les valeurs dans le fichier Spark `log4j2.properties` .
`emr-job-submitter`	Configuration pour le pod soumissionnaire de tâches.

À utiliser spécifiquement avec CreateManagedEndpoint APIs:

Classifications	Descriptions
`jeg-config`	Modifiez les valeurs dans le fichier `jupyter_enterprise_gateway_config.py` Jupyter Enterprise Gateway.
`jupyter-kernel-overrides`	Modifiez la valeur de l'image du noyau dans le fichier Jupyter Kernel Spec.

Les classifications de configuration vous permettent de personnaliser les applications. Elles correspondent souvent à un fichier XML de configuration de l'application, tel que `spark-hive-site.xml`. Pour plus d'informations, consultez la rubrique Configuration des applications.

Modifications

Les modifications suivantes sont incluses dans la version 7.8.0 d'Amazon EMR sur EKS :

- Fonctionnalités FGAC natives, notamment :
  - Iceberg Support permet d'exécuter des tâches qui exécutent des actions sur des tables de formation autres que des lacs dans un cluster virtuel de contrôle d'accès (FGAC) détaillé. (Il existe une solution de repli vers IAM.)
  - Support de table S3
- Spark Connect

emr-7.8.0 - Dernière version

Notes de mise à jour : `emr-7.8.0-latest` renvoie actuellement à `emr-7.8.0-20250228`.

Régions : `emr-7.8.0-latest` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-7.8.0:latest`

emr-7.8.0-20250228

Notes de publication : `emr-7.8.0-20250228` a été publié en février 2025. Il s'agit de la version initiale d'Amazon EMR 7.8.0 (Spark).

Régions : `emr-emr-7.8.0-20250228` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-7.8.0-20250228`

emr-7.8.0-flink-latest

Notes de publication : pointe `emr-7.8.0-flink-latest` actuellement vers `emr-7.8.0-flink-20250228`

Régions : `emr-7.8.0-flink-latest` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-7.8.0-flink:latest`

emr-7.8.0-flink-20250228

Notes de publication : `7.8.0-flink-20250228` a été publié en février 2025. Il s'agit de la version initiale d'Amazon EMR 7.8.0 (Flink).

Régions : `emr-7.8.0-flink-20250228` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-7.8.0-flink:20250228`

Versions d'Amazon EMR on EKS 7.7.0

Cette page décrit les fonctionnalités nouvelles et mises à jour d'Amazon EMR spécifiques au déploiement d'Amazon EMR on EKS. Pour en savoir plus sur Amazon EMR exécuté sur Amazon EC2 et sur la version 7.7.0 d'Amazon EMR en général, consultez Amazon EMR 7.7.0 dans le guide de mise à jour d'Amazon EMR.

Amazon EMR sur EKS versions 7.7

Les versions 7.7.0 d'Amazon EMR suivantes sont disponibles pour Amazon EMR sur EKS. Sélectionnez une version EMR-7.7.0-xxxx spécifique pour afficher plus de détails, tels que la balise d'image du conteneur associée.

Flink releases

Les versions 7.7.0 d'Amazon EMR suivantes sont disponibles pour Amazon EMR sur EKS lorsque vous exécutez des applications Flink.

- emr-7.7.0-flink-latest

- emr-7.7.0-flink-20250131

Spark releases

Les versions 7.7.0 d'Amazon EMR suivantes sont disponibles pour Amazon EMR sur EKS lorsque vous exécutez des applications Spark.

- emr-7.7.0 - version la plus récente
- emr-7.7.0-20250131
- emr-7.7.0-spark-rapids-latest
- emr-7.7.0-spark-rapids-20250131
- emr-7.7.0-java11-latest
- emr-7.7.0-java11-20250131
- emr-7.7.0-java8-latest
- emr-7.7.0-java8-20250131
- emr-7.7.0-spark-rapids-java8-latest
- emr-7.7.0-spark-rapids-java8-20250131
- notebook-spark/emr-7.7.0-latest
- notebook-spark/emr-7.7.0-20250131
- notebook-spark/emr-7.7.0-spark-rapids-latest
- notebook-spark/emr-7.7.0-spark-rapids-20250131
- notebook-spark/emr-7.7.0-java11-latest
- notebook-spark/emr-7.7.0-java11-20250131
- notebook-spark/emr-7.7.0-java8-latest
- notebook-spark/emr-7.7.0-java8-20250131
- notebook-spark/emr-7.7.0-spark-rapids-java8-latest
- notebook-spark/emr-7.7.0-spark-rapids-java8-20250131
- notebook-python/emr-7.7.0-latest
- notebook-python/emr-7.7.0-20250131
- notebook-python/emr-7.7.0-spark-rapids-latest
- notebook-python/emr-7.7.0-spark-rapids-20250131
- notebook-python/emr-7.7.0-java11-latest

- notebook-python/emr-7.7.0-java11-20250131
- notebook-python/emr-7.7.0-java8-latest
- notebook-python/emr-7.7.0-java8-20250131
- notebook-python/emr-7.7.0-spark-rapids-java8-latest
- notebook-python/emr-7.7.0-spark-rapids-java8-20250131
- livy/emr-7.7.0-latest
- livy/emr-7.7.0-20250131
- livy/emr-7.7.0-java11-latest
- livy/emr-7.7.0-java11-20250131
- livy/emr-7.7.0-java8-latest
- livy/emr-7.7.0-java8-20250131

Notes de mise à jour

Notes de mise à jour pour Amazon EMR sur EKS 7.7.0

- Applications prises en charge – AWS SDK pour Java 2.29.25 and 1.12.779, Apache Spark 3.5.3-amzn-0, Apache Hudi 0.15.0-amzn-3, Apache Iceberg 1.6.1-amzn-2, Delta 3.2.1-amzn-1, Apache Spark RAPIDS 24.10.1-amzn-0, Jupyter Enterprise Gateway 2.6.0, Apache Flink 1.20.0-amzn-0, Flink Operator 1.10.0-amzn-0
- Composants pris en charge - `emr-ddb` `emr-goodies``emr-s3-select`,,`emrfs`,`hadoop-client`,`hudi`,`hudi-spark`,`iceberg`,`spark-kubernetes`.
- Classifications de configuration prises en charge

  À utiliser avec StartJobRunet CreateManagedEndpoint APIs:

Classifications	Descriptions
`core-site`	Modifiez les valeurs dans le fichier Hadoop `core-site.xml` .
`emrfs-site`	Modifiez les paramètres EMRFS.
`spark-metrics`	Modifiez les valeurs dans le fichier Spark `metrics.properties` .

Classifications	Descriptions
`spark-defaults`	Modifiez les valeurs dans le fichier Spark `spark-defaults.conf` .
`spark-env`	Modifiez les valeurs dans l'environnement Spark.
`spark-hive-site`	Modifiez les valeurs dans le fichier Spark `hive-site.xml` .
`spark-log4j2`	Modifiez les valeurs dans le fichier Spark `log4j2.properties` .
`emr-job-submitter`	Configuration pour le pod soumissionnaire de tâches.

À utiliser spécifiquement avec CreateManagedEndpoint APIs:

Classifications	Descriptions
`jeg-config`	Modifiez les valeurs dans le fichier `jupyter_enterprise_gateway_config.py` Jupyter Enterprise Gateway.
`jupyter-kernel-overrides`	Modifiez la valeur de l'image du noyau dans le fichier Jupyter Kernel Spec.

Les classifications de configuration vous permettent de personnaliser les applications. Elles correspondent souvent à un fichier XML de configuration de l'application, tel que `spark-hive-site.xml`. Pour plus d'informations, consultez la rubrique Configuration des applications.

Modifications

Les modifications suivantes sont incluses dans la version 7.7.0 d'Amazon EMR sur EKS :

- La version d'Iceberg utilisée depuis EMR 7.7.0 ne prend plus en charge Java 8. En outre, Iceberg est exclu des images Java 8 suivantes : `emr-7.7.0-java8-latest` et`emr-7.7.0-spark-rapids-java8-latest`.

emr-7.7.0 - version la plus récente

Notes de mise à jour : `emr-7.7.0-latest` renvoie actuellement à `emr-7.7.0-20250131`.

Régions : `emr-7.7.0-latest` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-7.7.0:latest`

emr-7.7.0-20250131

Notes de publication : `emr-7.7.0-20250131` a été publié en février 2025. Il s'agit de la version initiale d'Amazon EMR 7.7.0 (Spark).

Régions : `emr-emr-7.7.0-20250131` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-7.7.0-20250131`

emr-7.7.0-flink-latest

Notes de publication : pointe `emr-7.7.0-flink-latest` actuellement vers `emr-7.7.0-flink-20250131`

Régions : `emr-7.7.0-flink-latest` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-7.7.0-flink:latest`

emr-7.7.0-flink-20250131

Notes de publication : `7.7.0-flink-20250131` a été publié en février 2025. Il s'agit de la version initiale d'Amazon EMR 7.7.0 (Flink).

Régions : `emr-7.7.0-flink-20250131` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-7.7.0-flink:20250131`

Versions d'Amazon EMR on EKS 7.6.0

Cette page décrit les fonctionnalités nouvelles et mises à jour d'Amazon EMR spécifiques au déploiement d'Amazon EMR on EKS. Pour en savoir plus sur Amazon EMR exécuté sur Amazon EC2 et sur la version 7.6.0 d'Amazon EMR en général, consultez Amazon EMR 7.6.0 dans le guide de mise à jour d'Amazon EMR.

Amazon EMR sur les versions 7.6 d'EKS

Les versions 7.6.0 d'Amazon EMR suivantes sont disponibles pour Amazon EMR sur EKS. Sélectionnez une version EMR-7.6.0-xxxx spécifique pour afficher plus de détails, tels que la balise d'image du conteneur associée.

Flink releases

Les versions 7.6.0 d'Amazon EMR suivantes sont disponibles pour Amazon EMR sur EKS lorsque vous exécutez des applications Flink.

- emr-7.6.0-flink-latest
- emr-7.6.0-flink-20241213

Spark releases

Les versions 7.6.0 d'Amazon EMR suivantes sont disponibles pour Amazon EMR sur EKS lorsque vous exécutez des applications Spark.

- emr-7.6.0 - Dernière version
- emr-7.6.0-20241213
- emr-7.6.0-spark-rapids-latest
- emr-7.6.0-spark-rapids-20241213
- emr-7.6.0-java11-latest

- emr-7.6.0-java11-20241213
- emr-7.6.0-java8-latest
- emr-7.6.0-java8-20241213
- emr-7.6.0-spark-rapids-java8-latest
- emr-7.6.0-spark-rapids-java8-20241213
- notebook-spark/emr-7.6.0-latest
- notebook-spark/emr-7.6.0-20241213
- notebook-spark/emr-7.6.0-spark-rapids-latest
- notebook-spark/emr-7.6.0-spark-rapids-20241213
- notebook-spark/emr-7.6.0-java11-latest
- notebook-spark/emr-7.6.0-java11-20241213
- notebook-spark/emr-7.6.0-java8-latest
- notebook-spark/emr-7.6.0-java8-20241213
- notebook-spark/emr-7.6.0-spark-rapids-java8-latest
- notebook-spark/emr-7.6.0-spark-rapids-java8-20241213
- notebook-python/emr-7.6.0-latest
- notebook-python/emr-7.6.0-20241213
- notebook-python/emr-7.6.0-spark-rapids-latest
- notebook-python/emr-7.6.0-spark-rapids-20241213
- notebook-python/emr-7.6.0-java11-latest
- notebook-python/emr-7.6.0-java11-20241213
- notebook-python/emr-7.6.0-java8-latest
- notebook-python/emr-7.6.0-java8-20241213
- notebook-python/emr-7.6.0-spark-rapids-java8-latest
- notebook-python/emr-7.6.0-spark-rapids-java8-20241213
- livy/emr-7.6.0-latest
- livy/emr-7.6.0-20241213
- livy/emr-7.6.0-java11-latest
- livy/emr-7.6.0-java11-20241213

- livy/emr-7.6.0-java8-latest
- livy/emr-7.6.0-java8-20241213

Notes de mise à jour

Notes de mise à jour pour Amazon EMR sur EKS 7.6.0

- Applications prises en charge – AWS SDK pour Java 2.29.25 and 1.12.779, Apache Spark 3.5.3-amzn-0, Apache Hudi 0.15.0-amzn-3, Apache Iceberg 1.6.1-amzn-2, Delta 3.2.1-amzn-1, Apache Spark RAPIDS 24.10.1-amzn-0, Jupyter Enterprise Gateway 2.6.0, Apache Flink 1.20.0-amzn-0, Flink Operator 1.10.0-amzn-0
- Composants pris en charge : `aws-sagemaker-spark-sdk`, `emr-ddb`, `emr-goodies`, `emr-s3-select`, `emrfs`, `hadoop-client`, `hudi`, `hudi-spark`, `iceberg`, `spark-kubernetes`.
- Classifications de configuration prises en charge

  À utiliser avec StartJobRunet CreateManagedEndpoint APIs:

Classifications	Descriptions
`core-site`	Modifiez les valeurs dans le fichier Hadoop `core-site.xml` .
`emrfs-site`	Modifiez les paramètres EMRFS.
`spark-metrics`	Modifiez les valeurs dans le fichier Spark `metrics.properties` .
`spark-defaults`	Modifiez les valeurs dans le fichier Spark `spark-defaults.conf` .
`spark-env`	Modifiez les valeurs dans l'environnement Spark.
`spark-hive-site`	Modifiez les valeurs dans le fichier Spark `hive-site.xml` .
`spark-log4j2`	Modifiez les valeurs dans le fichier Spark `log4j2.properties` .

Classifications	Descriptions
`emr-job-submitter`	Configuration pour le pod soumissionnaire de tâches.

À utiliser spécifiquement avec CreateManagedEndpoint APIs:

Classifications	Descriptions
`jeg-config`	Modifiez les valeurs dans le fichier `jupyter_enterprise_gateway_config.py` Jupyter Enterprise Gateway.
`jupyter-kernel-overrides`	Modifiez la valeur de l'image du noyau dans le fichier Jupyter Kernel Spec.

Les classifications de configuration vous permettent de personnaliser les applications. Elles correspondent souvent à un fichier XML de configuration de l'application, tel que `spark-hive-site.xml`. Pour plus d'informations, consultez la rubrique Configuration des applications.

Fonctionnalités notables

Les fonctionnalités suivantes sont incluses dans la version 7.6.0 d'Amazon EMR sur EKS :

- Support de configuration de surveillance pour l'opérateur Apache Spark — La configuration de surveillance vous permet de configurer facilement l'archivage des journaux de votre application Spark et des journaux des opérateurs sur Amazon S3 ou Amazon CloudWatch. Vous pouvez choisir l'un ou les deux. Cela ajoute un sidecar d'agent de journalisation à vos modules d'opérateur, de pilote et d'exécuteur Spark, puis transmet les journaux de ces composants aux récepteurs que vous avez configurés. Pour plus d'informations, consultez Utilisation de la configuration de surveillance pour surveiller l'opérateur Spark Kubernetes et les tâches Spark.

Modifications

Les modifications suivantes sont incluses dans la version 7.6.0 d'Amazon EMR sur EKS :

- Aucune modification pour la version.

emr-7.6.0 - Dernière version

Notes de mise à jour : `emr-7.6.0-latest` renvoie actuellement à `emr-7.6.0-20241213`.

Régions : `emr-7.6.0-latest` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-7.6.0:latest`

emr-7.6.0-20241213

Notes de publication : `7.6.0-20241213` a été publié en janvier 2024. Il s'agit de la version initiale d'Amazon EMR 7.6.0 (Spark).

Régions : `emr-7.6.0-20241213` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-7.6.0:20241213`

emr-7.6.0-flink-latest

Notes de publication : pointe `emr-7.6.0-flink-latest` actuellement vers `emr-7.6.0-flink-20241213`

Régions : `emr-7.6.0-flink-latest` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-7.6.0-flink:latest`

emr-7.6.0-flink-20241213

Notes de publication : `7.6.0-flink-20241213` a été publié en janvier 2024. Il s'agit de la version initiale d'Amazon EMR 7.6.0 (Flink).

Régions : `emr-7.6.0-flink-20241213` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-7.6.0-flink:20241213`

Publications d'Amazon EMR on EKS 7.5.0

Cette page décrit les fonctionnalités nouvelles et mises à jour d'Amazon EMR spécifiques au déploiement d'Amazon EMR on EKS. Pour en savoir plus sur Amazon EMR exécuté sur Amazon EC2 et sur la version 7.5.0 d'Amazon EMR en général, consultez Amazon EMR 7.5.0 dans le guide de mise à jour d'Amazon EMR.

Amazon EMR sur EKS versions 7.5

Les versions 7.5.0 d'Amazon EMR suivantes sont disponibles pour Amazon EMR sur EKS. Sélectionnez une version EMR-7.5.0-xxxx spécifique pour afficher plus de détails, tels que la balise d'image du conteneur associée.

Notes de mise à jour

Notes de mise à jour pour Amazon EMR sur EKS 7.5.0

- Applications prises en charge – AWS SDK pour Java 2.28.8 and 1.12.772, Apache Spark 3.5.2-amzn-1, Apache Hudi 0.15.0-amzn-1, Apache Iceberg 1.6.1-amzn-0, Delta 3.2.0-amzn-1, Apache Spark RAPIDS 24.08.1-amzn-1, Jupyter Enterprise Gateway 2.6.0, Apache Flink 1.19.1-amzn-1, Flink Operator 1.9.0-amzn-0
- Composants pris en charge : `aws-sagemaker-spark-sdk`, `emr-ddb`, `emr-goodies`, `emr-s3-select`, `emrfs`, `hadoop-client`, `hudi`, `hudi-spark`, `iceberg`, `spark-kubernetes`.

Versions d'Amazon EMR on EKS 7.4.0

Cette page décrit les fonctionnalités nouvelles et mises à jour d'Amazon EMR spécifiques au déploiement d'Amazon EMR on EKS. Pour en savoir plus sur Amazon EMR exécuté sur Amazon EC2 et sur la version 7.4.0 d'Amazon EMR en général, consultez Amazon EMR 7.4.0 dans le guide de mise à jour d'Amazon EMR.

Amazon EMR sur EKS versions 7.4

Les versions 7.4.0 d'Amazon EMR suivantes sont disponibles pour Amazon EMR sur EKS. Sélectionnez une version EMR-7.4.0-xxxx spécifique pour afficher plus de détails, tels que la balise d'image du conteneur associée.

Notes de mise à jour

Notes de mise à jour pour Amazon EMR sur EKS 7.4.0

- Applications prises en charge – AWS SDK pour Java 2.25.70 and 1.12.772, Apache Spark 3.5.2-amzn-0, Apache Hudi 0.15.0-amzn-1, Apache Iceberg 1.6.1-amzn-0, Delta 3.2.0-amzn-1, Apache Spark RAPIDS 24.08.1-amzn-0, Jupyter Enterprise Gateway 2.6.0, Apache Flink 1.19.1-amzn-0, Flink Operator 1.9.0-amzn-1
- Composants pris en charge : `aws-sagemaker-spark-sdk`, `emr-ddb`, `emr-goodies`, `emr-s3-select`, `emrfs`, `hadoop-client`, `hudi`, `hudi-spark`, `iceberg`, `spark-kubernetes`.

Versions d'Amazon EMR on EKS 7.3.0

Cette page décrit les fonctionnalités nouvelles et mises à jour d'Amazon EMR spécifiques au déploiement d'Amazon EMR on EKS. Pour en savoir plus sur Amazon EMR exécuté sur Amazon EC2 et sur la version 7.3.0 d'Amazon EMR en général, consultez Amazon EMR 7.3.0 dans le guide de mise à jour d'Amazon EMR.

Amazon EMR sur EKS versions 7.3

Les versions 7.3.0 d'Amazon EMR suivantes sont disponibles pour Amazon EMR sur EKS. Sélectionnez une version EMR-7.3.0-xxxx spécifique pour afficher plus de détails, tels que la balise d'image du conteneur associée.

Flink releases

Les versions 7.3.0 d'Amazon EMR suivantes sont disponibles pour Amazon EMR sur EKS lorsque vous exécutez des applications Flink.

- emr-7.3.0-flink-latest
- emr-7.3.0-flink-29240920

Spark releases

Les versions 7.3.0 d'Amazon EMR suivantes sont disponibles pour Amazon EMR sur EKS lorsque vous exécutez des applications Spark.

- emr-7.3.0-dernier
- emr-7.3.0-20240920
- emr-7.3.0-spark-rapids-latest
- emr-7.3.0-spark-rapids-29240920
- emr-7.3.0-java11-latest
- emr-7.3.0-java11-29240920
- emr-7.3.0-java8-latest
- emr-7.3.0-java8-29240920
- emr-7.3.0-spark-rapids-java8-latest
- emr-7.3.0-spark-rapids-java8-29240920
- notebook-spark/emr-7.3.0-latest
- notebook-spark/emr-7.3.0-20240920
- notebook-spark/emr-7.3.0-spark-rapids-latest
- notebook-spark/emr-7.3.0-spark-rapids-29240920
- notebook-spark/emr-7.3.0-java11-latest
- notebook-spark/emr-7.3.0-java11-29240920
- notebook-spark/emr-7.3.0-java8-latest
- notebook-spark/emr-7.3.0-java8-29240920
- notebook-spark/emr-7.3.0-spark-rapids-java8-latest
- notebook-spark/emr-7.3.0-spark-rapids-java8-29240920
- notebook-python/emr-7.3.0-latest
- notebook-python/emr-7.3.0-20240920
- notebook-python/emr-7.3.0-spark-rapids-latest
- notebook-python/emr-7.3.0-spark-rapids-29240920
- notebook-python/emr-7.3.0-java11-latest
- notebook-python/emr-7.3.0-java11-29240920

- notebook-python/emr-7.3.0-java8-latest
- notebook-python/emr-7.3.0-java8-29240920
- notebook-python/emr-7.3.0-spark-rapids-java8-latest
- notebook-python/emr-7.3.0-spark-rapids-java8-29240920
- livy/emr-7.3.0-latest
- livy/emr-7.3.0-20240920
- livy/emr-7.3.0-java11-latest
- livy/emr-7.3.0-java11-29240920
- livy/emr-7.3.0-java8-latest
- livy/emr-7.3.0-java8-29240920

Notes de mise à jour

Notes de mise à jour pour Amazon EMR sur EKS 7.3.0

- Applications prises en charge – AWS SDK pour Java 2.25.70 and 1.12.747, Apache Spark 3.5.1-amzn-1, Apache Hudi 0.15.0-amzn-0, Apache Iceberg 1.5.2-amzn-0, Delta 3.2.0-amzn-0, Apache Spark RAPIDS 24.06.1-amzn-0, Jupyter Enterprise Gateway 2.6.0, Apache Flink 1.18.1-amzn-2, Flink Operator 1.9.0-amzn-0
- Composants pris en charge : `aws-sagemaker-spark-sdk`, `emr-ddb`, `emr-goodies`, `emr-s3-select`, `emrfs`, `hadoop-client`, `hudi`, `hudi-spark`, `iceberg`, `spark-kubernetes`.
- Classifications de configuration prises en charge

  À utiliser avec StartJobRunet CreateManagedEndpoint APIs:

Classifications	Descriptions
`core-site`	Modifiez les valeurs dans le fichier Hadoop `core-site.xml` .
`emrfs-site`	Modifiez les paramètres EMRFS.
`spark-metrics`	Modifiez les valeurs dans le fichier Spark `metrics.properties` .

Classifications	Descriptions
`spark-defaults`	Modifiez les valeurs dans le fichier Spark `spark-defaults.conf` .
`spark-env`	Modifiez les valeurs dans l'environnement Spark.
`spark-hive-site`	Modifiez les valeurs dans le fichier Spark `hive-site.xml` .
`spark-log4j2`	Modifiez les valeurs dans le fichier Spark `log4j2.properties` .
`emr-job-submitter`	Configuration pour le pod soumissionnaire de tâches.

À utiliser spécifiquement avec CreateManagedEndpoint APIs:

Classifications	Descriptions
`jeg-config`	Modifiez les valeurs dans le fichier `jupyter_enterprise_gateway_config.py` Jupyter Enterprise Gateway.
`jupyter-kernel-overrides`	Modifiez la valeur de l'image du noyau dans le fichier Jupyter Kernel Spec.

Les classifications de configuration vous permettent de personnaliser les applications. Elles correspondent souvent à un fichier XML de configuration de l'application, tel que `spark-hive-site.xml`. Pour plus d'informations, consultez la rubrique Configuration des applications.

Fonctionnalités notables

Les fonctionnalités suivantes sont incluses dans la version 7.3.0 d'Amazon EMR sur EKS.

- Mises à niveau des applications — Amazon EMR on EKS inclut désormais Flink Operator 1.9.0. Outre d'autres fonctionnalités, le Flink Kubernetes vous permet désormais de définir des quotas de processeur et de mémoire pour l'autoscaler.
- Support d'Apache Iceberg pour Apache Flink — Apache Iceberg est un format open source performant de grandes tables analytiques. À partir d'Amazon EMR 7.3.0, vous pouvez utiliser les tables Apache Iceberg lorsque vous exécutez Apache Flink sur Amazon EMR sur EKS. Pour plus d'informations, consultez le document Amazon EMR sur EKS utilisant Apache Iceberg avec Amazon EMR sur EKS.
- Support de Delta Lake pour Apache Flink — Delta Lake est un framework de couche de stockage pour les architectures Lakehouse généralement construites sur Amazon S3. Avec Amazon EMR 7.3.0 et versions ultérieures, vous pouvez utiliser des tables Delta lorsque vous exécutez Apache Flink sur Amazon EMR sur EKS. Pour plus d'informations, consultez la section Utilisation de Delta Lake avec Amazon EMR sur EKS.

Modifications

Les modifications suivantes sont incluses dans la version 7.3.0 d'Amazon EMR sur EKS.

- Avec Amazon EMR sur EKS 7.3.0 et versions ultérieures, Apache Flink utilise désormais le runtime Java 17 par défaut.

emr-7.3.0-dernier

Notes de mise à jour : `emr-7.3.0-latest` renvoie actuellement à `emr-7.3.0-20240920`.

Régions : `emr-7.3.0-latest` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-7.3.0:latest`

emr-7.3.0-20240920

Notes de version : la version `7.3.0-20240920` a été publiée en décembre 2023. Il s'agit de la version initiale d'Amazon EMR 7.3.0 (Spark).

Régions : `emr-7.3.0-20240920` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-7.3.0:29240920`

emr-7.3.0-flink-latest

Notes de mise à jour : `emr-7.3.0-flink-latest` renvoie actuellement à `emr-7.3.0-flink-29240920`.

Régions : `emr-7.3.0-flink-latest` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-7.3.0-flink:latest`

emr-7.3.0-flink-29240920

Notes de version : la version `7.3.0-flink-29240920` a été publiée en décembre 2023. Il s'agit de la version initiale d'Amazon EMR 7.3.0 (Flink).

Régions : `emr-7.3.0-flink-29240920` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-7.3.0-flink:29240920`

Amazon EMR on EKS versions 7.2.0

Cette page décrit les fonctionnalités nouvelles et mises à jour d'Amazon EMR spécifiques au déploiement d'Amazon EMR on EKS. Pour en savoir plus sur Amazon EMR exécuté sur Amazon EC2 et sur la version 7.2.0 d'Amazon EMR en général, consultez Amazon EMR 7.2.0 dans le guide de mise à jour d'Amazon EMR.

Amazon EMR sur EKS versions 7.2

Les versions 7.2.0 d'Amazon EMR suivantes sont disponibles pour Amazon EMR sur EKS. Sélectionnez une version EMR-7.2.0-xxxx spécifique pour afficher plus de détails, tels que la balise d'image du conteneur associée.

Flink releases

Les versions 7.2.0 d'Amazon EMR suivantes sont disponibles pour Amazon EMR sur EKS lorsque vous exécutez des applications Flink.

- emr-7.2.0-flink-latest
- emr-7.2.0-flink-20240610

Spark releases

Les versions 7.2.0 d'Amazon EMR suivantes sont disponibles pour Amazon EMR sur EKS lorsque vous exécutez des applications Spark.

- emr-7.2.0-dernier
- emr-7.2.0-20240610
- emr-7.2.0-spark-rapids-latest
- emr-7.2.0-spark-rapids-20240610
- emr-7.2.0-java11-latest
- emr-7.2.0-java11-20240610
- emr-7.2.0-java8-latest
- emr-7.2.0-java8-20240610
- emr-7.2.0-spark-rapids-java8-latest
- emr-7.2.0-spark-rapids-java8-20240610
- notebook-spark/emr-7.2.0-latest
- notebook-spark/emr-7.2.0-20240610
- notebook-spark/emr-7.2.0-spark-rapids-latest
- notebook-spark/emr-7.2.0-spark-rapids-20240610
- notebook-spark/emr-7.2.0-java11-latest
- notebook-spark/emr-7.2.0-java11-20240610
- notebook-spark/emr-7.2.0-java8-latest
- notebook-spark/emr-7.2.0-java8-20240610
- notebook-spark/emr-7.2.0-spark-rapids-java8-latest

- notebook-spark/emr-7.2.0-spark-rapids-java8-20240610
- notebook-python/emr-7.2.0-latest
- notebook-python/emr-7.2.0-20240610
- notebook-python/emr-7.2.0-spark-rapids-latest
- notebook-python/emr-7.2.0-spark-rapids-20240610
- notebook-python/emr-7.2.0-java11-latest
- notebook-python/emr-7.2.0-java11-20240610
- notebook-python/emr-7.2.0-java8-latest
- notebook-python/emr-7.2.0-java8-20240610
- notebook-python/emr-7.2.0-spark-rapids-java8-latest
- notebook-python/emr-7.2.0-spark-rapids-java8-20240610
- livy/emr-7.2.0-latest
- livy/emr-7.2.0-20240610
- livy/emr-7.2.0-java11-latest
- livy/emr-7.2.0-java11-20240610
- livy/emr-7.2.0-java8-latest
- livy/emr-7.2.0-java8-20240610

Notes de mise à jour

Notes de mise à jour pour Amazon EMR sur EKS 7.2.0

- Applications prises en charge – AWS SDK pour Java 2.23.18 and 1.12.705, Apache Spark 3.5.1-amzn-1, Apache Hudi 0.14.1-amzn-0, Apache Iceberg 1.5.0-amzn-0, Delta 3.1.0, Apache Spark RAPIDS 24.02.0-amzn-1, Jupyter Enterprise Gateway 2.6.0, Apache Flink 1.18.1-amzn-0, Flink Operator 1.8.0-amzn-1
- Composants pris en charge : `aws-sagemaker-spark-sdk`, `emr-ddb`, `emr-goodies`, `emr-s3-select`, `emrfs`, `hadoop-client`, `hudi`, `hudi-spark`, `iceberg`, `spark-kubernetes`.
- Classifications de configuration prises en charge

  À utiliser avec StartJobRunet CreateManagedEndpoint APIs:

Classifications	Descriptions
`core-site`	Modifiez les valeurs dans le fichier Hadoop `core-site.xml` .
`emrfs-site`	Modifiez les paramètres EMRFS.
`spark-metrics`	Modifiez les valeurs dans le fichier Spark `metrics.properties` .
`spark-defaults`	Modifiez les valeurs dans le fichier Spark `spark-defaults.conf` .
`spark-env`	Modifiez les valeurs dans l'environnement Spark.
`spark-hive-site`	Modifiez les valeurs dans le fichier Spark `hive-site.xml` .
`spark-log4j2`	Modifiez les valeurs dans le fichier Spark `log4j2.properties` .
`emr-job-submitter`	Configuration pour le pod soumissionnaire de tâches.

À utiliser spécifiquement avec CreateManagedEndpoint APIs:

Classifications	Descriptions
`jeg-config`	Modifiez les valeurs dans le fichier `jupyter_enterprise_gateway_config.py` Jupyter Enterprise Gateway.
`jupyter-kernel-overrides`	Modifiez la valeur de l'image du noyau dans le fichier Jupyter Kernel Spec.

Les classifications de configuration vous permettent de personnaliser les applications. Elles correspondent souvent à un fichier XML de configuration de l'application, tel que `spark-hive-site.xml`. Pour plus d'informations, consultez la rubrique Configuration des applications.

Fonctionnalités notables

Les fonctionnalités suivantes sont incluses dans la version 7.2.0 d'Amazon EMR sur EKS.

- Mises à niveau des applications — Les mises à niveau des applications Amazon EMR on EKS 7.2.0 incluent Spark 3.5.1, Flink 1.18.1 et Flink Operator 1.8.0.
- Mises à jour d'Autoscaler for Flink — La version 7.2.0 utilise la configuration open source `job.autoscaler.restart.time-tracking.enabled` pour permettre l'estimation du temps de redimensionnement, de sorte que vous n'avez plus à attribuer manuellement des valeurs empiriques à l'heure de redémarrage. Si vous exécutez la version 7.1.0 ou une version antérieure, vous pouvez toujours utiliser l'autoscaling d'Amazon EMR.
- Intégration d'Apache Hudi Apache Flink sur Amazon EMR sur EKS — Cette version ajoute une intégration entre Apache Hudi et Apache Flink, afin que vous puissiez utiliser l'opérateur Flink Kubernetes pour exécuter des tâches Hudi. Hudi vous permet d'utiliser des opérations au niveau des enregistrements que vous pouvez utiliser pour simplifier la gestion des données et le développement de pipelines de données.
- Intégration d'Amazon S3 Express One Zone à Amazon EMR sur EKS — Avec la version 7.2.0 ou ultérieure, vous pouvez télécharger des données dans la zone S3 Express One avec Amazon EMR sur EKS. S3 Express One Zone est une classe de stockage Amazon S3 à zone unique à hautes performances qui fournit un accès aux données constant à un chiffre en millisecondes pour la plupart des applications sensibles à la latence. À son lancement, S3 Express One Zone offre la latence la plus faible et les meilleures performances de stockage d’objets cloud dans Amazon S3.
- Support des configurations par défaut dans l'opérateur Spark — L'opérateur Spark sur Amazon EKS prend désormais en charge les mêmes configurations par défaut que le modèle d'exécution de la tâche de démarrage sur Amazon EMR sur EKS pour la version 7.2.0 et ultérieure.
  Cela signifie que les fonctionnalités telles qu'Amazon S3 et EMRFS ne nécessitent plus de configurations manuelles dans le fichier yaml.

emr-7.2.0-dernier

Notes de mise à jour : `emr-7.2.0-latest` renvoie actuellement à `emr-7.2.0-20240610`.

Régions : `emr-7.2.0-latest` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-7.2.0:latest`

emr-7.2.0-20240610

Notes de version : la version `7.2.0-20240610` a été publiée en décembre 2023. Il s'agit de la version initiale d'Amazon EMR 7.2.0 (Spark).

Régions : `emr-7.2.0-20240610` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-7.2.0:20240610`

emr-7.2.0-flink-latest

Notes de mise à jour : `emr-7.2.0-flink-latest` renvoie actuellement à `emr-7.2.0-flink-20240610`.

Régions : `emr-7.2.0-flink-latest` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-7.2.0-flink:latest`

emr-7.2.0-flink-20240610

Notes de version : la version `7.2.0-flink-20240610` a été publiée en décembre 2023. Il s'agit de la version initiale d'Amazon EMR 7.2.0 (Flink).

Régions : `emr-7.2.0-flink-20240610` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-7.2.0-flink:20240610`

Versions d'Amazon EMR on EKS 7.1.0

Cette page décrit les fonctionnalités nouvelles et mises à jour d'Amazon EMR spécifiques au déploiement d'Amazon EMR on EKS. Pour en savoir plus sur Amazon EMR exécuté sur Amazon EC2 et sur la version 7.1.0 d'Amazon EMR en général, consultez Amazon EMR 7.1.0 dans le guide de mise à jour d'Amazon EMR.

Amazon EMR sur EKS versions 7.1

Les versions 7.1.0 d'Amazon EMR suivantes sont disponibles pour Amazon EMR sur EKS. Sélectionnez une version EMR-7.1.0-xxxx spécifique pour afficher plus de détails, tels que la balise d'image du conteneur associée.

Flink releases

Les versions 7.1.0 d'Amazon EMR suivantes sont disponibles pour Amazon EMR sur EKS lorsque vous exécutez des applications Flink.

- emr-7.1.0-flink-latest
- emr-7.1.0-flink-20240321

Spark releases

Les versions 7.1.0 d'Amazon EMR suivantes sont disponibles pour Amazon EMR sur EKS lorsque vous exécutez des applications Spark.

- emr-7.1.0 - Dernière version
- emr-7.1.0-20240321
- emr-7.1.0-spark-rapids-latest
- emr-7.1.0-spark-rapids-20240321
- emr-7.1.0-java11-latest
- emr-7.1.0-java11-20240321
- emr-7.1.0-java8-latest
- emr-7.1.0-java8-20240321
- emr-7.1.0-spark-rapids-java8-latest

- emr-7.1.0-spark-rapids-java8-20240321
- notebook-spark/emr-7.1.0-latest
- notebook-spark/emr-7.1.0-20240321
- notebook-spark/emr-7.1.0-spark-rapids-latest
- notebook-spark/emr-7.1.0-spark-rapids-20240321
- notebook-spark/emr-7.1.0-java11-latest
- notebook-spark/emr-7.1.0-java11-20240321
- notebook-spark/emr-7.1.0-java8-latest
- notebook-spark/emr-7.1.0-java8-20240321
- notebook-spark/emr-7.1.0-spark-rapids-java8-latest
- notebook-spark/emr-7.1.0-spark-rapids-java8-20240321
- notebook-python/emr-7.1.0-latest
- notebook-python/emr-7.1.0-20240321
- notebook-python/emr-7.1.0-spark-rapids-latest
- notebook-python/emr-7.1.0-spark-rapids-20240321
- notebook-python/emr-7.1.0-java11-latest
- notebook-python/emr-7.1.0-java11-20240321
- notebook-python/emr-7.1.0-java8-latest
- notebook-python/emr-7.1.0-java8-20240321
- notebook-python/emr-7.1.0-spark-rapids-java8-latest
- notebook-python/emr-7.1.0-spark-rapids-java8-20240321
- livy/emr-7.1.0-latest
- livy/emr-7.1.0-20240321
- livy/emr-7.1.0-java11-latest
- livy/emr-7.1.0-java11-20240321
- livy/emr-7.1.0-java8-latest
- livy/emr-7.1.0-java8-20240321

Notes de mise à jour

Notes de mise à jour pour Amazon EMR sur EKS 7.1.0

- Applications prises en charge – AWS SDK pour Java 2.23.18 and 1.12.656, Apache Spark 3.5.0-amzn-1, Apache Hudi 0.14.1-amzn-0, Apache Iceberg 1.4.3-amzn-0, Delta 3.0.0, Apache Spark RAPIDS 23.10.0-amzn-1, Jupyter Enterprise Gateway 2.6.0, Apache Flink 1.18.1-amzn-0, Flink Operator 1.6.1-amzn-1
- Composants pris en charge : `aws-sagemaker-spark-sdk`, `emr-ddb`, `emr-goodies`, `emr-s3-select`, `emrfs`, `hadoop-client`, `hudi`, `hudi-spark`, `iceberg`, `spark-kubernetes`.
- Classifications de configuration prises en charge

  À utiliser avec StartJobRunet CreateManagedEndpoint APIs:

Classifications	Descriptions
`core-site`	Modifiez les valeurs dans le fichier Hadoop `core-site.xml` .
`emrfs-site`	Modifiez les paramètres EMRFS.
`spark-metrics`	Modifiez les valeurs dans le fichier Spark `metrics.properties` .
`spark-defaults`	Modifiez les valeurs dans le fichier Spark `spark-defaults.conf` .
`spark-env`	Modifiez les valeurs dans l'environnement Spark.
`spark-hive-site`	Modifiez les valeurs dans le fichier Spark `hive-site.xml` .
`spark-log4j2`	Modifiez les valeurs dans le fichier Spark `log4j2.properties` .
`emr-job-submitter`	Configuration pour le pod soumissionnaire de tâches.

  À utiliser spécifiquement avec CreateManagedEndpoint APIs:

Classifications	Descriptions
`jeg-config`	Modifiez les valeurs dans le fichier `jupyter_enterprise_gateway_config.py` Jupyter Enterprise Gateway.
`jupyter-kernel-overrides`	Modifiez la valeur de l'image du noyau dans le fichier Jupyter Kernel Spec.

Les classifications de configuration vous permettent de personnaliser les applications. Elles correspondent souvent à un fichier XML de configuration de l'application, tel que `spark-hive-site.xml`. Pour plus d'informations, consultez la rubrique Configuration des applications.

Fonctionnalités notables

Les fonctionnalités suivantes sont incluses dans la version 7.1.0 d'Amazon EMR sur EKS.

- Support d'Apache Livy pour Amazon EMR sur EKS — Avec Amazon EMR on EKS versions 7.1.0 et supérieures, vous pouvez utiliser Apache Livy sur un cluster Amazon EKS pour créer une interface REST Apache Livy afin de soumettre des tâches Spark ou des extraits de code Spark. Cela vous permet de récupérer les résultats de manière synchrone et asynchrone, tout en continuant à tirer parti des avantages d'Amazon EMR on EKS, tels que le runtime Spark optimisé pour Amazon EMR, les points de terminaison Livy compatibles SSL et une expérience de configuration programmatique.

emr-7.1.0 - Dernière version

Notes de mise à jour : `emr-7.1.0-latest` renvoie actuellement à `emr-7.1.0-20240321`.

Régions : `emr-7.1.0-latest` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-7.1.0:latest`

emr-7.1.0-20240321

Notes de version : la version `7.1.0-20240321` a été publiée en décembre 2023. Il s'agit de la version initiale d'Amazon EMR 7.1.0 (Spark).

Régions : `emr-7.1.0-20240321` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-7.1.0:20240321`

emr-7.1.0-flink-latest

Notes de mise à jour : `emr-7.1.0-flink-latest` renvoie actuellement à `emr-7.1.0-flink-20240321`.

Régions : `emr-7.1.0-flink-latest` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-7.1.0-flink:latest`

emr-7.1.0-flink-20240321

Notes de version : la version `7.1.0-flink-20240321` a été publiée en décembre 2023. Il s'agit de la version initiale d'Amazon EMR 7.1.0 (Flink).

Régions : `emr-7.1.0-flink-20240321` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-7.1.0-flink:20240321`

Versions 7.0.0 d’Amazon EMR sur EKS

Cette page décrit les fonctionnalités nouvelles et mises à jour d'Amazon EMR spécifiques au déploiement d'Amazon EMR on EKS. Pour en savoir plus sur Amazon EMR exécuté sur Amazon EC2 et sur la version 7.0.0 d'Amazon EMR en général, consultez Amazon EMR 7.0.0 dans le guide de mise à jour d'Amazon EMR.

Versions 7.0 d’Amazon EMR sur EKS

Les versions 7.0.0 suivantes d’Amazon EMR sont disponibles pour Amazon EMR sur EKS. Sélectionnez une version emr-7.0.0-XXXX spécifique pour afficher plus d’informations, comme la balise de l’image du conteneur correspondant.

Flink releases

Les versions 7.0.0 suivantes d’Amazon EMR sont disponibles pour Amazon EMR sur EKS lorsque vous exécutez des applications Flink.

- emr-7.0.0-flink-latest
- emr-7.0.0-flink-2024321
- emr-7.0.0-flink-20231211

Spark releases

Les versions 7.0.0 suivantes d’Amazon EMR sont disponibles pour Amazon EMR sur EKS lorsque vous exécutez des applications Spark.

- emr-7.0.0-latest
- emr-7.0.0-20231211
- emr-7.0.0-spark-rapids-latest
- emr-7.0.0-spark-rapids-20231211
- emr-7.0.0-java11-latest
- emr-7.0.0-java11-20231211
- emr-7.0.0-java8-latest
- emr-7.0.0-java8-20231211
- emr-7.0.0-spark-rapids-java8-latest
- emr-7.0.0-spark-rapids-java8-20231211
- notebook-spark/emr-7.0.0-latest
- notebook-spark/emr-7.0.0-20231211
- notebook-spark/emr-7.0.0-spark-rapids-latest
- notebook-spark/emr-7.0.0-spark-rapids-20231211
- notebook-spark/emr-7.0.0-java11-latest

- notebook-spark/emr-7.0.0-java11-20231211
- notebook-spark/emr-7.0.0-java8-latest
- notebook-spark/emr-7.0.0-java8-20231211
- notebook-spark/emr-7.0.0-spark-rapids-java8-latest
- notebook-spark/emr-7.0.0-spark-rapids-java8-20231211
- notebook-python/emr-7.0.0-latest
- notebook-python/emr-7.0.0-20231211
- notebook-python/emr-7.0.0-spark-rapids-latest
- notebook-python/emr-7.0.0-spark-rapids-20231211
- notebook-python/emr-7.0.0-java11-latest
- notebook-python/emr-7.0.0-java11-20231211
- notebook-python/emr-7.0.0-java8-latest
- notebook-python/emr-7.0.0-java8-20231211
- notebook-python/emr-7.0.0-spark-rapids-java8-latest
- notebook-python/emr-7.0.0-spark-rapids-java8-20231211

Notes de mise à jour

Notes de mise à jour pour Amazon EMR sur EKS 7.0.0

- Applications prises en charge – AWS SDK pour Java 2.20.160-amzn-0 and 1.12.595, Apache Spark 3.5.0-amzn-0, Apache Flink 1.18.0-amzn-0, Flink Operator 1.6.1, Apache Hudi 0.14.0-amzn-1, Apache Iceberg 1.4.2-amzn-0, Delta 3.0.0, Apache Spark RAPIDS 23.10.0-amzn-0, Jupyter Enterprise Gateway 2.6.0
- Composants pris en charge : `aws-sagemaker-spark-sdk`, `emr-ddb`, `emr-goodies`, `emr-s3-select`, `emrfs`, `hadoop-client`, `hudi`, `hudi-spark`, `iceberg`, `spark-kubernetes`.
- Classifications de configuration prises en charge

  À utiliser avec StartJobRunet CreateManagedEndpoint APIs:

Classifications	Descriptions
`core-site`	Modifiez les valeurs dans le fichier Hadoop `core-site.xml` .
`emrfs-site`	Modifiez les paramètres EMRFS.
`spark-metrics`	Modifiez les valeurs dans le fichier Spark `metrics.properties` .
`spark-defaults`	Modifiez les valeurs dans le fichier Spark `spark-defaults.conf` .
`spark-env`	Modifiez les valeurs dans l'environnement Spark.
`spark-hive-site`	Modifiez les valeurs dans le fichier Spark `hive-site.xml` .
`spark-log4j`	Modifiez les valeurs dans le fichier Spark `log4j2.properties` .
`emr-job-submitter`	Configuration pour le pod soumissionnaire de tâches.

À utiliser spécifiquement avec CreateManagedEndpoint APIs:

Classifications	Descriptions
`jeg-config`	Modifiez les valeurs dans le fichier `jupyter_enterprise_gateway_config.py` Jupyter Enterprise Gateway.
`jupyter-kernel-overrides`	Modifiez la valeur de l'image du noyau dans le fichier Jupyter Kernel Spec.

Les classifications de configuration vous permettent de personnaliser les applications. Elles correspondent souvent à un fichier XML de configuration de l'application, tel que `spark-hive-site.xml`. Pour plus d'informations, consultez la rubrique Configuration des applications.

Fonctionnalités notables

Les fonctionnalités suivantes sont incluses dans la version 7.0 d'Amazon EMR sur EKS.

- Mises à niveau d'application – Les mises à niveau d'application Amazon EMR sur EKS 7.0.0 incluent Spark 3.5, Flink 1.18 et Flink Operator 1.6.1.
- Réglage automatique des paramètres de l'outil de mise à l'échelle automatique Flink : les paramètres par défaut utilisés par l'outil de mise à l'échelle automatique Flink pour ses calculs de mise à l'échelle peuvent ne pas être optimaux pour une tâche donnée. Amazon EMR sur EKS 7.0.0 utilise les tendances historiques de mesures capturées spécifiques pour calculer le paramètre optimal pour la tâche en question.

Modifications

Les fonctionnalités suivantes sont incluses dans la version 7.0 d'Amazon EMR sur EKS.

- Amazon Linux 2023 — Avec Amazon EMR sur EKS 7.0.0 et versions ultérieures, toutes les images de conteneur sont basées sur Amazon Linux 2023.
- Spark utilise Java 17 comme environnement d'exécution par défaut : dans Amazon EMR sur EKS 7.0.0, Spark utilise Java 17 comme environnement d'exécution par défaut. Si nécessaire, vous pouvez passer à Java 8 ou Java 11 avec l'étiquette de version correspondante, comme indiqué dans la liste Versions 7.0 d'Amazon EMR sur EKS.

emr-7.0.0-latest

Notes de mise à jour : `emr-7.0.0-latest` renvoie actuellement à `emr-7.0.0-2024321`.

Régions : `emr-7.0.0-latest` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-7.0.0:latest`

emr-7.0.0-2024321

Notes de publication : `7.0.0-2024321` a été publié le 11 mars 2024. Par rapport à la version précédente, cette version a été actualisée avec les packages Amazon Linux récemment mis à jour et les correctifs critiques.

Régions : `emr-7.0.0-2024321` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-7.0.0:2024321`

emr-7.0.0-20231211

Notes de version : la version `7.0.0-20231211` a été publiée en décembre 2023. Il s'agit de la première version d'Amazon EMR 7.0.0 (Spark).

Régions : `emr-7.0.0-20231211` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-7.0.0:20231211`

emr-7.0.0-flink-latest

Notes de mise à jour : `emr-7.0.0-flink-latest` renvoie actuellement à `emr-7.0.0-flink-2024321`.

Régions : `emr-7.0.0-flink-latest` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-7.0.0-flink:latest`

emr-7.0.0-flink-2024321

Notes de publication : `7.0.0-flink-2024321` a été publié le 11 mars 2024. Par rapport à la version précédente, cette version a été actualisée avec les packages Amazon Linux récemment mis à jour et les correctifs critiques.

Régions : `emr-7.0.0-flink-2024321` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-7.0.0-flink:2024321`

emr-7.0.0-flink-20231211

Notes de version : la version `7.0.0-flink-20231211` a été publiée en décembre 2023. Il s'agit de la première version d'Amazon EMR 7.0.0 (Flink).

Régions : `emr-7.0.0-flink-20231211` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-7.0.0-flink:20231211`

Versions 6.15.0 d'Amazon EMR sur EKS

Cette page décrit les fonctionnalités nouvelles et mises à jour d'Amazon EMR spécifiques au déploiement d'Amazon EMR on EKS. Pour en savoir plus sur Amazon EMR exécuté sur Amazon EC2 et sur la version 6.15.0 d'Amazon EMR en général, consultez Amazon EMR 6.15.0 dans le guide de mise à jour d'Amazon EMR.

Versions 6.15 d'Amazon EMR sur EKS

Les versions 6.15.0 suivantes d'Amazon EMR sont disponibles pour Amazon EMR sur EKS. Sélectionnez une version emr-6.15.0-XXXX spécifique pour afficher plus d'informations, comme la balise de l'image du conteneur correspondant.

Flink releases

Les versions 6.15.0 suivantes d'Amazon EMR sont disponibles pour Amazon EMR sur EKS lorsque vous exécutez des applications Flink.

- emr-6.15.0-flink-latest
- emr-6.15.0-flink-20240105
- emr-6.15.0-flink-20231109

Spark releases

Les versions 6.15.0 suivantes d'Amazon EMR sont disponibles pour Amazon EMR sur EKS lorsque vous exécutez des applications Spark.

- emr-6.15.0-latest
- emr-6.15.0-20231109
- emr-6.15.0-spark-rapids-latest
- emr-6.15.0-spark-rapids-20231109
- emr-6.15.0-java11-latest
- emr-6.15.0-java11-20231109
- emr-6.15.0-java17-latest
- emr-6.15.0-java17-20231109
- emr-6.15.0-java17-al2023-latest
- emr-6.15.0-java17-al2023-20231109
- emr-6.15.0-spark-rapids-java17-latest
- emr-6.15.0-spark-rapids-java17-20231109
- emr-6.15.0-spark-rapids-java17-al2023-latest
- emr-6.15.0-spark-rapids-java17-al2023-20231109
- notebook-spark/emr-6.15.0-latest
- notebook-spark/emr-6.15.0-20231109
- notebook-spark/emr-6.15.0-spark-rapids-latest
- notebook-spark/emr-6.15.0-spark-rapids-20231109
- notebook-spark/emr-6.15.0-java11-latest
- notebook-spark/emr-6.15.0-java11-20231109
- notebook-spark/emr-6.15.0-java17-latest
- notebook-spark/emr-6.15.0-java17-20231109
- notebook-spark/emr-6.15.0-java17-al2023-latest
- notebook-spark/emr-6.15.0-java17-al2023-20231109
- notebook-python/emr-6.15.0-latest
- notebook-python/emr-6.15.0-20231109
- notebook-python/emr-6.15.0-spark-rapids-latest

- notebook-python/emr-6.15.0-spark-rapids-20231109
- notebook-python/emr-6.15.0-java11-latest
- notebook-python/emr-6.15.0-java11-20231109
- notebook-python/emr-6.15.0-java17-latest
- notebook-python/emr-6.15.0-java17-20231109
- notebook-python/emr-6.15.0-java17-al2023-latest
- notebook-python/emr-6.15.0-java17-al2023-20231109

Notes de mise à jour

Notes de version pour Amazon EMR sur EKS 6.15.0

- Applications prises en charge – AWS SDK pour Java 1.12.569, Apache Spark 3.4.1-amzn-2, Apache Flink 1.17.1-amzn-1, Apache Hudi 0.14.0-amzn-0, Apache Iceberg 1.4.0-amzn-0, Delta 2.4.0, Apache Spark RAPIDS 23.08.01-amzn-0, Jupyter Enterprise Gateway 2.6.0
- Composants pris en charge : `aws-sagemaker-spark-sdk`, `emr-ddb`, `emr-goodies`, `emr-s3-select`, `emrfs`, `hadoop-client`, `hudi`, `hudi-spark`, `iceberg`, `spark-kubernetes`.
- Classifications de configuration prises en charge

  À utiliser avec StartJobRunet CreateManagedEndpoint APIs:

Classifications	Descriptions
`core-site`	Modifiez les valeurs dans le fichier Hadoop `core-site.xml` .
`emrfs-site`	Modifiez les paramètres EMRFS.
`spark-metrics`	Modifiez les valeurs dans le fichier Spark `metrics.properties` .
`spark-defaults`	Modifiez les valeurs dans le fichier Spark `spark-defaults.conf` .
`spark-env`	Modifiez les valeurs dans l'environnement Spark.

Classifications	Descriptions
`spark-hive-site`	Modifiez les valeurs dans le fichier Spark `hive-site.xml` .
`spark-log4j`	Modifiez les valeurs dans le fichier Spark `log4j2.properties` .
`emr-job-submitter`	Configuration pour le pod soumissionnaire de tâches.

À utiliser spécifiquement avec CreateManagedEndpoint APIs:

Classifications	Descriptions
`jeg-config`	Modifiez les valeurs dans le fichier `jupyter_enterprise_gateway_config.py` Jupyter Enterprise Gateway.
`jupyter-kernel-overrides`	Modifiez la valeur de l'image du noyau dans le fichier Jupyter Kernel Spec.

Les classifications de configuration vous permettent de personnaliser les applications. Elles correspondent souvent à un fichier XML de configuration de l'application, tel que `spark-hive-site.xml`. Pour plus d'informations, consultez la rubrique Configuration des applications.

Fonctionnalités notables

Les fonctionnalités suivantes sont incluses dans la version 6.15 d'Amazon EMR sur EKS.

- Amazon EMR sur EKS avec Apache Flink : avec Amazon EMR sur EKS 6.15.0, vous pouvez exécuter votre application basée sur Apache Flink ainsi que d'autres types d'applications sur le même cluster Amazon EKS. Cela permet d'améliorer l'utilisation des ressources et de simplifier la gestion de l'infrastructure. Vous pouvez tirer parti des instances Spot dans une application Flink grâce à la mise hors service progressive et accélérer les redémarrages grâce à la récupération précise et la récupération locale des tâches avec Amazon EBS. Les fonctionnalités d'accessibilité

et de surveillance incluent la possibilité de lancer une application Flink avec des fichiers JAR stockés dans Amazon S3, l'accès au catalogue de données AWS Glue, le suivi de l'intégration avec Amazon S3 et Amazon CloudWatch et la rotation des journaux de conteneurs.

emr-6.15.0-latest

Notes de mise à jour : `emr-6.15.0-latest` renvoie actuellement à `emr-6.15.0-20240105`.

Régions : `emr-6.15.0-latest` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-6.15.0:latest`

emr-6.15.0-20240105

Notes de publication : `6.15.0-20240105` a été publié le 17 janvier 2024. Par rapport à la version précédente, cette version a été actualisée avec les packages Amazon Linux récemment mis à jour et les correctifs critiques.

Régions : `emr-6.15.0-20240105` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-6.15.0:20240105`

emr-6.15.0-20231109

Notes de version : la version `6.15.0-20231109` a été publiée le 17 novembre 2023. Il s’agit de la première version d’Amazon EMR 6.15.0.

Régions : `emr-6.15.0-20231109` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-6.15.0:20231109`

emr-6.15.0-flink-latest

Notes de mise à jour : `emr-6.15.0-flink-latest` renvoie actuellement à `emr-6.15.0-flink-20240105`.

Régions : `emr-6.15.0-flink-latest` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-6.15.0-flink:latest`

emr-6.15.0-flink-20240105

Notes de publication : `6.15.0-flink-20240105` a été publié le 17 janvier 2024. Par rapport à la version précédente, cette version a été actualisée avec les packages Amazon Linux récemment mis à jour et les correctifs critiques.

Régions : `emr-6.15.0-flink-20240105` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-6.15.0-flink:20240105`

emr-6.15.0-flink-20231109

Notes de version : la version `6.15.0-flink-20231109` a été publiée le 17 novembre 2023. Il s'agit de la première version d'Amazon EMR 6.15.0.

Régions : `emr-6.15.0-flink-20231109` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-6.15.0-flink:20231109`

Versions 6.14.0 d'Amazon EMR sur EKS

Cette page décrit les fonctionnalités nouvelles et mises à jour d'Amazon EMR spécifiques au déploiement d'Amazon EMR on EKS. Pour en savoir plus sur Amazon EMR exécuté sur Amazon EC2 et sur la version 6.14.0 d'Amazon EMR en général, consultez Amazon EMR 6.14.0 dans le guide de mise à jour d'Amazon EMR.

Versions 6.14 d'Amazon EMR sur EKS

Les versions 6.14.0 suivantes d'Amazon EMR sont disponibles pour Amazon EMR sur EKS. Sélectionnez une version emr-6.14.0-XXXX spécifique pour voir plus de détails tels que la balise de l'image du conteneur correspondant.

- emr-6.14.0-latest
- emr-6.14.0-20231005
- emr-6.14.0-spark-rapids-latest
- emr-6.14.0-spark-rapids-20231005
- emr-6.14.0-java11-latest
- emr-6.14.0-java11-20231005
- emr-6.14.0-java17-latest
- emr-6.14.0-java17-20231005
- emr-6.14.0-java17-al2023-latest
- emr-6.14.0-java17-al2023-20231005
- emr-6.14.0-spark-rapids-java17-latest
- emr-6.14.0-spark-rapids-java17-20231005
- emr-6.14.0-spark-rapids-java17-al2023-latest
- emr-6.14.0-spark-rapids-java17-al2023-20231005
- notebook-spark/emr-6.14.0-latest
- notebook-spark/emr-6.14.0-20231005
- notebook-spark/emr-6.14.0-spark-rapids-latest
- notebook-spark/emr-6.14.0-spark-rapids-20231005
- notebook-spark/emr-6.14.0-java11-latest
- notebook-spark/emr-6.14.0-java11-20231005
- notebook-spark/emr-6.14.0-java17-latest
- notebook-spark/emr-6.14.0-java17-20231005
- notebook-spark/emr-6.14.0-java17-al2023-latest
- notebook-spark/emr-6.14.0-java17-al2023-20231005
- notebook-python/emr-6.14.0-latest
- notebook-python/emr-6.14.0-20231005
- notebook-python/emr-6.14.0-spark-rapids-latest
- notebook-python/emr-6.14.0-spark-rapids-20231005
- notebook-python/emr-6.14.0-java11-latest

- notebook-python/emr-6.14.0-java11-20231005
- notebook-python/emr-6.14.0-java17-latest
- notebook-python/emr-6.14.0-java17-20231005
- notebook-python/emr-6.14.0-java17-al2023-latest
- notebook-python/emr-6.14.0-java17-al2023-20231005

Notes de mise à jour

Notes de mise à jour pour Amazon EMR sur EKS 6.14.0

- Applications prises en charge - AWS SDK pour Java 1.12.543, Apache Spark 3.4.1-amzn-1, Apache Hudi 0.13.1-amzn-2, Apache Iceberg 1.3.0-amzn-0, Delta 2.4.0, Apache Spark RAPIDS 23.06.0-amzn-2, Jupyter Enterprise Gateway 2.7.0
- Composants pris en charge : `aws-sagemaker-spark-sdk`, `emr-ddb`, `emr-goodies`, `emr-s3-select`, `emrfs`, `hadoop-client`, `hudi`, `hudi-spark`, `iceberg`, `spark-kubernetes`.
- Classifications de configuration prises en charge

  À utiliser avec StartJobRunet CreateManagedEndpoint APIs:

Classifications	Descriptions
`core-site`	Modifiez les valeurs dans le fichier Hadoop `core-site.xml` .
`emrfs-site`	Modifiez les paramètres EMRFS.
`spark-metrics`	Modifiez les valeurs dans le fichier Spark `metrics.properties` .
`spark-defaults`	Modifiez les valeurs dans le fichier Spark `spark-defaults.conf` .
`spark-env`	Modifiez les valeurs dans l'environnement Spark.
`spark-hive-site`	Modifiez les valeurs dans le fichier Spark `hive-site.xml` .

Classifications	Descriptions
`spark-log4j`	Modifiez les valeurs dans le fichier Spark `log4j2.properties` .
`emr-job-submitter`	Configuration pour le pod soumissionnaire de tâches.

À utiliser spécifiquement avec CreateManagedEndpoint APIs:

Classifications	Descriptions
`jeg-config`	Modifiez les valeurs dans le fichier `jupyter_enterprise_gateway_config.py` Jupyter Enterprise Gateway.
`jupyter-kernel-overrides`	Modifiez la valeur de l'image du noyau dans le fichier Jupyter Kernel Spec.

Les classifications de configuration vous permettent de personnaliser les applications. Elles correspondent souvent à un fichier XML de configuration de l'application, tel que `spark-hive-site.xml`. Pour plus d'informations, consultez la rubrique Configuration des applications.

Fonctionnalités notables

Les fonctionnalités suivantes sont incluses dans la version 6.14 d'Amazon EMR sur EKS.

- Prise en charge d'Apache Livy : Amazon EMR sur EKS prend désormais en charge Apache Livy avec `spark-submit`.

emr-6.14.0-latest

Notes de mise à jour : `emr-6.14.0-latest` renvoie actuellement à `emr-6.14.0-20231005`.

Régions : `emr-6.14.0-latest` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-6.14.0:latest`

emr-6.14.0-20231005

Notes de mise à jour : la version `6.14.0-20231005` a été publiée le 17 octobre 2023. Il s'agit de la première version 6.14.0 d'Amazon EMR.

Régions : `emr-6.14.0-20231005` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-6.14.0:20231005`

Versions 6.13.0 d'Amazon EMR on EKS

Cette page décrit les fonctionnalités nouvelles et mises à jour d'Amazon EMR spécifiques au déploiement d'Amazon EMR on EKS. Pour en savoir plus sur Amazon EMR exécuté sur Amazon EC2 et sur la version 6.13.0 d'Amazon EMR en général, consultez Amazon EMR 6.13.0 dans le guide de mise à jour d'Amazon EMR.

Versions 6.13 d'Amazon EMR on EKS

Les versions 6.13.0 suivantes d'Amazon EMR sont disponibles pour Amazon EMR on EKS. Sélectionnez une version emr-6.13.0-XXXX spécifique pour voir plus de détails tels que la balise de l'image du conteneur correspondant.

- emr-6.13.0-latest
- emr-6.13.0-20230814
- emr-6.13.0-spark-rapids-latest
- emr-6.13.0-spark-rapids-20230814
- emr-6.13.0-java11-latest
- emr-6.13.0-java11-20230814
- emr-6.13.0-java17-latest

- emr-6.13.0-java17-20230814
- emr-6.13.0-java17-al2023-latest
- emr-6.13.0-java17-al2023-20230814
- emr-6.13.0-spark-rapids-java17-latest
- emr-6.13.0-spark-rapids-java17-20230814
- emr-6.13.0-spark-rapids-java17-al2023-latest
- emr-6.13.0-spark-rapids-java17-al2023-20230814
- notebook-spark/emr-6.13.0-latest
- notebook-spark/emr-6.13.0-20230814
- notebook-spark/emr-6.13.0-spark-rapids-latest
- notebook-spark/emr-6.13.0-spark-rapids-20230814
- notebook-spark/emr-6.13.0-java11-latest
- notebook-spark/emr-6.13.0-java11-20230814
- notebook-spark/emr-6.13.0-java17-latest
- notebook-spark/emr-6.13.0-java17-20230814
- notebook-spark/emr-6.13.0-java17-al2023-latest
- notebook-spark/emr-6.13.0-java17-al2023-20230814
- notebook-python/emr-6.13.0-latest
- notebook-python/emr-6.13.0-20230814
- notebook-python/emr-6.13.0-spark-rapids-latest
- notebook-python/emr-6.13.0-spark-rapids-20230814
- notebook-python/emr-6.13.0-java11-latest
- notebook-python/emr-6.13.0-java11-20230814
- notebook-python/emr-6.13.0-java17-latest
- notebook-python/emr-6.13.0-java17-20230814
- notebook-python/emr-6.13.0-java17-al2023-latest
- notebook-python/emr-6.13.0-java17-al2023-20230814

Notes de mise à jour

Notes de mise à jour pour Amazon EMR on EKS 6.13.0

- Applications prises en charge - AWS SDK pour Java 1.12.513, Apache Spark 3.4.1-amzn-0, Apache Hudi 0.13.1-amzn-0, Apache Iceberg 1.3.0-amzn-0, Delta 2.4.0, Apache Spark RAPIDS 23.06.0-amzn-1, Jupyter Enterprise Gateway 2.6.0.amzn
- Composants pris en charge : `aws-sagemaker-spark-sdk`, `emr-ddb`, `emr-goodies`, `emr-s3-select`, `emrfs`, `hadoop-client`, `hudi`, `hudi-spark`, `iceberg`, `spark-kubernetes`.
- Classifications de configuration prises en charge

  À utiliser avec StartJobRunet CreateManagedEndpoint APIs:

Classifications	Descriptions
`core-site`	Modifiez les valeurs dans le fichier Hadoop `core-site.xml` .
`emrfs-site`	Modifiez les paramètres EMRFS.
`spark-metrics`	Modifiez les valeurs dans le fichier Spark `metrics.properties` .
`spark-defaults`	Modifiez les valeurs dans le fichier Spark `spark-defaults.conf` .
`spark-env`	Modifiez les valeurs dans l'environnement Spark.
`spark-hive-site`	Modifiez les valeurs dans le fichier Spark `hive-site.xml` .
`spark-log4j`	Modifiez les valeurs dans le fichier Spark `log4j2.properties` .
`emr-job-submitter`	Configuration pour le pod soumissionnaire de tâches.

  À utiliser spécifiquement avec CreateManagedEndpoint APIs:

Classifications	Descriptions
`jeg-config`	Modifiez les valeurs dans le fichier `jupyter_enterprise_gateway_config.py` Jupyter Enterprise Gateway.
`jupyter-kernel-overrides`	Modifiez la valeur de l'image du noyau dans le fichier Jupyter Kernel Spec.

Les classifications de configuration vous permettent de personnaliser les applications. Elles correspondent souvent à un fichier XML de configuration de l'application, tel que `spark-hive-site.xml`. Pour plus d'informations, consultez la rubrique Configuration des applications.

Fonctionnalités notables

Les fonctionnalités suivantes sont incluses dans la version 6.13 d'Amazon EMR on EKS.

- Amazon Linux 2023 - Avec Amazon EMR sur EKS 6.13 ou version ultérieure, vous pouvez lancer Spark avec le système d'exploitation AL2 023 avec le moteur d'exécution Java 17. Pour ce faire, utilisez l'étiquette de version dont le nom comporte `al2023`. Par exemple : `emr-6.13.0-java17-al2023-latest`. Nous vous recommandons de valider et d'exécuter des tests de performances avant de déplacer vos charges de travail de production vers AL2 023 et Java 17.
- Amazon EMR sur EKS avec Apache Flink (version préliminaire publique) : les versions 6.13 et ultérieures d'Amazon EMR sur EKS prennent en charge Apache Flink, disponible en version préliminaire publique. Grâce à ce lancement, vous pouvez exécuter votre application Apache Flink ainsi que d'autres types d'applications sur le même cluster Amazon EKS. Cela permet d'améliorer l'utilisation des ressources et de simplifier la gestion de l'infrastructure. Si vous exécutez déjà des environnements de big data sur Amazon EKS, vous pouvez désormais laisser Amazon EMR automatiser votre provisionnement et votre gestion.

emr-6.13.0-latest

Notes de mise à jour : `emr-6.13.0-latest` renvoie actuellement à `emr-6.13.0-20230814`.

Régions : `emr-6.13.0-latest` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-6.13.0:latest`

emr-6.13.0-20230814

Notes de mise à jour : la version `6.13.0-20230814` a été publiée le 7 septembre 2023. Il s'agit de la première version 6.13.0 d'Amazon EMR.

Régions : `emr-6.13.0-20230814` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-6.13.0:20230814`

Versions 6.12.0 d'Amazon EMR on EKS

Cette page décrit les fonctionnalités nouvelles et mises à jour d'Amazon EMR spécifiques au déploiement d'Amazon EMR on EKS. Pour en savoir plus sur Amazon EMR exécuté sur Amazon EC2 et sur la version 6.12.0 d'Amazon EMR en général, consultez Amazon EMR 6.12.0 dans le guide de mise à jour d'Amazon EMR.

Versions 6.12 d'Amazon EMR on EKS

Les versions 6.12.0 suivantes d'Amazon EMR sont disponibles pour Amazon EMR on EKS. Sélectionnez une version emr-6.12.0-XXXX spécifique pour voir plus de détails tels que la balise de l'image du conteneur correspondant.

- emr-6.12.0-latest
- emr-6.12.0-20240321
- emr-6.12.0-20230701
- emr-6.12.0- spark-rapids-latest
- emr-6.12.0-spark-rapids-20230701
- emr-6.12.0-java11-latest
- emr-6.12.0-java11-20230701
- emr-6.12.0-java17-latest

- emr-6.12.0-java17-20230701
- emr-6.12.0- 17-dernier spark-rapids-java
- emr-6.12.0- 17-20230701 spark-rapids-java
- notebook-spark/emr-6.12.0-latest
- notebook-spark/emr-6.12.0-20230701
- bloc-bloc-spark/emr-6.12.0- spark-rapids-latest
- notebook-spark/emr-6.12.0-spark-rapids-20230701
- notebook-python/emr-6.12.0-latest
- notebook-python/emr-6.12.0-20230701
- bloc-bloc-python/emr-6.12.0- spark-rapids-latest
- notebook-python/emr-6.12.0-spark-rapids-20230701

Notes de mise à jour

Notes de mise à jour pour Amazon EMR on EKS 6.12.0

- Applications prises en charge - AWS SDK pour Java 1.12.490, Apache Spark 3.4.0-amzn-0, Apache Hudi 0.13.1-amzn-0, Apache Iceberg 1.3.0-amzn-0, Delta 2.4.0, Apache Spark RAPIDS 23.06.0-amzn-0, Jupyter Enterprise Gateway 2.6.0
- Composants pris en charge : `aws-sagemaker-spark-sdk`, `emr-ddb`, `emr-goodies`, `emr-s3-select`, `emrfs`, `hadoop-client`, `hudi`, `hudi-spark`, `iceberg`, `spark-kubernetes`.
- Classifications de configuration prises en charge

  À utiliser avec StartJobRunet CreateManagedEndpoint APIs:

Classifications	Descriptions
`core-site`	Modifiez les valeurs dans le fichier Hadoop `core-site.xml` .
`emrfs-site`	Modifiez les paramètres EMRFS.
`spark-metrics`	Modifiez les valeurs dans le fichier Spark `metrics.properties` .

Classifications	Descriptions
`spark-defaults`	Modifiez les valeurs dans le fichier Spark `spark-defaults.conf` .
`spark-env`	Modifiez les valeurs dans l'environnement Spark.
`spark-hive-site`	Modifiez les valeurs dans le fichier Spark `hive-site.xml` .
`spark-log4j`	Modifiez les valeurs dans le fichier Spark `log4j2.properties` .
`emr-job-submitter`	Configuration pour le pod soumissionnaire de tâches.

À utiliser spécifiquement avec CreateManagedEndpoint APIs:

Classifications	Descriptions
`jeg-config`	Modifiez les valeurs dans le fichier `jupyter_enterprise_gateway_config.py` Jupyter Enterprise Gateway.
`jupyter-kernel-overrides`	Modifiez la valeur de l'image du noyau dans le fichier Jupyter Kernel Spec.

Les classifications de configuration vous permettent de personnaliser les applications. Elles correspondent souvent à un fichier XML de configuration de l'application, tel que `spark-hive-site.xml`. Pour plus d'informations, consultez la rubrique Configuration des applications.

Fonctionnalités notables

Les fonctionnalités suivantes sont incluses dans la version 6.12 d'Amazon EMR on EKS.

- Java 17 – Grâce à Amazon EMR on EKS en version 6.12 et supérieure, vous pouvez lancer Spark avec le moteur d'exécution Java 17. Pour ce faire, indiquez `emr-6.12.0-java17-latest` comme étiquette de version. Nous vous recommandons de valider et d'exécuter des tests de performances avant de transférer vos charges de travail de production des versions antérieures de l'image Java vers l'image Java 17.

emr-6.12.0-latest

Notes de mise à jour : `emr-6.12.0-latest` renvoie actuellement à `emr-6.12.0-20240321`.

Régions : `emr-6.12.0-latest` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-6.12.0:latest`

emr-6.12.0-20240321

Notes de publication : `6.12.0-20240321` a été publié le 11 mars 2024. Par rapport à la version précédente, cette version a été actualisée avec les packages Amazon Linux récemment mis à jour et les correctifs critiques.

Régions : `emr-6.12.0-20240321` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-6.12.0:20240321`

emr-6.12.0-20230701

Notes de mise à jour : la version `6.12.0-20230701` a été publiée le 1er juillet 2023. Il s'agit de la première version 6.12.0 d'Amazon EMR.

Régions : `emr-6.12.0-20230701` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-6.12.0:20230701`

Versions 6.11.0 d'Amazon EMR on EKS

Cette page décrit les fonctionnalités nouvelles et mises à jour d'Amazon EMR spécifiques au déploiement d'Amazon EMR on EKS. Pour en savoir plus sur Amazon EMR exécuté sur Amazon EC2 et sur la version 6.11.0 d'Amazon EMR en général, consultez Amazon EMR 6.11.0 dans le guide de mise à jour d'Amazon EMR.

Versions 6.11 d'Amazon EMR on EKS

Les versions 6.11.0 suivantes d'Amazon EMR sont disponibles pour Amazon EMR on EKS. Sélectionnez une version emr-6.11.0-XXXX spécifique pour voir plus de détails tels que la balise de l'image du conteneur correspondant.

- emr-6.11.0-latest
- emr-6.11.0-20230905
- emr-6.11.0-20230509

- emr-6.11.0- spark-rapids-latest
- emr-6.11.0-spark-rapids-20230509
- emr-6.11.0-java11-latest
- emr-6.11.0-java11-20230509
- notebook-spark/emr-6.11.0-latest
- notebook-spark/emr-6.11.0-20230509
- notebook-python/emr-6.11.0-latest
- notebook-python/emr-6.11.0-20230509

Notes de mise à jour

Notes de mise à jour pour Amazon EMR on EKS 6.11.0

- Applications prises en charge - AWS SDK pour Java 1.12.446, Apache Spark 3.3.2-amzn-0, Apache Hudi 0.13.0-amzn-0, Apache Iceberg 1.2.0-amzn-0, Delta 2.2.0, Apache Spark RAPIDS 23.02.0-amzn-0, Jupyter Enterprise Gateway 2.6.0
- Composants pris en charge : `aws-sagemaker-spark-sdk`, `emr-ddb`, `emr-goodies`, `emr-s3-select`, `emrfs`, `hadoop-client`, `hudi`, `hudi-spark`, `iceberg`, `spark-kubernetes`.

- Classifications de configuration prises en charge

  À utiliser avec StartJobRunet CreateManagedEndpoint APIs:

Classifications	Descriptions
`core-site`	Modifiez les valeurs dans le fichier Hadoop `core-site.xml` .
`emrfs-site`	Modifiez les paramètres EMRFS.
`spark-metrics`	Modifiez les valeurs dans le fichier Spark `metrics.properties` .
`spark-defaults`	Modifiez les valeurs dans le fichier Spark `spark-defaults.conf` .
`spark-env`	Modifiez les valeurs dans l'environnement Spark.
`spark-hive-site`	Modifiez les valeurs dans le fichier Spark `hive-site.xml` .
`spark-log4j`	Modifiez les valeurs dans le fichier Spark `log4j.properties` .

  À utiliser spécifiquement avec CreateManagedEndpoint APIs:

Classifications	Descriptions
`jeg-config`	Modifiez les valeurs dans le fichier `jupyter_enterprise_gateway_config.py` Jupyter Enterprise Gateway.
`jupyter-kernel-overrides`	Modifiez la valeur de l'image du noyau dans le fichier Jupyter Kernel Spec.

Les classifications de configuration vous permettent de personnaliser les applications. Elles correspondent souvent à un fichier XML de configuration de l'application, tel que `spark-hive-site.xml`. Pour plus d'informations, consultez la rubrique Configuration des applications.

Fonctionnalités notables

Les fonctionnalités suivantes sont incluses dans la version 6.11 d'Amazon EMR on EKS.

- Image de base Amazon EMR on EKS dans la galerie publique Amazon ECR – Si vous utilisez la fonctionnalité d'image personnalisée, notre image de base fournit les fichiers jar, la configuration et les bibliothèques essentiels pour interagir avec Amazon EMR on EKS. Vous pouvez désormais trouver l'image de base dans la galerie publique Amazon ECR.
- Rotation des journaux des conteneurs Spark – Amazon EMR on EKS 6.11 prend en charge la rotation des journaux des conteneurs Spark. Vous pouvez activer cette fonctionnalité avec `containerLogRotationConfiguration` dans le cadre de l'opération `MonitoringConfiguration` de l'API `StartJobRun`. Vous pouvez configurer `rotationSize` et `maxFilestoKeep` pour spécifier le nombre et la taille des fichiers journaux que vous souhaitez qu'Amazon EMR on EKS conserve dans les pods de pilotes et d'exécuteurs Spark. Pour de plus amples informations, veuillez consulter Utilisation de la rotation des journaux des conteneurs Spark.
- Prise en charge de Volcano dans l'opérateur Spark et spark-submit – Amazon EMR on EKS 6.11 prend en charge l'exécution de tâches Spark avec Volcano en tant que planificateur personnalisé Kubernetes dans l'opérateur Spark et spark-submit. Vous pouvez utiliser des fonctionnalités telles que la planification groupée, la gestion des files d'attente, la préemption et la planification équitable pour obtenir un débit de planification élevé et une capacité optimisée. Pour de plus amples informations, veuillez consulter Utilisation de Volcano comme planificateur personnalisé pour Apache Spark sur Amazon EMR on EKS.

emr-6.11.0-latest

Notes de mise à jour : `emr-6.11.0-latest` renvoie actuellement à `emr-20230905`.

Régions : `emr-6.11.0-latest` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-6.11.0:latest`

emr-6.11.0-20230905

Notes de publication : `6.11.0-20230905` a été publié le 29 septembre 2023. Par rapport à la version précédente, cette version a été actualisée avec les packages Amazon Linux récemment mis à jour et les correctifs critiques.

Régions : `emr-6.11.0-20230509` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-6.11.0:20230509`

emr-6.11.0-20230509

Notes de mise à jour : la version `6.11.0-20230509` a été publiée le 9 mai 2023. Il s'agit de la première version 6.11.0 d'Amazon EMR.

Régions : `emr-6.11.0-20230509` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-6.11.0:20230509`

Versions 6.10.0 d'Amazon EMR on EKS

Les versions 6.10.0 suivantes d'Amazon EMR sont disponibles pour Amazon EMR on EKS. Sélectionnez une version emr-6.10.0-XXXX spécifique pour voir plus de détails tels que la balise de l'image du conteneur correspondant.

- emr-6.10.0-latest
- emr-6.10.0-20230905
- emr-6.10.0-20230624
- emr-6.10.0-20230421
- emr-6.10.0-20230403
- emr-6.10.0-20230220

- emr-6.10.0- spark-rapids-latest
- emr-6.10.0-spark-rapids-20230624
- emr-6.10.0-spark-rapids-20230220
- emr-6.10.0-java11-latest
- emr-6.10.0-java11-20230624
- emr-6.10.0-java11-20230220
- notebook-spark/emr-6.10.0-latest
- notebook-spark/emr-6.10.0-20230624
- notebook-spark/emr-6.10.0-20230220
- notebook-python/emr-6.10.0-latest
- notebook-python/emr-6.10.0-20230624
- notebook-python/emr-6.10.0-20230220

Notes de mise à jour pour Amazon EMR 6.10.0

- Applications prises en charge - AWS SDK pour Java 1.12.397, Spark 3.3.1-amzn-0, Hudi 0.12.2-amzn-0, Iceberg 1.1.0-amzn-0, Delta 2.2.0.
- Composants pris en charge : `aws-sagemaker-spark-sdk`, `emr-ddb`, `emr-goodies`, `emr-s3-select`, `emrfs`, `hadoop-client`, `hudi`, `hudi-spark`, `iceberg`, `spark-kubernetes`.
- Classifications de configuration prises en charge :

  À utiliser avec StartJobRunet CreateManagedEndpoint APIs:

Classifications	Descriptions
`core-site`	Modifiez les valeurs dans le fichier `core-site.xml` de Hadoop.
`emrfs-site`	Modifiez les paramètres EMRFS.
`spark-metrics`	Modifiez les valeurs dans le fichier `metrics.properties` de Spark.
`spark-defaults`	Modifiez les valeurs dans le fichier `spark-defaults.conf` de Spark.

Classifications	Descriptions
`spark-env`	Modifiez les valeurs dans l'environnement Spark.
`spark-hive-site`	Modifiez les valeurs dans le fichier `hive-site.xml` de Spark.
`spark-log4j`	Modifiez les valeurs dans le fichier `log4j.properties` de Spark.

À utiliser spécifiquement avec CreateManagedEndpoint APIs:

Classifications	Descriptions
`jeg-config`	Modifiez les valeurs dans le fichier `jupyter_enterprise_gateway_config.py` Jupyter Enterprise Gateway.
`jupyter-kernel-overrides`	Modifiez la valeur de l'image du noyau dans le fichier Jupyter Kernel Spec.

Les classifications de configuration vous permettent de personnaliser les applications. Elles correspondent souvent à un fichier XML de configuration de l'application, tel que `spark-hive-site.xml`. Pour plus d'informations, consultez la rubrique Configuration des applications.

Fonctionnalités notables

- Opérateur Spark – Grâce à Amazon EMR on EKS en version 6.10.0 et supérieure, vous pouvez utiliser l'opérateur Kubernetes pour Apache Spark, ou l'opérateur Spark, pour déployer et gérer des applications Spark avec le moteur d'exécution de la version Amazon EMR sur vos propres clusters Amazon EKS. Pour de plus amples informations, veuillez consulter Exécution de tâches Spark à l'aide de l'opérateur Spark.
- Java 11 – Grâce à Amazon EMR on EKS en version 6.10 et supérieure, vous pouvez lancer Spark avec le moteur d'exécution Java 11. Pour ce faire, indiquez `emr-6.10.0-java11-latest` comme étiquette de version. Nous vous recommandons de valider et d'exécuter des tests de

performance avant de transférer vos charges de travail de production de l'image Java 8 vers l'image Java 11.

- Pour l'intégration d'Amazon Redshift à Apache Spark, Amazon EMR on EKS 6.10.0 supprime la dépendance à `minimal-json.jar` et ajoute automatiquement les fichiers jar `spark-redshift` associés requis au chemin de classe de l'exécuteur pour Spark : `spark-redshift.jar`, `spark-avro.jar` et `RedshiftJDBC.jar`.

Modifications

- Le validateur optimisé pour EMRFS S3 est désormais activé par défaut pour les formats parquet, ORC et texte (y compris CSV et JSON).

emr-6.10.0-latest

Notes de mise à jour : `emr-6.10.0-latest` renvoie actuellement à `emr-6.10.0-20230905`.

Régions : `emr-6.10.0-latest` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-6.10.0:latest`

emr-6.10.0-20230905

Notes de publication : `6.10.0-20230905` a été publié le 29 septembre 2023. Par rapport à la version précédente, cette version a été actualisée avec des packages Amazon Linux récemment mis à jour et des correctifs critiques.

Régions : `emr-6.10.0-20230905` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-6.10.0:20230905`

emr-6.10.0-20230624

Notes de mise à jour : la version `6.10.0-20230624` a été publiée le 7 juillet 2023. Par rapport à la version précédente, cette version a été actualisée avec des packages Amazon Linux récemment mis à jour et des correctifs critiques.

Régions : `emr-6.10.0-20230624` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-6.10.0:20230624`

emr-6.10.0-20230421

Notes de mise à jour : la version `6.10.0-20230421` a été publiée le 28 avril 2023. Par rapport à la version précédente, cette version a été actualisée avec des packages Amazon Linux récemment mis à jour et des correctifs critiques.

Régions : `emr-6.10.0-20230421` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-6.10.0:20230421`

emr-6.10.0-20230403

Notes de mise à jour : la version `6.10.0-20230403` a été publiée le 12 avril 2023. Par rapport à la version précédente, cette version a été actualisée avec des packages Amazon Linux récemment mis à jour et des correctifs critiques.

Régions : `emr-6.10.0-20230403` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-6.10.0:20230403`

emr-6.10.0-20230220

Notes de mise à jour : la version `emr-6.10.0-20230220` a été publiée le 20 février 2023. Il s'agit de la première version 6.10.0 d'Amazon EMR.

Régions : `emr-6.10.0-20230220` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-6.10.0:20230220`

Versions 6.9.0 d'Amazon EMR on EKS

Les versions 6.9.0 suivantes d'Amazon EMR sont disponibles pour Amazon EMR on EKS. Sélectionnez une version emr-6.9.0-XXXX spécifique pour voir plus de détails tels que la balise de l'image du conteneur correspondant.

- emr-6.9.0-latest
- emr-6.9.0-20230905
- emr-6.9.0-20230624
- emr-6.9.0-20221108
- emr-6.9.0- spark-rapids-latest
- emr-6.9.0-spark-rapids-20230624
- emr-6.9.0-spark-rapids-20221108
- notebook-spark/emr-6.9.0-latest
- notebook-spark/emr-6.9.0-20230624
- notebook-spark/emr-6.9.0-20221108
- notebook-python/emr-6.9.0-latest
- notebook-python/emr-6.9.0-20230624
- notebook-python/emr-6.9.0-20221108

Notes de mise à jour pour Amazon EMR 6.9.0

- Applications prises en charge - AWS SDK pour Java 1.12.331, Spark 3.3.0-amzn-1, Hudi 0.12.1-amzn-0, Iceberg 0.14.1-amzn-0, Delta 2.1.0.
- Composants pris en charge : `aws-sagemaker-spark-sdk`, `emr-ddb`, `emr-goodies`, `emr-s3-select`, `emrfs`, `hadoop-client`, `hudi`, `hudi-spark`, `iceberg`, `spark-kubernetes`.
- Classifications de configuration prises en charge :

  À utiliser avec StartJobRunet CreateManagedEndpoint APIs:

Classifications	Descriptions
`core-site`	Modifiez les valeurs dans le fichier core-site .xml de Hadoop.

Classifications	Descriptions
`emrfs-site`	Modifiez les paramètres EMRFS.
`spark-metrics`	Modifiez les valeurs dans le fichier metrics.properties de Spark.
`spark-defaults`	Modifiez les valeurs dans le fichier spark-defaults.conf de Spark.
`spark-env`	Modifiez les valeurs dans l'environnement Spark.
`spark-hive-site`	Modifiez les valeurs dans le fichier hive-site.xml de Spark.
`spark-log4j`	Modifiez les valeurs dans le fichier log4j.properties de Spark.

À utiliser spécifiquement avec CreateManagedEndpoint APIs:

Classifications	Descriptions
`jeg-config`	Modifiez les valeurs dans le fichier `jupyter_enterprise_gateway_config.py` Jupyter Enterprise Gateway.
`jupyter-kernel-overrides`	Modifiez la valeur de l'image du noyau dans le fichier Jupyter Kernel Spec.

Les classifications de configuration vous permettent de personnaliser les applications. Elles correspondent souvent à un fichier XML de configuration de l'application, tel que `spark-hive-site.xml`. Pour plus d'informations, consultez la rubrique Configuration des applications.

Fonctionnalités notables

- Accélérateur Nvidia RAPIDS pour Apache Spark - Amazon EMR sur EKS pour accélérer Spark à EC2 l'aide de types d'instances d'unités de traitement graphique (GPU). Pour utiliser l'image Spark avec RAPIDS Accelerator, spécifiez l'étiquette de version emr-6.9.0-. spark-rapids-latest Consultez la page de documentation pour en savoir plus.
- Connecteur Spark-Redshift – L'intégration d'Amazon Redshift à Apache Spark est incluse dans les versions 6.9.0 et ultérieures d'Amazon EMR. Auparavant un outil open-source, l'intégration native est un connecteur Spark que vous pouvez utiliser pour créer des applications Apache Spark capables de lire et d'écrire des données sur Amazon Redshift et Amazon Redshift sans serveur. Pour de plus amples informations, veuillez consulter Utilisation de l'intégration Amazon Redshift pour Apache Spark sur Amazon EMR on EKS.
- Delta Lake – Delta Lake est un format de stockage open-source qui permet de créer des lacs de données avec une cohérence transactionnelle, une définition cohérente des jeux de données, des changements dans l'évolution des schémas et la prise en charge des mutations de données. Consultez Utilisation de Delta Lake pour en savoir plus.
- Modifier PySpark les paramètres - Les points de terminaison interactifs prennent désormais en charge la modification des paramètres Spark associés aux PySpark sessions dans le bloc-notes Jupyter d'EMR Studio. Consultez Modifier les paramètres de PySpark session pour en savoir plus.

Problèmes résolus

- Lorsque vous utilisez le connecteur DynamoDB avec Spark sur les versions 6.6.0, 6.7.0 et 6.8.0 d'Amazon EMR, toutes les lectures de votre table renvoient un résultat vide, même si la division d'entrée fait référence à des données non vides. La version 6.9.0 d'Amazon EMR résout ce problème.
- Amazon EMR on EKS 6.8.0 ne remplit pas correctement le hachage de création dans les métadonnées des fichiers Parquet générées à l'aide d'Apache Spark. Ce problème peut entraîner l'échec des outils qui analysent la chaîne de version des métadonnées à partir des fichiers Parquet générés par Amazon EMR on EKS 6.8.0.

Problème connu

- Si vous utilisez l'intégration Amazon Redshift pour Apache Spark et que vous disposez d'une heure, d'un timestamp, d'un timestamp ou d'un timestamptz au format Parquet avec une précision de la microseconde, le connecteur arrondit les valeurs temporelles à la milliseconde la plus proche. Pour contourner le problème, utilisez le paramètre `unload_s3_format` de format de déchargement du texte.

emr-6.9.0-latest

Notes de mise à jour : `emr-6.9.0-latest` renvoie actuellement à `emr-6.9.0-20230905`.

Régions : `emr-6.9.0-latest` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-6.9.0:latest`

emr-6.9.0-20230905

Notes de version : `emr-6.9.0-20230905`. Par rapport à la version précédente, cette version a été actualisée avec les packages Amazon Linux récemment mis à jour et les correctifs critiques.

Régions : `emr-6.9.0-20230905` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-6.9.0:20230905`

emr-6.9.0-20230624

Notes de mise à jour : la version `emr-6.9.0-20230624` a été publiée le 7 juillet 2023.

Régions : `emr-6.9.0-20230624` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-6.9.0:20230624`

emr-6.9.0-20221108

Notes de publication : la version `emr-6.9.0-20221108` a été publiée le 8 décembre 2022. Il s'agit de la première version 6.9.0 d'Amazon EMR.

Régions : `emr-6.9.0-20221108` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-6.9.0:20221108`

Versions 6.8.0 d'Amazon EMR on EKS

Les versions 6.8.0 suivantes d'Amazon EMR sont disponibles pour Amazon EMR on EKS. Sélectionnez une version emr-6.8.0-XXXX spécifique pour voir plus de détails tels que la balise de l'image du conteneur correspondant.

- emr-6.8.0-latest
- emr-6.8.0-20230905
- emr-6.8.0-20230624
- emr-6.8.0-20221219
- emr-6.8.0-20220802

Notes de mise à jour pour Amazon EMR 6.8.0

- Applications prises en charge - AWS SDK pour Java 1.12.170, Spark 3.3.0-amzn-0, Hudi 0.11.1-amzn-0, Iceberg 0.14.0-amzn-0.
- Composants pris en charge : `aws-sagemaker-spark-sdk`, `emr-ddb`, `emr-goodies`, `emr-s3-select`, `emrfs`, `hadoop-client`, `hudi`, `hudi-spark`, `iceberg`, `spark-kubernetes`.
- Classifications de configuration prises en charge :

Classifications	Descriptions
`core-site`	Modifiez les valeurs dans le fichier core-site .xml de Hadoop.
`emrfs-site`	Modifiez les paramètres EMRFS.
`spark-metrics`	Modifiez les valeurs dans le fichier metrics.p roperties de Spark.
`spark-defaults`	Modifiez les valeurs dans le fichier spark-def aults.conf de Spark.
`spark-env`	Modifiez les valeurs dans l'environnement Spark.

Classifications	Descriptions
spark-hive-site	Modifiez les valeurs dans le fichier hive-site .xml de Spark.
spark-log4j	Modifiez les valeurs dans le fichier log4j.pro perties de Spark.

Les classifications de configuration vous permettent de personnaliser les applications. Elles correspondent souvent à un fichier XML de configuration de l'application, tel que spark-hive-site.xml. Pour plus d'informations, consultez la rubrique Configuration des applications.

Fonctionnalités notables

- Spark3.3.0 – Amazon EMR on EKS 6.8 inclut Spark 3.3.0, qui prend en charge l'utilisation d'étiquettes de sélecteur de nœud distinctes pour les pods d'exécuteurs du pilote Spark. Ces nouvelles étiquettes vous permettent de définir les types de nœuds pour les modules pilote et exécuteur séparément dans l' StartJobRun API, sans utiliser de modèles de modules.
  - Propriété du sélecteur de nœud du pilote : spark.kubernetes.driver.node.selector.[labelKey]
  - Propriété du sélecteur de nœud de l'exécuteur : spark.kubernetes.driver.node.selector.[labelKey]
- Amélioration du message d'échec des tâches – Cette version introduit la configuration spark.stage.extraDetailsOnFetchFailures.enabled et spark.stage.extraDetailsOnFetchFailures.maxFailuresToInclude pour suivre les échecs des tâches dus au code de l'utilisateur. Ces informations seront utilisées pour améliorer le message d'échec affiché dans le journal du pilote lorsqu'une étape est interrompue en raison d'un échec de récupération lors du réarrangement.

Nom de la propriété	Valeur par défaut	Signification	Depuis la version
spark.stage.extraDetailsOnFetchFailures.enabled	false	Si elle est définie sur true, cette propriété est utilisée pour améliorer le message d'échec affiché dans le journal du pilote	emr-6.8

Nom de la propriété	Valeur par défaut	Signification	Depuis la version
		lorsqu'une étape est interrompue en raison d'échecs de récupération lors du réarrangement. Par défaut, les 5 derniers échecs de tâches causés par le code utilisateur sont suivis et le message d'erreur de l'échec est ajouté aux journaux des pilotes. Pour augmenter le nombre d'échecs de tâches avec des exceptions utilisateur à suivre, consultez la configuration `spark.stage.extraDetailsOnFetchFailures.maxFailuresToInclude` .	

Nom de la propriété	Valeur par défaut	Signification	Depuis la version
`spark.stage.extraDetailsOnFetchFailures.maxFailuresToInclude`	5	Nombre d'échecs de tâches à suivre par étape et par tentative. Cette propriété est utilisée pour améliorer le message d'échec avec des exceptions utilisateur affiché dans le journal du pilote lorsqu'une étape est interrompue en raison d'échecs de récupération lors du réarrangement. Cette propriété ne fonctionne que si Config spark.stage.extraDetailsOnFetchFailures.enabled est défini sur true.	emr-6.8

Pour plus d'informations, consultez la documentation de configuration d'Apache Spark.

Problème connu

- Amazon EMR on EKS 6.8.0 ne remplit pas correctement le hachage de création dans les métadonnées des fichiers Parquet générées à l'aide d'Apache Spark. Ce problème peut entraîner l'échec des outils qui analysent la chaîne de version des métadonnées à partir des fichiers Parquet générés par Amazon EMR on EKS 6.8.0. Les clients qui analysent la chaîne de version à partir des métadonnées Parquet et qui dépendent du hachage de création doivent passer à une version différente d'Amazon EMR et réécrire le fichier.

Problème résolu

- Fonctionnalité d'interruption du noyau pour les noyaux PySpark – Les charges de travail interactives en cours qui sont déclenchées par l'exécution de cellules dans un bloc-notes peuvent être arrêtées à l'aide de la fonctionnalité `Interrupt Kernel`. Un correctif a été introduit pour que cette fonctionnalité soit disponible pour les noyaux pySpark. Ceci est également disponible en open source sur Changes pour gérer les interruptions pour PySpark Kubernetes Kernel #1115.

emr-6.8.0-latest

Notes de mise à jour : `emr-6.8.0-latest` renvoie actuellement à `emr-6.8.0-20230624`.

Régions : `emr-6.8.0-latest` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-6.8.0:latest`

emr-6.8.0-20230905

Notes de publication : `emr-6.8.0-20230905` a été publié le 29 septembre 2023. Par rapport à la version précédente, cette version a été actualisée avec les packages Amazon Linux récemment mis à jour et les correctifs critiques.

Régions : `emr-6.8.0-20230905` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-6.8.0:20230905`

emr-6.8.0-20230624

Notes de mise à jour : la version `emr-6.8.0-20230624` a été publiée le 7 juillet 2023. Par rapport à la version précédente, cette version a été actualisée avec les packages Amazon Linux récemment mis à jour et les correctifs critiques.

Régions : `emr-6.8.0-20230624` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-6.8.0:20230624`

emr-6.8.0-20221219

Notes de mise à jour : la version `emr-6.8.0-20221219` a été publiée le 19 janvier 2023. Par rapport à la version précédente, cette version a été actualisée avec les packages Amazon Linux récemment mis à jour et les correctifs critiques.

Régions : `emr-6.8.0-20221219` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-6.8.0:20221219`

emr-6.8.0-20220802

Notes de mise à jour : la version `emr-6.8.0-20220802` a été publiée le 27 septembre 2022. Il s'agit de la première version 6.8.0 d'Amazon EMR.

Régions : `emr-6.8.0-20220802` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-6.8.0:20220802`

Versions 6.7.0 d'Amazon EMR on EKS

Les versions 6.7.0 suivantes d'Amazon EMR sont disponibles pour Amazon EMR on EKS. Sélectionnez une version emr-6.7.0-XXXX spécifique pour voir plus de détails tels que la balise de l'image du conteneur correspondant.

- emr-6.7.0-latest
- emr-6.7.0-20240321
- emr-6.7.0-20230624
- emr-6.7.0-20221219
- emr-6.7.0-20220630

Notes de mise à jour pour Amazon EMR 6.7.0

- Applications prises en charge : Spark 3.2.1-amzn-0, Jupyter Enterprise Gateway 2.6, Hudi 0.11-amzn-0, Iceberg 0.13.1.

- Composants pris en charge : `aws-hm-client` (connecteur Glue), `aws-sagemaker-spark-sdk`, `emr-s3-select`, `emrfs`, `emr-ddb`, `hudi-spark`.
- Avec la mise à jour vers JEG 2.6, la gestion du noyau est désormais asynchrone, ce qui signifie que JEG ne bloque pas les transactions lorsqu'un lancement de noyau est en cours. Cela améliore considérablement l'expérience utilisateur en fournissant les éléments suivants :
  - possibilité d'exécuter des commandes dans les blocs-notes en cours d'exécution lorsque d'autres lancements de noyau sont en cours
  - possibilité de lancer plusieurs noyaux simultanément sans affecter les noyaux déjà en cours d'exécution
- Classifications de configuration prises en charge :

Classifications	Descriptions
`core-site`	Modifiez les valeurs dans le fichier Hadoop `core-site.xml` .
`emrfs-site`	Modifiez les paramètres EMRFS.
`spark-metrics`	Modifiez les valeurs dans le fichier Spark `metrics.properties` .
`spark-defaults`	Modifiez les valeurs dans le fichier Spark `spark-defaults.conf` .
`spark-env`	Modifiez les valeurs dans l'environnement Spark.
`spark-hive-site`	Modifiez les valeurs dans le fichier Spark `hive-site.xml` .
`spark-log4j`	Modifiez les valeurs dans le fichier Spark `log4j.properties` .

Les classifications de configuration vous permettent de personnaliser les applications. Elles correspondent souvent à un fichier XML de configuration de l'application, tel que `spark-hive-site.xml`. Pour plus d'informations, consultez Configuration des applications.

Problèmes résolus

- Amazon EMR on EKS 6.7 corrige un problème dans la version 6.6 lors de l'utilisation de la fonctionnalité de modèles de pods d'Apache Spark avec des points de terminaison interactifs. Le problème était présent dans les versions 6.4, 6.5 et 6.6 d'Amazon EMR on EKS. Vous pouvez désormais utiliser des modèles de pods pour définir le mode de démarrage de vos pods de pilotes et d'exécuteurs Spark lorsque vous utilisez des points de terminaison interactifs pour exécuter des analyses interactives.
- Dans les versions précédentes d'Amazon EMR on EKS, la passerelle Jupyter Enterprise Gateway bloquait les transactions lors du lancement du noyau, ce qui entravait l'exécution des sessions de bloc-notes en cours d'exécution. Vous pouvez désormais exécuter des commandes dans les blocs-notes en cours d'exécution lorsque d'autres lancements de noyau sont en cours. Vous pouvez également lancer plusieurs noyaux simultanément sans risquer de perdre la connectivité avec les noyaux déjà en cours d'exécution.

emr-6.7.0-latest

Notes de mise à jour : `emr-6.7.0-latest` renvoie actuellement à `emr-6.7.0-20240321`.

Régions : `emr-6.7.0-latest` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-6.7.0:latest`

emr-6.7.0-20240321

Notes de publication : `emr-6.7.0-20240321` a été publié le 11 mars 2024. Par rapport à la version précédente, cette version a été actualisée avec les packages Amazon Linux récemment mis à jour et les correctifs critiques.

Régions : `emr-6.7.0-20240321` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-6.7.0:20240321`

emr-6.7.0-20230624

Notes de mise à jour : la version `emr-6.7.0-20230624` a été publiée le 7 juillet 2023. Par rapport à la version précédente, cette version a été actualisée avec les packages Amazon Linux récemment mis à jour et les correctifs critiques.

Régions : `emr-6.7.0-20230624` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-6.7.0:20230624`

emr-6.7.0-20221219

Notes de mise à jour : la version `emr-6.7.0-20221219` a été publiée le 19 janvier 2023. Par rapport à la version précédente, cette version a été actualisée avec les packages Amazon Linux récemment mis à jour et les correctifs critiques.

Régions : `emr-6.7.0-20221219` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-6.7.0:20221219`

emr-6.7.0-20220630

Notes de mise à jour : la version `emr-6.7.0-20220630` a été publiée le 12 juillet 2022. Il s'agit de la première version 6.7.0 d'Amazon EMR.

Régions : `emr-6.7.0-20220630` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-6.7.0:20220630`

Versions 6.6.0 d'Amazon EMR on EKS

Les versions 6.6.0 suivantes d'Amazon EMR sont disponibles pour Amazon EMR on EKS. Sélectionnez une version emr-6.6.0-XXXX spécifique pour voir plus de détails tels que la balise de l'image du conteneur correspondant.

- emr-6.6.0-latest
- emr-6.6.0-20240321
- emr-6.6.0-20230624
- emr-6.6.0-20221219
- emr-6.6.0-20220411

Notes de mise à jour pour Amazon EMR 6.6.0

- Applications prises en charge : Spark 3.2.0-amzn-0, Jupyter Enterprise Gateway (endpoints, public preview), Hudi 0.10.1-amzn-0, Iceberg 0.13.1.
- Composants pris en charge : `aws-hm-client` (connecteur Glue), `aws-sagemaker-spark-sdk`, `emr-s3-select`, `emrfs`, `emr-ddb`, `hudi-spark`.
- Classifications de configuration prises en charge :

Classifications	Descriptions
`core-site`	Modifiez les valeurs dans le fichier core-site .xml de Hadoop.
`emrfs-site`	Modifiez les paramètres EMRFS.
`spark-metrics`	Modifiez les valeurs dans le fichier metrics.p roperties de Spark.
`spark-defaults`	Modifiez les valeurs dans le fichier spark-def aults.conf de Spark.
`spark-env`	Modifiez les valeurs dans l'environnement Spark.
`spark-hive-site`	Modifiez les valeurs dans le fichier hive-site .xml de Spark.
`spark-log4j`	Modifiez les valeurs dans le fichier log4j.pro perties de Spark.

Les classifications de configuration vous permettent de personnaliser les applications. Ils correspondent souvent à un fichier XML de configuration de l'application, tel que spark-hive-site .xml. Pour plus d'informations, consultez Configuration des applications.

Problème connu

- La fonctionnalité de modèle de pod Spark avec des points de terminaison interactifs n'est pas disponible dans les versions 6.4, 6.5 et 6.6 d'Amazon EMR on EKS.

Problème résolu

- Les journaux interactifs des points de terminaison sont chargés sur Cloudwatch et S3.

emr-6.6.0-latest

Notes de mise à jour : `emr-6.6.0-latest` renvoie actuellement à `emr-6.6.0-20240321`.

Régions : `emr-6.6.0-latest` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-6.6.0:latest`

emr-6.6.0-20240321

Notes de publication : `emr-6.6.0-20240321` a été publié le 11 mars 2024. Par rapport à la version précédente, cette version a été actualisée avec les packages Amazon Linux récemment mis à jour et les correctifs critiques.

Régions : `emr-6.6.0-20240321` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-6.6.0:20240321`

emr-6.6.0-20230624

Notes de mise à jour : la version `emr-6.6.0-20230624` a été publiée le 27 janvier 2023. Par rapport à la version précédente, cette version a été actualisée avec les packages Amazon Linux récemment mis à jour et les correctifs critiques.

Régions : `emr-6.6.0-20230624` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-6.6.0:20230624`

emr-6.6.0-20221219

Notes de mise à jour : la version `emr-6.6.0-20221219` a été publiée le 27 janvier 2023. Par rapport à la version précédente, cette version a été actualisée avec les packages Amazon Linux récemment mis à jour et les correctifs critiques.

Régions : `emr-6.6.0-20221219` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-6.6.0:20221219`

emr-6.6.0-20220411

Notes de mise à jour : la version `emr-6.6.0-20220411` a été publiée le 20 mai 2022. Il s'agit de la première version 6.6.0 d'Amazon EMR.

Régions : `emr-6.6.0-20220411` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-6.6.0:20220411`

Versions 6.5.0 d'Amazon EMR on EKS

Les versions 6.5.0 suivantes d'Amazon EMR sont disponibles pour Amazon EMR on EKS. Sélectionnez une version emr-6.5.0-XXXX spécifique pour voir plus de détails tels que la balise de l'image du conteneur correspondant.

- emr-6.5.0-latest
- emr-6.5.0-20240321
- emr-6.5.0-20221219
- emr-6.5.0-20220802
- emr-6.5.0-20211119

Notes de mise à jour pour Amazon EMR 6.5.0

- Applications prises en charge : Spark 3.1.2-amzn-1, Jupyter Enterprise Gateway (points de terminaison, version préliminaire publique).
- Composants pris en charge : `aws-hm-client` (connecteur Glue), `aws-sagemaker-spark-sdk`, `emr-s3-select`, `emrfs`, `emr-ddb`, `hudi-spark`.
- Classifications de configuration prises en charge :

Classifications	Descriptions
`core-site`	Modifiez les valeurs dans le fichier core-site .xml de Hadoop.
`emrfs-site`	Modifiez les paramètres EMRFS.
`spark-metrics`	Modifiez les valeurs dans le fichier metrics.p roperties de Spark.
`spark-defaults`	Modifiez les valeurs dans le fichier spark-def aults.conf de Spark.
`spark-env`	Modifiez les valeurs dans l'environnement Spark.
`spark-hive-site`	Modifiez les valeurs dans le fichier hive-site .xml de Spark.
`spark-log4j`	Modifiez les valeurs dans le fichier log4j.pro perties de Spark.

Les classifications de configuration vous permettent de personnaliser les applications. Ils correspondent souvent à un fichier XML de configuration de l'application, tel que spark-hive-site .xml. Pour plus d'informations, consultez Configuration des applications.

Problème connu

- La fonctionnalité de modèle de pod Spark avec des points de terminaison interactifs n'est pas disponible dans les versions 6.4 et 6.5 d'Amazon EMR on EKS.

emr-6.5.0-latest

Notes de mise à jour : `emr-6.5.0-latest` renvoie actuellement à `emr-6.5.0-20240321`.

Régions : `emr-6.5.0-latest` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-6.5.0:latest`

emr-6.5.0-20240321

Notes de publication : `emr-6.5.0-20240321` a été publié le 11 mars 2024. Par rapport à la version précédente, cette version a été actualisée avec les packages Amazon Linux récemment mis à jour et les correctifs critiques.

Régions : `emr-6.5.0-20240321` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-6.5.0:20240321`

emr-6.5.0-20221219

Notes de mise à jour : la version `emr-6.5.0-20221219` a été publiée le 19 janvier 2023. Par rapport à la version précédente, cette version a été actualisée avec les packages Amazon Linux récemment mis à jour et les correctifs critiques.

Régions : `emr-6.5.0-20221219` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-6.5.0:20221219`

emr-6.5.0-20220802

Notes de mise à jour : la version `emr-6.5.0-20220802` a été publiée le 24 août 2022. Par rapport à la version précédente, cette version a été actualisée avec les packages Amazon Linux récemment mis à jour.

Régions : `emr-6.5.0-20220802` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-6.5.0:20220802`

emr-6.5.0-20211119

Notes de mise à jour : la version `emr-6.5.0-20211119` a été publiée le 20 janvier 2022. Il s'agit de la première version 6.5.0 d'Amazon EMR.

Régions : `emr-6.5.0-20211119` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-6.5.0:20211119`

Versions 6.4.0 d'Amazon EMR on EKS

Les versions 6.4.0 suivantes d'Amazon EMR sont disponibles pour Amazon EMR on EKS. Sélectionnez une version emr-6.4.0-XXXX spécifique pour voir plus de détails tels que la balise de l'image du conteneur correspondant.

- emr-6.4.0-latest
- emr-6.4.0-20240321
- emr-6.4.0-20221219

- emr-6.4.0-20210830

Notes de mise à jour pour Amazon EMR 6.4.0

- Applications prises en charge : Spark 3.1.2-amzn-0, Jupyter Enterprise Gateway (points de terminaison, version préliminaire publique).
- Composants pris en charge : `aws-hm-client` (connecteur Glue), `aws-sagemaker-spark-sdk`, `emr-s3-select`, `emrfs`, `emr-ddb`, `hudi-spark`.
- Classifications de configuration prises en charge :

Classifications	Descriptions
`core-site`	Modifiez les valeurs dans le fichier core-site .xml de Hadoop.
`emrfs-site`	Modifiez les paramètres EMRFS.
`spark-metrics`	Modifiez les valeurs dans le fichier metrics.p roperties de Spark.
`spark-defaults`	Modifiez les valeurs dans le fichier spark-def aults.conf de Spark.
`spark-env`	Modifiez les valeurs dans l'environnement Spark.
`spark-hive-site`	Modifiez les valeurs dans le fichier hive-site .xml de Spark.
`spark-log4j`	Modifiez les valeurs dans le fichier log4j.pro perties de Spark.

  Les classifications de configuration vous permettent de personnaliser les applications. Ils correspondent souvent à un fichier XML de configuration de l'application, tel que spark-hive-site .xml. Pour plus d'informations, consultez Configuration des applications.

Problème connu

- La fonctionnalité de modèle de pod Spark avec des points de terminaison interactifs n'est pas disponible dans la version 6.4 d'Amazon EMR on EKS.

emr-6.4.0-latest

Notes de mise à jour : `emr-6.4.0-latest` renvoie actuellement à `emr-6.4.0-20240321`.

Régions : `emr-6.4.0-latest` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-6.4.0:latest`

emr-6.4.0-20240321

Notes de publication : `emr-6.4.0-20240321` a été publié le 11 mars 2024. Par rapport à la version précédente, cette version a été actualisée avec les packages Amazon Linux récemment mis à jour et les correctifs critiques.

Régions : `emr-6.4.0-20240321` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-6.4.0:20240321`

emr-6.4.0-20221219

Notes de mise à jour : la version `emr-6.4.0-20221219` a été publiée le 27 janvier 2023. Par rapport à la version précédente, cette version a été actualisée avec les packages Amazon Linux récemment ajoutés.

Régions : `emr-6.4.0-20221219` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-6.4.0:20221219`

emr-6.4.0-20210830

Notes de publication : la version `emr-6.4.0-20210830` a été publiée le 9 décembre 2021. Il s'agit de la première version 6.4.0 d'Amazon EMR.

Régions : `emr-6.4.0-20210830` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-6.4.0:20210830`

Versions 6.3.0 d'Amazon EMR on EKS

Les versions 6.3.0 suivantes d'Amazon EMR sont disponibles pour Amazon EMR on EKS. Sélectionnez une version emr-6.3.0-XXXX spécifique pour voir plus de détails tels que la balise de l'image du conteneur correspondant.

- emr-6.3.0-latest
- emr-6.3.0-20240321
- emr-6.3.0-20220802
- emr-6.3.0-20211008
- emr-6.3.0-20210802
- emr-6.3.0-20210429

Notes de mise à jour pour Amazon EMR 6.3.0

- Nouvelles fonctionnalités – À partir d'Amazon EMR 6.3.0 de la série de versions 6.x, Amazon EMR on EKS prend en charge la fonctionnalité de modèle de pod de Spark. Vous pouvez également activer la fonctionnalité de rotation du journal d'événements Spark pour Amazon EMR on EKS. Pour plus d'informations, consultez Utilisation de modèles de pods et Utilisation de la rotation des journaux des événements Spark.
- Applications prises en charge : Spark 3.1.1-amzn-0, Jupyter Enterprise Gateway (points de terminaison, version préliminaire publique).
- Composants pris en charge : `aws-hm-client` (connecteur Glue), `aws-sagemaker-spark-sdk`, `emr-s3-select`, `emrfs`, `emr-ddb`, `hudi-spark`.
- Classifications de configuration prises en charge :

Classifications	Descriptions
`core-site`	Modifiez les valeurs dans le fichier core-site .xml de Hadoop.
`emrfs-site`	Modifiez les paramètres EMRFS.
`spark-metrics`	Modifiez les valeurs dans le fichier metrics.p roperties de Spark.
`spark-defaults`	Modifiez les valeurs dans le fichier spark-def aults.conf de Spark.
`spark-env`	Modifiez les valeurs dans l'environnement Spark.
`spark-hive-site`	Modifiez les valeurs dans le fichier hive-site .xml de Spark.
`spark-log4j`	Modifiez les valeurs dans le fichier log4j.pro perties de Spark.

Les classifications de configuration vous permettent de personnaliser les applications. Ils correspondent souvent à un fichier XML de configuration de l'application, tel que spark-hive-site .xml. Pour plus d'informations, consultez Configuration des applications.

emr-6.3.0-latest

Notes de mise à jour : `emr-6.3.0-latest` renvoie actuellement à `emr-6.3.0-20240321`.

Régions : `emr-6.3.0-latest` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-6.3.0:latest`

emr-6.3.0-20240321

Notes de publication : `emr-6.3.0-20240321` a été publié le 11 mars 2024. Par rapport à la version précédente, cette version a été actualisée avec les packages Amazon Linux récemment mis à jour et les correctifs critiques.

Régions : `emr-6.3.0-20240321` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-6.3.0:20240321`

emr-6.3.0-20220802

Notes de mise à jour : la version `emr-6.3.0-20220802` a été publiée le 27 septembre 2022. Par rapport à la version précédente, cette version a été actualisée avec les packages Amazon Linux récemment mis à jour.

Régions : `emr-6.3.0-20220802` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-6.3.0:20220802`

emr-6.3.0-20211008

Notes de publication : la version `emr-6.3.0-20211008` a été publiée le 9 décembre 2021. Par rapport à la version précédente, cette version contient des correctifs et des mises à jour de sécurité.

Régions : `emr-6.3.0-20211008` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-6.3.0:20211008`

emr-6.3.0-20210802

Notes de mise à jour : la version `emr-6.3.0-20210802` a été publiée le 2 août 2021. Par rapport à la version précédente, cette version contient des correctifs et des mises à jour de sécurité.

Régions : `emr-6.3.0-20210802` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-6.3.0:20210802`

emr-6.3.0-20210429

Notes de mise à jour : la version `emr-6.3.0-20210429` a été publiée le 29 avril 2021. Il s'agit de la première version 6.3.0 d'Amazon EMR.

Régions : `emr-6.3.0-20210429` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-6.3.0:20210429`

Versions 6.2.0 d'Amazon EMR on EKS

Les versions 6.2.0 suivantes d'Amazon EMR sont disponibles pour Amazon EMR on EKS. Sélectionnez une version emr-6.2.0-XXXX spécifique pour voir plus de détails tels que la balise de l'image du conteneur correspondant.

- emr-6.2.0-latest
- emr-6.2.0-20240321
- emr-6.2.0-20220802
- emr-6.2.0-20211008
- emr-6.2.0-20210802
- emr-6.2.0-20210615
- emr-6.2.0-20210129
- emr-6.2.0-20201218
- emr-6.2.0-20201201

Notes de mise à jour pour Amazon EMR 6.2.0

- Applications prises en charge : Spark 3.0.1-amzn-0, Jupyter Enterprise Gateway (points de terminaison, version préliminaire publique).

- Composants pris en charge : `aws-hm-client` (connecteur Glue), `aws-sagemaker-spark-sdk`, `emr-s3-select`, `emrfs`, `emr-ddb`, `hudi-spark`.
- Classifications de configuration prises en charge :

Classifications	Descriptions
`core-site`	Modifiez les valeurs dans le fichier core-site .xml de Hadoop.
`emrfs-site`	Modifiez les paramètres EMRFS.
`spark-metrics`	Modifiez les valeurs dans le fichier metrics.p roperties de Spark.
`spark-defaults`	Modifiez les valeurs dans le fichier spark-def aults.conf de Spark.
`spark-env`	Modifiez les valeurs dans l'environnement Spark.
`spark-hive-site`	Modifiez les valeurs dans le fichier hive-site .xml de Spark.
`spark-log4j`	Modifiez les valeurs dans le fichier log4j.pro perties de Spark.

Les classifications de configuration vous permettent de personnaliser les applications. Ils correspondent souvent à un fichier XML de configuration de l'application, tel que spark-hive-site .xml. Pour plus d'informations, consultez Configuration des applications.

emr-6.2.0-latest

Notes de mise à jour : `emr-6.2.0-latest` renvoie actuellement à `emr-6.2.0-20240321`.

Régions : `emr-6.2.0-latest` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-6.2.0:20240321`

emr-6.2.0-20240321

Notes de publication : `emr-6.2.0-20240321` a été publié le 11 mars 2024. Par rapport à la version précédente, cette version a été actualisée avec les packages Amazon Linux récemment mis à jour et les correctifs critiques.

Régions : `emr-6.2.0-20240321` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-6.2.0:20240321`

emr-6.2.0-20220802

Notes de mise à jour : la version `emr-6.2.0-20220802` a été publiée le 27 septembre 2022. Par rapport à la version précédente, cette version a été actualisée avec les packages Amazon Linux récemment mis à jour.

Régions : `emr-6.2.0-20220802` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-6.2.0:20220802`

emr-6.2.0-20211008

Notes de publication : la version `emr-6.2.0-20211008` a été publiée le 9 décembre 2021. Par rapport à la version précédente, cette version contient des correctifs et des mises à jour de sécurité.

Régions : `emr-6.2.0-20211008` est disponible dans les régions : USA Est (Virginie du Nord), USA Ouest (Oregon), Asie-Pacifique (Tokyo), Europe (Irlande), Amérique du Sud (São Paulo).

Balise de l'image du conteneur : `emr-6.2.0:20211008`

emr-6.2.0-20210802

Notes de mise à jour : la version `emr-6.2.0-20210802` a été publiée le 2 août 2021. Par rapport à la version précédente, cette version contient des correctifs et des mises à jour de sécurité.

Régions : `emr-6.2.0-20210802` est disponible dans les régions : USA Est (Virginie du Nord), USA Ouest (Oregon), Asie-Pacifique (Tokyo), Europe (Irlande), Amérique du Sud (São Paulo).

Balise de l'image du conteneur : `emr-6.2.0:20210802`

emr-6.2.0-20210615

Notes de mise à jour : la version `emr-6.2.0-20210615` a été publiée le 15 juin 2021. Par rapport à la version précédente, cette version contient des correctifs et des mises à jour de sécurité.

Régions : `emr-6.2.0-20210615` est disponible dans les régions : USA Est (Virginie du Nord), USA Ouest (Oregon), Asie-Pacifique (Tokyo), Europe (Irlande), Amérique du Sud (São Paulo).

Balise de l'image du conteneur : `emr-6.2.0:20210615`

emr-6.2.0-20210129

Notes de mise à jour : la version `emr-6.2.0-20210129` a été publiée le 29 janvier 2021. Par rapport à la version `emr-6.2.0-20201218`, cette version contient des correctifs et des mises à jour de sécurité.

Régions : `emr-6.2.0-20210129` est disponible dans les régions : USA Est (Virginie du Nord), USA Ouest (Oregon), Asie-Pacifique (Tokyo), Europe (Irlande), Amérique du Sud (São Paulo).

Balise de l'image du conteneur : `emr-6.2.0-20210129`

emr-6.2.0-20201218

Notes de publication : la version `emr-6.2.0-20201218` a été publiée le 18 décembre 2020. Par rapport à la version `emr-6.2.0-20201201`, cette version contient des correctifs et des mises à jour de sécurité.

Régions : `emr-6.2.0-20201218` est disponible dans les régions : USA Est (Virginie du Nord), USA Ouest (Oregon), Asie-Pacifique (Tokyo), Europe (Irlande), Amérique du Sud (São Paulo).

Balise de l'image du conteneur : `emr-6.2.0-20201218`

emr-6.2.0-20201201

Notes de publication : la version `emr-6.2.0-20201201` a été publiée le 1er décembre 2020. Il s'agit de la première version 6.2.0 d'Amazon EMR.

Régions : `emr-6.2.0-20201201` est disponible dans les régions : USA Est (Virginie du Nord), USA Ouest (Oregon), Asie-Pacifique (Tokyo), Europe (Irlande), Amérique du Sud (São Paulo).

Balise de l'image du conteneur : `emr-6.2.0-20201201`

Versions 5.36.0 d'Amazon EMR on EKS

Les versions 5.36.0 suivantes d'Amazon EMR sont disponibles pour Amazon EMR on EKS. Sélectionnez une version emr-5.36.0-XXXX spécifique pour voir plus de détails tels que la balise de l'image du conteneur correspondant.

- emr-5.36.0-latest
- emr-5,36,0-20240321
- emr-5.36.0-20221219
- emr-5.36.0-20220620
- emr-5.36.0-20220525

Notes de mise à jour pour Amazon EMR 5.36.0

- Correction des problèmes de sécurité log4j2.
- Applications prises en charge : Spark 2.4.8-amzn-2, Jupyter Enterprise Gateway (points de terminaison, version préliminaire publique ; le noyau Scala n'est pas pris en charge), livy-0.7.1, fluentd-4.0.0.
- Composants pris en charge - aws-hm-client, emr-ddb aws-sagemaker-spark-sdk, emr-goodies, emr-kinesis, kerberos-server.
- Classifications de configuration prises en charge :

Classifications	Descriptions
`core-site`	Modifiez les valeurs dans le fichier core-site .xml de Hadoop.
`emrfs-site`	Modifiez les paramètres EMRFS.
`spark-metrics`	Modifiez les valeurs dans le fichier metrics.p roperties de Spark.

Classifications	Descriptions
spark-defaults	Modifiez les valeurs dans le fichier spark-defaults.conf de Spark.
spark-env	Modifiez les valeurs dans l'environnement Spark.
spark-hive-site	Modifiez les valeurs dans le fichier hive-site.xml de Spark.
spark-log4j	Modifiez les valeurs dans le fichier log4j.properties de Spark.

Les classifications de configuration vous permettent de personnaliser les applications. Ils correspondent souvent à un fichier XML de configuration de l'application, tel que spark-hive-site .xml. Pour plus d'informations, consultez Configuration des applications.

emr-5.36.0-latest

Notes de mise à jour : `emr-5.36.0-latest` renvoie actuellement à `emr-5.36.0-20240321`.

Régions : `emr-5.36.0-latest` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-5.36.0:latest`

emr-5,36,0-20240321

Notes de publication : `emr-5.36.0-20240321` a été publié le 11 mars 2024. Par rapport à la version précédente, cette version a été actualisée avec les packages Amazon Linux récemment mis à jour et les correctifs critiques.

Régions : `emr-5.36.0-20240321` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-5.36.0:20240321`

emr-5.36.0-20221219

Notes de mise à jour : la version `emr-5.36.0-20221219` a été publiée le 27 janvier 2023. Par rapport à la version précédente, cette version a été actualisée avec les packages Amazon Linux récemment mis à jour.

Régions : `emr-5.36.0-20221219` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-5.36.0:20221219`

emr-5.36.0-20220620

Notes de mise à jour : la version `emr-5.36.0-20220620` a été publiée le 27 juillet 2022. Par rapport à la version précédente, cette version a été actualisée avec les packages Amazon Linux récemment mis à jour.

Régions : `emr-5.36.0-20220620` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-5.36.0:20220620`

emr-5.36.0-20220525

Notes de mise à jour : la version `emr-5.36.0-20220525` a été publiée le 16 juin 2022. Il s'agit de la première version 5.36.0 d'Amazon EMR.

Régions : `emr-5.36.0-20220525` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-5.36.0:20220525`

Versions 5.35.0 d'Amazon EMR on EKS

Les versions 5.35.0 suivantes d'Amazon EMR sont disponibles pour Amazon EMR on EKS. Sélectionnez une version emr-5.35.0-XXXX spécifique pour voir plus de détails tels que la balise de l'image du conteneur correspondant.

- emr-5.35.0-latest
- emr-5.35.0-20240321
- emr-5.35.0-20221219
- emr-5.35.0-20220802
- emr-5.35.0-20220307

Notes de mise à jour pour Amazon EMR 5.35.0

- Correction des problèmes de sécurité log4j2.
- Applications prises en charge : Spark 2.4.8-amzn-1, Hudi 0.9.0-amzn-2, Jupyter Enterprise Gateway (points de terminaison, version préliminaire publique ; le noyau Scala n'est pas pris en charge).
- Composants pris en charge - aws-hm-client (connecteur Glue), emr-s3-select aws-sagemaker-spark-sdk, emrfs, emr-ddb, hudi-Spark.
- Classifications de configuration prises en charge :

Classifications	Descriptions
`core-site`	Modifiez les valeurs dans le fichier core-site .xml de Hadoop.
`emrfs-site`	Modifiez les paramètres EMRFS.
`spark-metrics`	Modifiez les valeurs dans le fichier metrics.p roperties de Spark.
`spark-defaults`	Modifiez les valeurs dans le fichier spark-def aults.conf de Spark.
`spark-env`	Modifiez les valeurs dans l'environnement Spark.
`spark-hive-site`	Modifiez les valeurs dans le fichier hive-site .xml de Spark.

Classifications	Descriptions
`spark-log4j`	Modifiez les valeurs dans le fichier log4j.properties de Spark.

Les classifications de configuration vous permettent de personnaliser les applications. Ils correspondent souvent à un fichier XML de configuration de l'application, tel que spark-hive-site .xml. Pour plus d'informations, consultez Configuration des applications.

emr-5.35.0-latest

Notes de mise à jour : `emr-5.35.0-latest` renvoie actuellement à `emr-5.35.0-20240321`.

Régions : `emr-5.35.0-latest` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-5.35.0:latest`

emr-5.35.0-20240321

Notes de publication : `emr-5.35.0-20240321` a été publié le 11 mars 2024. Par rapport à la version précédente, cette version a été actualisée avec les packages Amazon Linux récemment mis à jour et les correctifs critiques.

Régions : `emr-5.35.0-20240321` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-5.35.0:20240321`

emr-5.35.0-20221219

Notes de mise à jour : la version `emr-5.35.0-20221219` a été publiée le 27 janvier 2023. Par rapport à la version précédente, cette version a été actualisée avec les packages Amazon Linux récemment mis à jour.

Régions : `emr-5.35.0-20221219` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-5.35.0:20221219`

emr-5.35.0-20220802

Notes de mise à jour : la version `emr-5.35.0-20220802` a été publiée le 27 septembre 2022. Par rapport à la version précédente, cette version a été actualisée avec les packages Amazon Linux récemment mis à jour.

Régions : `emr-5.35.0-20220802` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-5.35.0:20220802`

emr-5.35.0-20220307

Notes de mise à jour : la version `emr-5.35.0-20220307` a été publiée le 30 mars 2022. Par rapport à la version précédente, cette version a été actualisée avec les packages Amazon Linux récemment mis à jour.

Régions : `emr-5.35.0-20220307` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-5.35.0:20220307`

Versions 5.34.0 d'Amazon EMR on EKS

Les versions 5.34.0 suivantes d'Amazon EMR sont disponibles pour Amazon EMR on EKS. Sélectionnez une version emr-5.34.0-XXXX spécifique pour voir plus de détails tels que la balise de l'image du conteneur correspondant.

- emr-5.34.0-latest
- emr-5.34.0-20240321
- emr-5.34.0-20220802

Notes de mise à jour pour Amazon EMR 5.34.0

- Applications prises en charge : Spark 2.4.8-amzn-0, Jupyter Enterprise Gateway (points de terminaison, version préliminaire publique ; le noyau Scala n'est pas pris en charge).
- Composants pris en charge : `aws-hm-client` (connecteur Glue), `aws-sagemaker-spark-sdk`, `emr-s3-select`, `emrfs`, `emr-ddb`, `hudi-spark`.
- Classifications de configuration prises en charge :

Classifications	Descriptions
`core-site`	Modifiez les valeurs dans le fichier core-site .xml de Hadoop.
`emrfs-site`	Modifiez les paramètres EMRFS.
`spark-metrics`	Modifiez les valeurs dans le fichier metrics.p roperties de Spark.
`spark-defaults`	Modifiez les valeurs dans le fichier spark-def aults.conf de Spark.
`spark-env`	Modifiez les valeurs dans l'environnement Spark.
`spark-hive-site`	Modifiez les valeurs dans le fichier hive-site .xml de Spark.
`spark-log4j`	Modifiez les valeurs dans le fichier log4j.pro perties de Spark.

Les classifications de configuration vous permettent de personnaliser les applications. Ils correspondent souvent à un fichier XML de configuration de l'application, tel que spark-hive-site .xml. Pour plus d'informations, consultez Configuration des applications.

emr-5.34.0-latest

Notes de mise à jour : `emr-5.34.0-latest` renvoie actuellement à `emr-5.34.0-20220802`.

Régions : `emr-5.34.0-latest` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-5.34.0:latest`

emr-5.34.0-20240321

Notes de publication : `emr-5.34.0-20240321` a été publié le 11 mars 2024. Par rapport à la version précédente, cette version a été actualisée avec les packages Amazon Linux récemment mis à jour et les correctifs critiques.

Régions : `emr-5.34.0-20240321` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-5.34.0:20240321`

emr-5.34.0-20220802

Notes de mise à jour : la version `emr-5.34.0-20220802` a été publiée le 24 août 2022. Par rapport à la version précédente, cette version a été actualisée avec les packages Amazon Linux récemment mis à jour.

Régions : `emr-5.34.0-20220802` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-5.34.0:20220802`

emr-5.34.0-20211208

Notes de mise à jour : la version `emr-5.34.0-20211208` a été publiée le 20 janvier 2022. Par rapport à la version précédente, cette version a été actualisée avec les packages Amazon Linux récemment mis à jour.

Régions : `emr-5.34.0-20211208` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-5.34.0:20211208`

Versions 5.33.0 d'Amazon EMR on EKS

Les versions 5.33.0 suivantes d'Amazon EMR sont disponibles pour Amazon EMR on EKS. Sélectionnez une version emr-5.33.0-XXXX spécifique pour voir plus de détails tels que la balise de l'image du conteneur correspondant.

- emr-5.33.0-latest
- emr-5.33.0-20240321
- emr-5.33.0-20221219
- emr-5.33.0-20220802
- emr-5.33.0-20211008
- emr-5.33.0-20210802
- emr-5.33.0-20210615
- emr-5.33.0-20210323

Notes de mise à jour pour Amazon EMR 5.33.0

- Nouvelle fonctionnalité – À partir d'Amazon EMR 5.33.0 de la série de versions 5.x, Amazon EMR on EKS prend en charge la fonctionnalité de modèle de pod de Spark. Pour de plus amples informations, veuillez consulter Utilisation de modèles de pods.
- Applications prises en charge : Spark 2.4.7-amzn-1, Jupyter Enterprise Gateway (points de terminaison, version préliminaire publique ; le noyau Scala n'est pas pris en charge).
- Composants pris en charge : `aws-hm-client` (connecteur Glue), `aws-sagemaker-spark-sdk`, `emr-s3-select`, `emrfs`, `emr-ddb`, `hudi-spark`.
- Classifications de configuration prises en charge :

Classifications	Descriptions
`core-site`	Modifiez les valeurs dans le fichier core-site .xml de Hadoop.
`emrfs-site`	Modifiez les paramètres EMRFS.
`spark-metrics`	Modifiez les valeurs dans le fichier metrics.p roperties de Spark.

Classifications	Descriptions
spark-defaults	Modifiez les valeurs dans le fichier spark-defaults.conf de Spark.
spark-env	Modifiez les valeurs dans l'environnement Spark.
spark-hive-site	Modifiez les valeurs dans le fichier hive-site.xml de Spark.
spark-log4j	Modifiez les valeurs dans le fichier log4j.properties de Spark.

Les classifications de configuration vous permettent de personnaliser les applications. Ils correspondent souvent à un fichier XML de configuration de l'application, tel que spark-hive-site .xml. Pour plus d'informations, consultez Configuration des applications.

emr-5.33.0-latest

Notes de mise à jour : `emr-5.33.0-latest` renvoie actuellement à `emr-5.33.0-20240321`.

Régions : `emr-5.33.0-latest` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-5.33.0:latest`

emr-5.33.0-20240321

Notes de publication : `emr-5.33.0-20240321` a été publié le 11 mars 2024. Par rapport à la version précédente, cette version a été actualisée avec les packages Amazon Linux récemment mis à jour et les correctifs critiques.

Régions : `emr-5.33.0-20240321` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-5.33.0:20240321`

emr-5.33.0-20221219

Notes de mise à jour : la version `emr-5.33.0-20221219` a été publiée le 19 janvier 2023. Par rapport à la version précédente, cette version a été actualisée avec les packages Amazon Linux récemment mis à jour et les correctifs critiques.

Régions : `emr-5.33.0-20221219` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-5.33.0:20221219`

emr-5.33.0-20220802

Notes de mise à jour : la version `emr-5.33.0-20220802` a été publiée le 24 août 2022. Par rapport à la version précédente, cette version a été actualisée avec les packages Amazon Linux récemment mis à jour.

Régions : `emr-5.33.0-20220802` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-5.33.0:20220802`

emr-5.33.0-20211008

Notes de publication : la version `emr-5.33.0-20211008` a été publiée le 9 décembre 2021. Par rapport à la version précédente, cette version contient des correctifs et des mises à jour de sécurité.

Régions : `emr-5.33.0-20211008` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-5.33.0:20211008`

emr-5.33.0-20210802

Notes de mise à jour : la version `emr-5.33.0-20210802` a été publiée le 2 août 2021. Par rapport à la version précédente, cette version contient des correctifs et des mises à jour de sécurité.

Régions : `emr-5.33.0-20210802` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-5.33.0:20210802`

emr-5.33.0-20210615

Notes de mise à jour : la version `emr-5.33.0-20210615` a été publiée le 15 juin 2021. Par rapport à la version précédente, cette version contient des correctifs et des mises à jour de sécurité.

Régions : `emr-5.33.0-20210615` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-5.33.0:20210615`

emr-5.33.0-20210323

Notes de mise à jour : la version `emr-5.33.0-20210323` a été publiée le 23 mars 2021. Il s'agit de la première version 5.33.0 d'Amazon EMR.

Régions : `emr-5.33.0-20210323` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-5.33.0-20210323`

Versions 5.32.0 d'Amazon EMR on EKS

Les versions 5.32.0 suivantes d'Amazon EMR sont disponibles pour Amazon EMR on EKS. Sélectionnez une version emr-5.32.0-XXXX spécifique pour voir plus de détails tels que la balise de l'image du conteneur correspondant.

- emr-5.32.0-latest
- emr-5.32.0-20240321
- emr-5.32.0-20220802
- emr-5.32.0-20211008

- emr-5.32.0-20210802
- emr-5.32.0-20210615
- emr-5.32.0-20210129
- emr-5.32.0-20201218
- emr-5.32.0-20201201

Notes de mise à jour pour Amazon EMR 5.32.0

- Applications prises en charge : Spark 2.4.7-amzn-0, Jupyter Enterprise Gateway (points de terminaison, version préliminaire publique ; le noyau Scala n'est pas pris en charge).
- Composants pris en charge : `aws-hm-client` (connecteur Glue), `aws-sagemaker-spark-sdk`, `emr-s3-select`, `emrfs`, `emr-ddb`, `hudi-spark`.
- Classifications de configuration prises en charge :

Classifications	Descriptions
`core-site`	Modifiez les valeurs dans le fichier core-site .xml de Hadoop.
`emrfs-site`	Modifiez les paramètres EMRFS.
`spark-metrics`	Modifiez les valeurs dans le fichier metrics.p roperties de Spark.
`spark-defaults`	Modifiez les valeurs dans le fichier spark-def aults.conf de Spark.
`spark-env`	Modifiez les valeurs dans l'environnement Spark.
`spark-hive-site`	Modifiez les valeurs dans le fichier hive-site .xml de Spark.
`spark-log4j`	Modifiez les valeurs dans le fichier log4j.pro perties de Spark.

Les classifications de configuration vous permettent de personnaliser les applications. Ils correspondent souvent à un fichier XML de configuration de l'application, tel que spark-hive-site .xml. Pour plus d'informations, consultez Configuration des applications.

emr-5.32.0-latest

Notes de mise à jour : `emr-5.32.0-latest` renvoie actuellement à `emr-5.32.0-20240321`.

Régions : `emr-5.32.0-latest` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-5.32.0:latest`

emr-5.32.0-20240321

Notes de publication : `emr-5.32.0-20240321` a été publié le 11 mars 2024. Par rapport à la version précédente, cette version a été actualisée avec les packages Amazon Linux récemment mis à jour et les correctifs critiques.

Régions : `emr-5.32.0-20240321` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-5.32.0:20240321`

emr-5.32.0-20220802

Notes de mise à jour : la version `emr-5.32.0-20220802` a été publiée le 24 août 2022. Par rapport à la version précédente, cette version a été actualisée avec les packages Amazon Linux récemment mis à jour.

Régions : `emr-5.32.0-20220802` est disponible dans toutes les régions prises en charge par Amazon EMR on EKS. Pour plus d'informations, consultez la rubrique Points de terminaison de service Amazon EMR on EKS.

Balise de l'image du conteneur : `emr-5.32.0:20220802`

emr-5.32.0-20211008

Notes de publication : la version `emr-5.32.0-20211008` a été publiée le 9 décembre 2021. Par rapport à la version précédente, cette version contient des correctifs et des mises à jour de sécurité.

Régions : `emr-5.32.0-20211008` est disponible dans les régions : USA Est (Virginie du Nord), USA Ouest (Oregon), Asie-Pacifique (Tokyo), Europe (Irlande), Amérique du Sud (São Paulo).

Balise de l'image du conteneur : `emr-5.32.0:20211008`

emr-5.32.0-20210802

Notes de mise à jour : la version `emr-5.32.0-20210802` a été publiée le 2 août 2021. Par rapport à la version précédente, cette version contient des correctifs et des mises à jour de sécurité.

Régions : `emr-5.32.0-20210802` est disponible dans les régions : USA Est (Virginie du Nord), USA Ouest (Oregon), Asie-Pacifique (Tokyo), Europe (Irlande), Amérique du Sud (São Paulo).

Balise de l'image du conteneur : `emr-5.32.0:20210802`

emr-5.32.0-20210615

Notes de mise à jour : la version `emr-5.32.0-20210615` a été publiée le 15 juin 2021. Par rapport à la version précédente, cette version contient des correctifs et des mises à jour de sécurité.

Régions : `emr-5.32.0-20210615` est disponible dans les régions : USA Est (Virginie du Nord), USA Ouest (Oregon), Asie-Pacifique (Tokyo), Europe (Irlande), Amérique du Sud (São Paulo).

Balise de l'image du conteneur : `emr-5.32.0:20210615`

emr-5.32.0-20210129

Notes de mise à jour : la version `emr-5.32.0-20210129` a été publiée le 29 janvier 2021. Par rapport à la version `emr-5.32.0-20201218`, cette version contient des correctifs et des mises à jour de sécurité.

Régions : `emr-5.32.0-20210129` est disponible dans les régions : USA Est (Virginie du Nord), USA Ouest (Oregon), Asie-Pacifique (Tokyo), Europe (Irlande), Amérique du Sud (São Paulo).

Balise de l'image du conteneur : `emr-5.32.0-20210129`

emr-5.32.0-20201218

Notes de publication : la version `5.32.0-20201218` a été publiée le 18 décembre 2020. Par rapport à la version `5.32.0-20201201`, cette version contient des correctifs et des mises à jour de sécurité.

Régions : `emr-5.32.0-20201218` est disponible dans les régions : USA Est (Virginie du Nord), USA Ouest (Oregon), Asie-Pacifique (Tokyo), Europe (Irlande), Amérique du Sud (São Paulo).

Balise de l'image du conteneur : `emr-5.32.0-20201218`

emr-5.32.0-20201201

Notes de publication : la version `5.32.0-20201201` a été publiée le 1er décembre 2020. Il s'agit de la première version 5.32.0 d'Amazon EMR.

Régions : `5.32.0-20201201` est disponible dans les régions : USA Est (Virginie du Nord), USA Ouest (Oregon), Asie-Pacifique (Tokyo), Europe (Irlande), Amérique du Sud (São Paulo).

Balise de l'image du conteneur : `emr-5.32.0-20201201`

Historique du document

Le tableau suivant décrit les modifications importantes apportées à la documentation depuis la dernière version d'Amazon EMR on EKS. Pour plus d'informations sur les mises à jour de cette documentation, vous pouvez vous abonner à un flux RSS.

Modifier	Description	Date
Ajout d'informations sur la version 7.12.0 d'EMR	Ajout d'informations sur la version 7.12.0 d'EMR avec Iceberg Materialized Views et Hudi Full Table Access	Novembre 2025
Ajout d'informations sur la version 7.11.0 d'EMR	Ajout d'informations sur la version 7.11.0 d'EMR avec intégration d'Unified Studio SageMaker	Novembre 2025
Mise à jour du contenu	Politiques gérées pour Amazon EMR on EKS— Autorisations supplémentaires pour`AmazonEMRContainersServiceRolePolicy` .	03 février 2025
Nouvelle version	Versions d'Amazon EMR on EKS 7.6.0	10 janvier 2025
Nouvelle version	Publications d'Amazon EMR on EKS 7.5.0	21 novembre 2024
Nouvelle version	Versions d'Amazon EMR on EKS 7.4.0	13 novembre 2024
Nouvelle version	Versions d'Amazon EMR on EKS 7.3.0	16 octobre 2024
Nouvelle version	Amazon EMR on EKS versions 7.2.0	25 juillet 2024
Nouvelle version	Versions d'Amazon EMR on EKS 7.1.0	17 avril 2024
Nouvelle version	Versions 7.0.0 d'Amazon EMR sur EKS	22 décembre 2023
Nouvelle version	Versions 6.15.0 d'Amazon EMR sur EKS	17 novembre 2023

Modifier	Description	Date
Nouvelle version	Versions 6.14.0 d'Amazon EMR sur EKS	17 octobre 2023
Mise à jour du contenu	Renommage des « points de terminaison gérés » en points de terminaison interactifs ; disponibilité générale des points de terminaison interactifs	29 septembre 2023
Nouvelle version	Versions 6.13.0 d'Amazon EMR on EKS et des documents de prévisualisation publics pour Exécution de tâches Flink à l'aide d'Amazon EMR on EKS	12 septembre 2023
Nouvelle version	Versions 6.12.0 d'Amazon EMR on EKS	21 juillet 2023
Nouveau contenu	Ajout de Utilisation de Volcano comme planificateur personnalisé pour Apache Spark sur Amazon EMR on EKS.	13 juin 2023
Nouveau contenu	Ajout de Utilisation de Volcano comme planificateur personnalisé pour Apache Spark sur Amazon EMR on EKS.	13 juin 2023
Nouveau contenu	Ajout de Utilisation de la rotation des journaux des conteneurs Spark.	12 juin 2023
Mise à jour du contenu	Mise à jour de la documentation relative aux images personnalisées pour trouver les informations relatives à l'image de base dans la galerie publique d'Amazon ECR.	8 juin 2023
Nouvelle version	Versions 6.11.0 d'Amazon EMR on EKS	8 juin 2023

Modifier	Description	Date
Nouveau contenu	Ajout de Exécution de tâches Spark à l'aide de l'opérateur Spark et réorganisation des sections d'exécution de tâches sous Exécution de tâches Spark avec Amazon EMR sur EKS.	5 juin 2023
Nouveau contenu	Ajout de deux sections : Utilisation de la mise à l'échelle automatique verticale avec les tâches Spark sur Amazon EMR et Utilisation des blocs-notes Jupyter auto-hébergés	4 mai 2023
Page d'historique des documents	Création d'une page d'historique des documents pour Amazon EMR on EKS.	13 mars 2023
Page de politiques gérées	Création d'une page de politiques gérées pour Amazon EMR on EKS.	13 mars 2023

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.